

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

LEARNING 3D PHYSICS FROM MULTI-VIEW VIDEOS

JINXI LI

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University
Department of Computing

Learning 3D Physics from Multi-view Videos

Jinxi Li

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
August 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Jinxi Li

Abstract

Modeling dynamic 3D scenes with physical accuracy and predicting their future states is critical for applications in augmented reality, robotics, and content creation. While recent advances in implicit neural representations have achieved remarkable results in reconstructing dynamic geometry and appearance, they primarily excel at interpolating within observed time and fail to extrapolate future frames due to a lack of explicit physical modeling. Early physics-integrated methods, such as those using Physics-Informed Neural Networks (PINNs) and numerical simulations, face limitations including poor generalizability, heavy computational overhead, and training instability, hindering their practical deployment.

This thesis addresses three core challenges in dynamic 3D scene modeling: ensuring physical plausibility in future predictions, improving efficiency in physics-aware modeling, and enabling explicit learning of underlying physical parameters. To tackle these challenges, we present three interconnected frameworks that progressively advance the state-of-the-art: In Chapter 3, we introduce NVFi, the first framework fusing dynamic radiance fields with mass and momentum conservation laws by PINNs loss functions. NVFi learns disentangled, physically consistent velocity fields from multi-view videos, eliminating unrealistic motion artifacts without relying on additional annotations. In Chapter 4, we propose FreeGave to address PINNs’ inefficiencies. It incorporates a divergence-free module to enforce local physical constraints without global PDE sampling and uses learnable physics encoding to decompose complex

motions, achieving efficient and stable physics-aware modeling for complex scenes. In Chapter 5, we develop TRACE, which shifts from implicit neural fitting to explicit physical parameter learning. By modeling each 3D Gaussian as a rigid particle with independent translation-rotation dynamics systems, TRACE directly learns core physical parameters and enables controllable future extrapolation while eliminating neural ODE integration overhead.

Collectively, these contributions advance dynamic 3D scene modeling from physics-unaware interpolation to accurate, efficient, and controllable future frame extrapolation. By progressively integrating physical constraints, optimizing computational efficiency, and enabling explicit parameter learning, this thesis establishes a robust framework for machines to perceive and predict dynamic 3D world dynamics, laying the foundation for next-generation intelligent systems interacting with physical environments.

Publications Arising from the Thesis

1. Jinxi Li, Ziyang Song, and Bo Yang, “NVFi: Neural velocity fields for 3d physics learning from dynamic videos”, in *Proceedings of Advances in Neural Information Processing Systems*, 2023.
2. Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang, “FreeGave: 3d physics learning from dynamic videos by gaussian velocity”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
3. Jinxi Li, Ziyang Song, and Bo Yang, “TRACE: learning 3d gaussian physical dynamics from multi-view videos”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Beyond the publications directly stemming from the core research work of this thesis, during my pursuit of the doctoral degree, I also contributed to the following research projects, which will not be covered in this thesis. (* denotes equal contribution)

4. Zhuoman Liu, Bo Yang, Yan Luximon, Ajay Kumar, Jinxi Li, “RayDF: Neural Ray-surface Distance Fields with Multi-view Consistency”, in *Proceedings of Advances in Neural Information Processing Systems*, 2023.
5. Ziyang Song, Jinxi Li, Bo Yang, “OSN: Infinite Representations of Dynamic 3D Scenes from Monocular Videos”, in *International Conference on Machine Learning*, 2024.

6. Shenxing Wei*, Jinxi Li*, Yafei Yang, Siyuan Zhou, Bo Yang, “RayletDF: Raylet Distance Fields for Generalizable 3D Surface Reconstruction from Point Clouds or Gaussians”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision, Highlight*, 2025.

Acknowledgments

As I look back on these four years of my PhD journey, every experience feels as vivid as if it happened yesterday. I want to express my gratitude to many people who have supported me along the way.

I must express my gratitude to my supervisor, Prof. YANG Bo. He has always guided my research with immense patience. Most importantly, during those moments when I repeatedly doubted my direction and future, he was always there to firmly support me and encourage me to keep going.

I want to thank my family. Their meticulous love and unwavering support have allowed me to pursue my dreams without any worries or distractions.

I am deeply grateful to my girlfriend. She has always had full confidence in me. It turns out that the reason to keep believing, the reason to support me struggling forward, is love.

My thanks go to my peers in the lab as well, especially Ziyang, Zihui and Yafei. The success is not about solo efforts, but about moving forward with the help of others; the journey is not a lonely melody, but a handful of memories woven by friendship.

I must mention my dog too. Through him, I have come to believe that pure and dustless things truly exist in this world.

Finally, I want to thank myself. I am grateful for all my past experiences that have shaped who I am today. I appreciate my consistent pursuit of mathematical truth

and my enduring motivation - to seek the reality of the world.

Table of Contents

Abstract	i
Publications Arising from the Thesis	iii
Acknowledgments	v
List of Figures	xi
List of Tables	xv
1 Introduction	1
1.1 Motivation	1
1.2 Research Challenges and Objectives	2
1.3 Contributions	4
1.4 Thesis Structure	7
2 Literature Review	8
2.1 Static 3D Representation Learning	8
2.2 Dynamic 3D Reconstruction	10

2.3	3D Physics Learning	11
2.4	Video Prediction	12
3	General Physics-Constrained Velocity Fields	13
3.1	Introduction	13
3.2	NVFi	16
3.2.1	Overview	16
3.2.2	Module Architectures	17
3.2.3	Optimization of Keyframe Dynamic Radiance Fields	19
3.2.4	Optimization of Interframe Velocity Fields	21
3.2.5	Joint Keyframe and Interframe Optimization	23
3.3	Experiments	25
3.3.1	Evaluation of Future Frame Extrapolation	28
3.3.2	Evaluation of 3D Semantic Scene Decomposition	29
3.3.3	Evaluation of Motion Transfer	33
3.3.4	Ablation Study	35
3.4	Conclusion	37
4	Efficient Physics Learning via Divergence-free Velocity Fields	39
4.1	Introduction	39
4.2	FreeGave	42
4.2.1	Canonical 3D Representation Module	43
4.2.2	Divergence-free Gaussian Velocity Module	43

4.2.3	Deformation-aided Optimization	47
4.3	Experiments	49
4.3.1	Evaluation for Future Frame Extrapolation	53
4.3.2	Analysis of Physics Code and Motion Patterns	62
4.3.3	Ablation Study	65
4.4	Conclusion	67
5	Translation-Rotation Dynamics System for Physics Learning	68
5.1	Introduction	68
5.2	TRACE	72
5.2.1	Preliminary	73
5.2.2	Translation Rotation Dynamics System	74
5.2.3	Training	77
5.3	Experiments	79
5.3.1	Main Results of Future Frame Extrapolation	81
5.3.2	Continual Learning	86
5.3.3	Analysis of Dynamics Parameters	88
5.3.4	Ablation Study	96
5.4	Conclusion	97
6	Conclusion and Future Work	99
6.1	Conclusion	99
6.2	Broader Impact and Future Directions	101

6.2.1	Physics-Aware Robot Policy Learning	101
6.2.2	Physics-Guided Video Generation	102

List of Figures

1.1	The main task and three major works of this thesis.	4
3.1	The three major components of our framework: Keyframe Dynamic Radiance Field, Interframe Velocity Field, and the Joint Keyframe and Interframe Optimization module.	15
3.2	The left block illustrates the network architecture of our keyframe dynamic radiance field based on HexPlane [10], and the right block shows the architecture of our interframe velocity field.	17
3.3	Qualitative results on NVIDIA Dynamic Scene dataset.	30
3.4	Qualitative results on synthetic smoke scene.	30
3.5	The optimization pipeline of our method for scene decomposition. The solid black lines represent forward inference while the dashed gray lines represent backward supervision.	30
3.6	Qualitative results for objectwise motion transfer.	34
3.7	Qualitative results of baselines and our method on the three tasks. . .	38

4.1	Given video frames of a real-world dynamic scene, our FreeGave can learn the underlying physics and accurately predict the future motion of a purple pen going through the hole of a tape, while existing baselines cannot.	40
4.2	The leftmost block shows the canonical 3D Gaussians module, the middle bottom block shows the deformation-aided optimization module and the right block shows our divergence-free Gaussian velocity module.	42
4.3	An illustration of the network architecture to learn basic velocity component $\mathbb{V}_t$. Each Gaussian kernel is assigned to different learned motion patterns.	45
4.4	The camera settings for our dataset collection.	51
4.5	Qualitative results for future frame extrapolation on ParticleNeRF Dataset.	56
4.6	Qualitative results for future frame extrapolation on ParticleNeRF Dataset.	57
4.7	Qualitative results of our method and baselines for future frame extrapolation and unsupervised motion segmentation.	58
4.8	Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.	59
4.9	Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.	60
4.10	Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.	61
4.11	Qualitative results of decomposed motion patterns (colored objects) and their corresponding velocity fields (colored trajectories).	62

4.12	Qualitative results for unsupervised motion segmentation on "Chess-board" of Dynamic Indoor Scene Dataset.	64
5.1	Given video frames of a real-world dynamic scene, our TRACE can learn the underlying physics and accurately predict the future motion.	69
5.2	The leftmost block shows the canonical 3D Gaussians module, the middle block shows the deformation field, and the rightmost block shows our translation rotation dynamics system module.	71
5.3	The proposed translation rotation dynamics system for a specific rigid particle. The rigid particle will be driven by its learned physical parameters over time, forming a trajectory in 3D space.	74
5.4	Qualitative results of RGB view synthesis for future frame extrapolation on four datasets.	83
5.5	The top row shows extrapolation results over future timestamps and the bottom shows qualitative results of our extrapolation.	84
5.6	Qualitative results of RGB view synthesis for longer extrapolation from our method.	85
5.7	Extrapolation over different scene motion complexities.	86
5.8	Quantitative Extrapolation Results (PSNR) of Continual Learning on Panoptic Sports Dataset.	89
5.9	Qualitative Results of Continual Learning on Panoptic Sports Dataset.	89
5.10	Qualitative extrapolation results for continual learning on ParticleNeRF dataset.	90
5.11	Clustering of translation rotation physics parameters.	90
5.12	Qualitative results of segmentation.	92

5.13 Qualitative results for Object/Part Segmentation on Dynamic Object dataset.	93
5.14 Qualitative results for Object/Part Segmentation on Dynamic Multipart dataset.	94
5.15 Qualitative results for object segmentation on "Factory" of Dynamic Indoor Scene dataset.	95

List of Tables

3.1	Quantitative results for both novel view interpolation and future frame extrapolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset.	28
3.2	Quantitative results on the NVIDIA Dynamic Scene dataset.	28
3.3	Quantitative results of scene decomposition on Synthetic Indoor Scene dataset.	33
3.4	Quantitative results of motion transfer on Synthetic Indoor Scene dataset.	35
3.5	Quantitative results of ablation studies on Dynamic Object dataset. .	36
3.6	Quantitative results of ablation studies on Dynamic Object dataset. .	36
4.1	Quantitative results of all methods for both novel view interpolation and future frame extrapolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset, and future frame extrapolation on ParticleNeRF Dataset. Bold numbers indicate the best performance.	54
4.2	Quantitative results for both novel view interpolation and future frame extrapolation on FreeGave-GoPro Dataset.	54
4.3	Results on NVIDIA Dynamic Scene Dataset.	55

4.4	The average time (hours) and GPU memory (GB) cost for training, the inference speed (fps), GPU memory (GB), and model size (MB) for test on all three datasets.	55
4.5	Quantitative results of scene decomposition on the Synthetic Indoor Scene Dataset.	63
4.6	Complete ablation study results on both Dynamic Object Dataset and Dynamic Indoor Scene Dataset	66
5.1	Quantitative results of all methods for both future frame extrapolation and novel view interpolation on Dynamic Object and Dynamic Indoor Scene Dataset.	82
5.2	Quantitative results of all methods for both future frame extrapolation and novel view interpolation on NVIDIA Dynamic Scene and Dynamic Multipart Dataset	82
5.3	Quantitative interpolation results (PSNR) of continual learning on ParticleNeRF Dataset.	88
5.4	Quantitative extrapolation results (PSNR) of continual learning on ParticleNeRF Dataset.	88
5.5	Quantitative results of motion segmentation results on Dynamic Indoor Scene dataset.	92
5.6	Segmentation results on basketball scene of Panoptic Sports Dataset.	92
5.7	Ablation studies on three datasets.	97

Chapter 1

Introduction

1.1 Motivation

The 3D world we inhabit is inherently dynamic—objects fall under gravity, vehicles navigate paths, and mechanical systems operate with rhythmic motions. Humans possess an innate ability to perceive the geometry, appearance, and underlying physical properties of these dynamic scenes through brief observation, enabling accurate prediction of future states based on learned physical rules. Translating this capability to machines is transformative for numerous cutting-edge applications: augmented reality systems requiring realistic scene interactions, robotics platforms needing to anticipate environmental changes, game engines demanding lifelike physics simulations, and film production seeking efficient dynamic content generation.

Recent advancements in implicit neural representations have revolutionized dynamic 3D scene modeling. Neural Radiance Fields (NeRF) [76] and its variants [86, 7, 10, 30] have achieved remarkable results in capturing time-varying geometry and appearance, while 3D Gaussian Splatting (3DGS) [49] and extensions [137, 125] further enhanced reconstruction fidelity for deformable objects [7, 110] and articulated structures [79, 104]. However, these methods primarily excel at interpolating novel views within

observed time windows, as they focus on fitting training sequences without explicitly encoding physical properties like velocities. This critical limitation prevents them from extrapolating future motion patterns—a fundamental requirement for predictive applications in real-world scenarios.

Early efforts to integrate physical priors into neural representations [89, 21, 14, 57, 4] introduced numerical differentiable simulations or Physics-Informed Neural Networks (PINNs) [91] to encode governing partial differential equations (PDEs), showing promise in modeling velocity and viscosity fields for simple scenes like smoke or basic continuum object motions. Despite these advances, significant barriers remain: learned physical properties are often object-specific [21], relying on additional foreground mask [57], lacking cross-scene transferability, and suffering from reconstruction artifacts at boundary regions. Moreover, PINN-based methods incur heavy computational overhead due to global spatiotemporal sampling for PDE validation and face training instability when balancing physical constraints with geometric reconstruction. These limitations underscore the need for more robust frameworks that can learn disentangled physical properties from multi-view videos alone, enabling accurate and efficient future frame prediction.

1.2 Research Challenges and Objectives

Modeling dynamic 3D scenes with accurate physical extrapolation capabilities presents three core challenges:

- **Physical plausibility in future prediction.** Traditional dynamic radiance fields and Gaussian splatting methods lack explicit physical constraints, leading to unrealistic drifts or distortions (*e.g.*, falling objects reversing direction) when extrapolating beyond observed sequences. Existing physics-integrated methods either fail to disentangle physical properties from scene geometry or require excessive manual annotations, limiting their generalizability.

- **Efficiency in physics-aware modeling.** PINN-based approaches, while effective in enforcing physical laws, introduce prohibitive computational costs through global PDE validation sampling. Additionally, balancing multi-objective training (geometry reconstruction vs. physics enforcement) leads to instability, making these methods impractical for complex real-world scenes.
- **Explicit physical parameter learning.** Prior methods rely on neural network fitting for motion and acceleration patterns on time, resulting in degraded controllability for future predictions and high inference overhead due to complex neural ordinary differential equation integration. Reverse-deriving underlying physical parameters (*e.g.*, rotation centers, motion dynamics) from visual inputs remains challenging, especially for deformable self-propelled objects and scenes with multiple objects exhibiting diverse motions.

Motivated by these challenges, this thesis aims to achieve the corresponding three primary research objectives:

- The first objective is to develop a physics-constrained dynamic modeling framework that can learn disentangled geometry, appearance, and physical velocity fields purely from multi-view RGB videos, without relying on additional annotations, thereby ensuring physical plausibility in future frame predictions and eliminating unrealistic motion artifacts. This objective is studied in Chapter 3.
- The second objective is to design an efficient physics learning paradigm that avoids the computationally heavy sampling in PINNs, while maintaining physical consistency through intrinsic constraints, and simplifies training by reducing the complexity of balancing conflicting objectives, making physics-aware modeling feasible for complex scenes. This objective is studied in Chapter 4.
- The third objective is to establish an explicit dynamics modeling framework that can reverse-derive underlying physical parameters from visual inputs, enabling

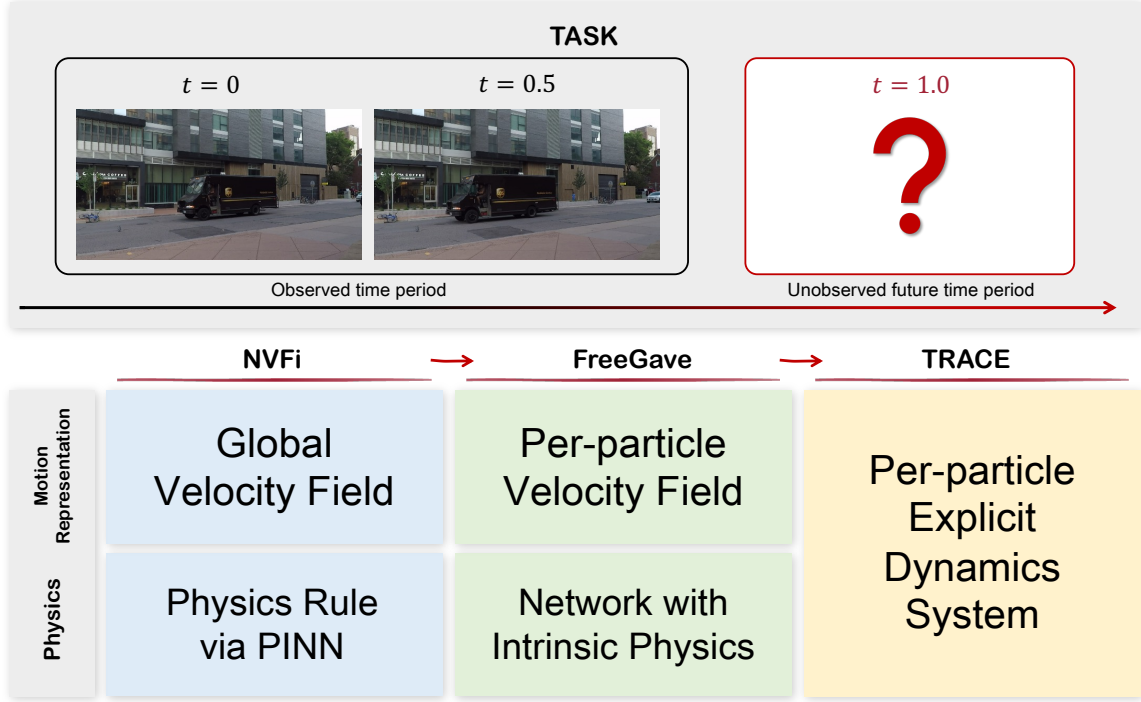


Figure 1.1: The main task and three major works of this thesis.

controllable extrapolation and reducing inference overhead by replacing complex neural ODE integration with explicit physical rule-based prediction. This objective is studied in Chapter 5.

1.3 Contributions

To address these challenges, this thesis presents three interconnected contributions that progressively advance the state-of-the-art in dynamic 3D scene modeling, as shown in Figure 1.1:

- In Chapter 3, we introduce NVFi, the first framework to fuse dynamic radiance fields with Physics-Informed Neural Networks (PINNs), unifying appearance modeling and physical constraint enforcement. Its core innovation is *physics-*

constrained neural velocity field estimation, which encodes mass and momentum conservation laws into loss functions via PINNs, compelling the model to learn physically consistent velocity fields. The framework comprises three components: (1) a keyframe dynamic radiance field capturing time-dependent volume density and appearance; (2) an interframe velocity field modeling 3D point velocities over time; (3) a joint optimization scheme integrating photometric losses with PDE-based physics constraints. NVFi enables disentangled velocity field learning without requiring object masks or material annotations, laying the foundation for physics-aware future prediction. This work is published in:

Jinxi Li, Ziyang Song, and Bo Yang. NVFi: Neural velocity fields for 3d physics learning from dynamic videos. *Proceedings of Advances in Neural Information Processing Systems*, 2023.[53]

- Building on NVFi’s validation of physical constraints, we address PINNs’ inefficiencies with FreeGave in Chapter 4, a framework enabling efficient physics learning without global PDE sampling. FreeGave introduces two key innovations: (1) A divergence-free module leveraging mathematical field decomposition to ensure velocity fields satisfy mass and local volume conservation at every spatial point, eliminating the need for global PDE validation; (2) Learnable physics code that decomposes complex motions into distinct patterns, enhancing velocity representation efficiency and reducing training complexity. By treating scenes composed by rigid particles with latent physics codes, FreeGave maintains physical plausibility while achieving superior reconstruction performance, making physics-aware modeling feasible for complex scenes. This work is published in:

Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang. FreeGave: 3d physics learning from dynamic videos by gaussian velocity. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. [55]

- To enable accurate controllable extrapolation, we propose TRACE in Chapter 5, a framework that explicitly learns underlying physical parameters through particle-based translation-rotation dynamics modeling. TRACE represents 3D scenes by rigid particles and learns independent dynamics systems, enabling reverse derivation of physical parameters. A critical innovation is the introduction of equivalent translation and rotation systems to mitigate coupling issues caused by unknown rotation centers, reducing dependency on accurate rotation center estimation. By training auxiliary deformation fields to address training instability, TRACE optimizes physical parameter learning effectively. Unlike interpolation-focused methods, TRACE achieves unprecedented extrapolation accuracy by explicitly modeling the physical rules governing motion dynamics. This work is published in:

Jinxi Li, Ziyang Song, and Bo Yang. TRACE: learning 3d gaussian physical dynamics from multi-view videos. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. [54]

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 presents a comprehensive review of static/dynamic 3D reconstruction and physics learning.
- Chapter 3 introduces our NVFi, which introduces mass and momentum conservation laws into dynamic reconstruction via PINN losses to learn physically plausible velocity fields for future frame prediction.
- Chapter 4 presents our FreeGave, which achieves efficient physics-aware modeling through a divergence-free module and learnable physics encoding, avoiding global PINN computations while maintaining physical consistency.
- Chapter 5 presents our TRACE, which proposes a particle-based translation-rotation dynamics system to explicitly learn underlying physical parameters, enabling accurate future frame extrapolation with controllable motion evolution.
- The last Chapter 6 concludes the entire thesis and identifies future directions.

Chapter 2

Literature Review

2.1 Static 3D Representation Learning

Static 3D objects and scenes have been explored through various representation paradigms in computer vision and graphics. Traditional explicit representations include voxels [20, 132, 133], point clouds [25, 145], octrees [107, 118], meshes [47, 34], and primitives [152, 143]. They are widely used in representing various objects and scenes, such as humans [106] and furniture [147]. However, these representations suffer from inherent discretization issues, where the fidelity of 3D shapes is fundamentally limited by spatial resolution constraints and memory footprint limitations.

Inspired by seminal advances in neural representation learning [17, 74, 81], recent research has shifted toward implicit neural functions for modeling 3D data. These implicit methods are broadly categorized into several types: 1) occupancy fields (OF) [17, 74], which regress the occupancy status of 3D points; 2) signed distance fields (SDF) [81], unsigned distance fields (UDF) [19, 115, 94], and their variant scaled-squared distance function (S^2DF) [134], which predict the distance from a point to the nearest surface; and 3) radiance fields (NeRF) [76, 6, 111], which model both geometry and appearance by regressing color and density. Fundamentally, these implicit neural

representations take 3D point coordinates as input to neural networks and directly regress diverse point-level information—such as occupancy, surface distance, color, and even semantics [149, 114]. Compared to traditional explicit representations, they enable continuous modeling of shape and appearance with a significantly reduced memory footprint, marking a key advantage in high-fidelity 3D reconstruction.

Despite their strengths in novel view synthesis and shape reconstruction [111, 129, 108], implicit coordinate-based representations face a critical limitation: their continuous nature leads to slow extraction of 2D views or 3D surfaces, hindering practical applications requiring real-time performance. To address this bottleneck, a recent breakthrough, 3D Gaussian Splatting (3DGS) [49], revisits explicit representation by modeling 3D shapes as a set of 3D Gaussian kernels with learnable properties (*e.g.*, position, scale, and opacity). This particle-based paradigm achieves exceptional quality in novel view synthesis [27, 13, 5] while enabling real-time rendering speeds. Their variants such as 2D Gaussian Splatting (2DGS) [40] and GeoSurf [117] further achieve accurate surface reconstruction without sacrificing appearance modeling and computational efficiency.

In our works, we adopt distinct 3D representation strategies tailored to the needs of each task: for NVFi, we leverage radiance fields (NeRF) as the core representation, leveraging their strength in continuous modeling of fine-grained shape and appearance detail. In contrast, FreeGave and TRACE adopt 3DGS as their core 3D shape and appearance representation. Its explicit particle-based formulation—with learnable Gaussian kernels—proves particularly amenable to particle-centric frameworks, such as learning physics codes and velocities for rigid particles in FreeGave and integrating with particle-based physics learning pipelines in TRACE.

2.2 Dynamic 3D Reconstruction

Recent advances in dynamic 3D reconstruction have closely followed the evolution of static 3D representation techniques, with most methods built upon established frameworks such as SDF [81], NeRF [76], and more recently 3DGS [49]. To capture temporal dynamics from time-varying visual inputs, these approaches commonly incorporate the time dimension t into their underlying static 3D representations, aiming to learn motion or deformation fields for either rigid or deformable objects and scenes [110, 59, 7, 32, 109, 69, 10, 30, 8, 26, 56, 82, 140, 24, 83, 127, 116, 65, 141].

Among these, NeRF-based methods have shown remarkable fidelity in representing dynamic scenes [10, 30], with techniques further categorized into deformation-based [82, 110, 7, 86, 8, 26, 140], flow-based [32, 59, 24, 116, 69], and direct space-time based approaches [127, 2, 56, 83]. Additionally, domain-specific NeRF variants have been developed to model dynamic objects like human bodies [84, 93, 85, 130, 123, 150, 87], and faces [7, 31, 120, 122]. More recently, methods leveraging 3DGS as the backbone have demonstrated impressive performance in novel view rendering [63, 70, 58, 88, 125, 97, 75, 3, 136, 48, 52, 137], benefiting from its efficiency in real-time rendering.

Despite their success in synthesizing novel views within the observed time period of training data, a critical limitation persists across these approaches: they primarily fit the visual observations or pixel correlations in the training sequence without encoding underlying physics priors. As a result, they lack the ability to predict physically meaningful future frames beyond the observed time window. Our works address this limitation through distinct strategies tailored to their respective frameworks: NVFi, built on NeRF, focuses on modeling the underlying physical principles of 3D scenes, enabling straightforward extrapolation of future dynamics. FreeGave, which adopts 3DGS, explicitly learns a physics code for each Gaussian kernel via a dedicated Gaussian velocity module and divergence-free parameterization, encoding physical priors into the representation. TRACE, also leveraging 3DGS, separates the learning

of translation and rotation dynamics systems for 3D rigid particles, allowing inference of physically consistent future frames that existing methods cannot achieve.

2.3 3D Physics Learning

Physics learning for 3D objects and scenes has emerged as a critical area at the intersection of deep learning and computational physics, aiming to model and predict physical behaviors from data or first principles. A prominent paradigm in this field is physics-informed neural networks (PINNs), introduced in seminal work by [91], which revolutionizes traditional numerical methods by representing tensor fields and partial differential equation (PDE) solutions via neural networks. PINNs convert PDE constraints into loss functions, framing the problem of solving PDEs as optimizing network weights to minimize these physics-based losses [91, 100, 51].

Building on this foundation, a wealth of subsequent works have expanded PINNs’ applications across diverse domains, including acoustics [101, 105], fluid dynamics [112, 33, 22], and 3D scene representations [14, 21, 57, 89], leveraging their mesh-free formulation for flexibility [98, 92, 37, 77, 119, 4, 121, 41, 11, 90, 148]. Comprehensive surveys [46, 36] further document their theoretical advancements and practical impacts. Despite their success, PINNs face notable limitations: they often require large volumes of data to regularize the learning process, and their soft PDE-based constraints are frequently insufficient to ensure accuracy—whether in modeling complex physics in boundary regions or achieving satisfactory results in general physical modeling tasks.

Beyond PINNs, another active line of research integrates explicit physics systems directly into the learning process to model dynamic behaviors. These methods embed domain-specific physical structures such as springs, graphs, or fluid dynamics solvers into neural networks, focusing on targets like elastic objects [151, 146, 64], fluids [42, 61], and other deformable systems [131, 42, 73, 151, 146, 64, 89, 22, 29, 124].

While effective for their specific domains, these approaches are often tailored to narrow applications (*e.g.*, spring systems for elastic objects or fluid-specific solvers) and lack generality across diverse deformation and transformation dynamics.

Our works address these limitations through targeted innovations: In NVFi, we identify that standard PINN constraints alone are insufficient for learning accurate 3D velocity fields. To mitigate this, we introduce a keyframe-based velocity propagation module, enhancing the fidelity of velocity field modeling. FreeGave avoids reliance on inefficient PINN losses entirely, instead directly encoding generic divergence-free constraints into the learning process to incorporate physical priors without PDE-based soft constraints. TRACE advances explicit physics integration by learning a general translation-rotation dynamics system, which is applicable to a broad range of deformation and transformation dynamics—unlike existing methods limited to specific systems (*e.g.*, springs for elasticity or fluid solvers). This generality enables robust modeling of diverse physical behaviors.

2.4 Video Prediction

Some studies focus on directly predicting future video frames [28, 35, 67]. Leveraging the expressive power of diffusion models and Transformer-based frameworks, a series of methods has emerged to tackle video prediction with remarkable success. These approaches [39, 38, 113, 78, 142, 126, 16, 66] excel at modeling long-range temporal dependencies and fine-grained visual details, often generating future frames with high fidelity in texture, color, and short-term motion continuity. However, a critical limitation persists in these state-of-the-art approaches: their tendency to produce physically implausible motions, despite achieving strong visual quality. As noted in [68, 45], this shortcoming stems from a key structural flaw: these methods primarily learn statistical correlations from pixel-level patterns in training data, without explicitly incorporating the underlying physics priors that govern real-world dynamics.

Chapter 3

General Physics-Constrained Velocity Fields

3.1 Introduction

The 3D world around us is constantly changing over time, where objects fall, vehicles move, and clocks tick. Humans can effortlessly learn the geometry and physical properties of such dynamic 3D scenes, and further predict their future motions following the learned physics rules, just by watching the things for a few seconds. Giving machines such ability to automatically infer the geometry and physics of complex dynamic 3D scenes is essential for many cutting-edge applications in augmented reality, games, and the movie industry. Recent advances in the emerging area of neural radiance field [76] and its succeeding variants [86, 7, 10, 30, 144, 80] have shown excellent results in modeling dynamic 3D scenes such as deformable things [7, 110] and articulated objects [79, 104]. While showing superior results in interpolating novel views within the observed time period, almost all these methods tend to fit training image sequences, without explicitly learning physical properties such as object velocities, thus being unable to extrapolate and predict future motions of 3D scenes.

More recently, a few studies [89, 21, 14, 57, 4] start to integrate physics priors into implicit neural representations to model dynamic 3D scenes such as floating smoke or simple moving objects. By introducing governing PDEs, *a.k.a.*, PINN [91], they show promising reconstruction of 3D scene geometry, appearance, velocity and/or viscosity fields. However, the learned physical properties are either tightly coupled with the target objects [21] in the scene or require additional foreground segmentation masks [57]. This means that the estimated physics knowledge from dynamic video is not clearly disentangled, thereby not transferable from one scene to another.

In this regard, we ask an ambitious question: *Can we learn disentangled physical properties alongside recovering geometry and appearance of dynamic 3D scenes purely from multi-view video frames?* Among various physical properties in the scene, we choose to focus on velocity as it primarily governs scene movement dynamics. Such disentangled velocity fields, once successfully learned, are expected to unlock multiple desirable applications: 1) Future frame extrapolation and prediction beyond the observed time period in training. For example, after watching a football flying in the penalty area, we can predict what will happen next. 2) Dynamics transformation from one scene to another. For instance, after watching a bird flipping wings, we can imagine the same physical behavior on the body of an airplane. 3) Semantic decomposition of 3D scenes. Intuitively, once the velocity field of an entire dynamic 3D scene is learned, all individual objects or parts undergoing different moving patterns can be easily segmented, without needing any extra human annotations for training.

However, accurately learning the physical velocity of a whole 3D scene space is particularly challenging, primarily due to the lack of ground truth 3D velocity annotations, the unknown object types or materials in the scene, and the sparse yet long-time visual trajectory in training. In addition, when separately learning the velocity field, it is usually an ill-posed problem even with the commonly used PINN technique [21].

To tackle these challenges, we introduce a new general framework to simultaneously model the geometry, appearance, and disentangled velocity of a dynamic 3D scene only



Figure 3.1: The three major components of our framework: Keyframe Dynamic Radiance Field, Interframe Velocity Field, and the Joint Keyframe and Interframe Optimization module.

from multi-view video frames. In particular, as shown in Figure 3.1, our framework consists of three major components: 1) a **keyframe dynamic radiance field** to learn time-dependent volume density and appearance for every 3D point in space; 2) an **interframe velocity field** to learn time-dependent 3D velocity for every point as well; and 3) a **joint keyframe and interframe optimization** method together with physics informed constraints to train both networks. For the first component, it is flexible to adopt any of the existing time-dependent NeRF architectures such as HexPlane [10] or K-Planes [30]. For the second component, the neural network actually can be as simple as MLPs.

The core of our framework is the third component, where we apply three losses to jointly optimize both networks: 1) the keyframe photometric loss, 2) the interframe photometric loss, and 3) the governing PDE losses. With these losses, our framework can precisely learn disentangled velocity fields, without needing additional regularization terms on volume density or information on object masks, types, or materials. Overall, our framework models general dynamic 3D scenes by learning **neural velocity fields** with physical priors. Our method is named **NVFi** and our contributions are:

- We introduce a general framework to model dynamic 3D scenes as physics-informed radiance fields from multi-view videos, without requiring information on object types, materials, or masks.

- We design a neural velocity field together with a joint keyframe and interframe optimization method to effectively train the networks.
- We demonstrate three applications for the learned velocity fields on two newly collected dynamic 3D datasets and a challenging real-world dataset, showing superior results in future frame extrapolation, semantic decomposition, and velocity transferring across 3D scenes.

3.2 NVFi

3.2.1 Overview

As shown in Figure 3.2, our framework consists of two neural networks together with their optimization methods. Given a set of images of a dynamic scene with known camera poses and intrinsics, the **keyframe dynamic radiance field** f_{Θ} simply takes a 3D point $\mathbf{p} = (x, y, z)$, viewing angle (θ, ϕ) , and timestamp t as input, directly regressing the volume density σ and color $\mathbf{c} = (r, g, b)$. For the network architecture, we simply adopt the recent HexPlane [10] which shows excellent performance in efficiently modeling dynamic video frames, though other NeRF variants can also be used. Formally, our keyframe dynamic radiance field is defined below and implementation details are in Appendix.

$$(\sigma, \mathbf{c}) = f_{\Theta}(x, y, z, \theta, \phi, t), \quad \Theta \text{ are trainable parameters.} \quad (3.1)$$

For the **interframe velocity field** g_{Φ} , it takes the 4D vector (x, y, z, t) as input, and predicts the 3D velocity $\mathbf{v} = (v_x, v_y, v_z)$ for point $\mathbf{p}$ at time t . For simplicity, the network g_{Φ} is parameterized by simple MLPs, though more advanced architectures can be applied as well. Formally, the velocity field is defined below.

$$\mathbf{v} = g_{\Phi}(\mathbf{p}, t) = g_{\Phi}(x, y, z, t), \quad \Phi \text{ are trainable parameters of MLPs.} \quad (3.2)$$

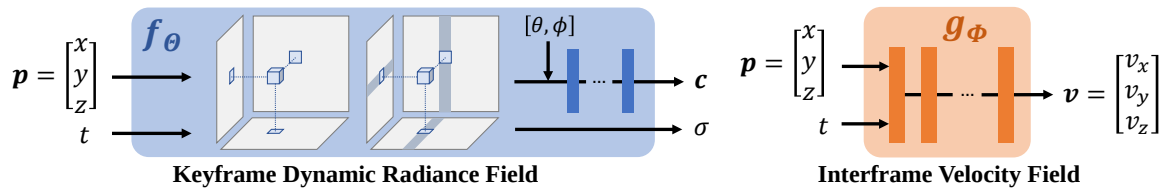


Figure 3.2: The left block illustrates the network architecture of our keyframe dynamic radiance field based on HexPlane [10], and the right block shows the architecture of our interframe velocity field.

With these two networks and training images sampled from a dynamic 3D scene, the key challenge is to effectively optimize these networks, such that the final learned velocity field is precise and disentangled, supporting future frame extrapolation, motion transfer, and semantic decomposition.

3.2.2 Module Architectures

(1) Keyframe Dynamic Radiance Field: We adapt HexPlane[10] to parameterize volumetric keyframes. It is a direct extension of static TensorRF[12] to a dynamic radiance field. Instead of using concatenation to combine the features from different planes as TensorRF and HexPlane, we use Hadamard Production as used in K-Plane[30], which shows stronger ability in representing complex geometries. The definition of our Keyframe Dynamic Radiance Field is as follows.

If we denote τ_k as the time step corresponding to the k^{th} keyframe, we can write:

$$\begin{aligned} \mathbf{A}(\mathbf{x}_i, \tau_k) &= \mathbf{B}(\mathbf{f}_1(x_i, y_i) \odot \mathbf{g}_1(z_i, \tau_k) \odot \mathbf{f}_2(x_i, z_i) \odot \mathbf{g}_2(y_i, \tau_k) \odot \mathbf{f}_3(y_i, z_i) \odot \mathbf{g}_3(x_i, \tau_k)), \\ \sigma(\mathbf{x}_i, \tau_k) &= \mathbf{1}^\top (\mathbf{h}_1(x_i, y_i) \odot \mathbf{k}_1(z_i, \tau_k) \odot \mathbf{h}_2(x_i, z_i) \odot \mathbf{k}_2(y_i, \tau_k) \odot \mathbf{h}_3(y_i, z_i) \odot \mathbf{k}_3(x_i, \tau_k)), \end{aligned}$$

where $\sigma(\mathbf{x}_i, \tau_k)$ is the predicted sigma for point $\mathbf{x}_i$, $A(\mathbf{x}_i, \tau_k)$ is the predicted appearance feature. And $\mathbf{f}_j$, $\mathbf{h}_j$, $\mathbf{g}_j$ and $\mathbf{k}_j$ are vector-valued grid functions with output dimension M_σ or M_{app} respectively. The final density value σ is the summation of the M_σ

dimension density feature, and the final appearance feature is a weighted sum of the M_{app} dimension output feature, where $\mathbf{B}$ is a simple linear layer. Note unlike HexPlane[10] and K-Planes[30], we only do grid interpolation to find the feature values for spatial coordinates, while we pick the time dimension according to the corresponding keyframe index. The final RGB color $\mathbf{c}$ is regressed from a 2-layer MLP with appearance feature and view direction as inputs. With points' density σ and colors $\mathbf{c}$, images are rendered via volumetric rendering equations:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad (3.3)$$

where δ_i represents the distance between the i_{th} and $(i+1)_{\text{th}}$ sample point, and N denotes the number of sampled points. Equation (3.3) connects real 3D points with image pixels by accumulating colors $\mathbf{c}_i$ and densities σ_i of sample points along the ray.

In our implementation, for all the experiments, we set M_σ as 24 and M_{app} as 48, while for different experiments, we use different keyframe numbers. For object extrapolation task, we set keyframe number as 16, due to the more complex dynamics of the deformable objects. For indoor extrapolation and segmentation tasks, we set keyframe number as 4. The color decoder MLP has 2 hidden layers, each with 128 nodes.

(2) Interframe Velocity Field: The velocity field is defined as follows:

$$\mathbf{v} = g_\Phi(\mathbf{p}, t) = g_\Phi(x, y, z, t) = w_\Phi(x, y, z, t) \mathbf{M}_\mathbf{v}(x, y, z), \quad (3.4)$$

where Φ are trainable parameters of MLPs.

Here we decompose the velocity field as a weight MLP w_Φ along with a velocity basis field. In our experiment, we find this can make the velocity convergence faster. The

velocity basis field is defined as:

$$\mathbf{M}_v(x, y, z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -z & 0 & x \\ -y & x & 0 \\ 0 & -z & y \end{bmatrix}. \quad (3.5)$$

The weight MLP is implemented as 4 hidden layers with 128 nodes, with a positional encoder of dim 3 as in [76]. The output is a weight vector $\mathbf{w} \in \mathbb{R}^6$, which could be regarded as the linear velocity in x, y, z directions and angular velocity rounding x, y, z axis respectively.

3.2.3 Optimization of Keyframe Dynamic Radiance Fields

Given dynamic video frames of T timestamps $\{1 \dots t \dots T\}$, there are two potential strategies to optimize dynamic neural radiance fields.

- **Strategy 1 - Dense Frame Optimization:** This strategy uses all available video frames of a specific dynamic 3D scene in the training set to optimize a dynamic radiance field. This means that for the network f_Θ , the time dimension t is densely sampled during optimization. Formally:

$$f_\Theta(x, y, z, \theta, \phi, t) : \quad t \in \{1 \dots t \dots T\} \quad (3.6)$$

However, it has two limitations: 1) It is inefficient to learn accurate 3D geometry and appearance because this strategy is somewhat equivalent to modeling a dense number of static radiance fields for all timestamps. 2) It is hard to obtain a disentangled physical velocity field for the entire 3D scene, as the change of physical geometries is tightly encoded within the dynamic radiance field.

- **Strategy 2 - Canonical Frame Optimization:** This strategy optimizes a canonical representation of 3D geometry and appearance, usually joined with another deformation or transportation network to warp the future $(T - 1)$ timestamps back to the first timestamp. It can be seen as:

$$f_{\Theta}(x, y, z, \theta, \phi, t) : \quad t \in \{1\} \quad (3.7)$$

However, such a strategy has a strong assumption that the corresponding point appearances across different timestamps keep unchanged. Therefore, for dramatically changing 3D scenes, the optimized geometry, appearance, and possible jointly learned physics properties tend to be inferior.

Our Strategy - Keyframe Optimization: In this regard, we propose to adopt a keyframe based strategy to learn the dynamic radiance field f_{Θ} . In particular, we uniformly sample K timestamps out of the total T stamps to optimize f_{Θ} . Formally:

$$f_{\Theta}(x, y, z, \theta, \phi, t_k) : \quad t_k \in \left\{ \lceil T/K \rceil, 2\lceil T/K \rceil, 3\lceil T/K \rceil, \dots, T \right\} \quad (3.8)$$

Color images are rendered from the above keyframe dynamic radiance fields by sampling points along rays. For each pixel, *i.e.* ray $\mathbf{r}$, in a keyframe at time t_k , the appearance $\mathbf{C}(\mathbf{r}, t_k)$ is obtained by volume rendering of NeRF [76] as shown in Equation 3.3. Then the network f_{Θ} can be optimized by the following keyframe photometric loss:

$$\ell_{keyframe} = ||\mathbf{C}(\mathbf{r}, t_k) - \bar{\mathbf{C}}(\mathbf{r}, t_k)||, \quad (3.9)$$

where $\bar{\mathbf{C}}(\mathbf{r}, t_k)$ is the ground truth color observation.

This keyframe optimization strategy, albeit simple, has two unique advantages: 1) It allows to sufficiently and accurately fit the sparsely sampled dynamic 3D scene geometry and appearance given the same network capacity. 2) It allows the remaining interframes belonging to the $(T - K)$ time stamps to be used for optimizing a disentangled velocity field, as discussed in Section 3.2.4.

3.2.4 Optimization of Interframe Velocity Fields

As to our interframe velocity field g_Φ , it is impossible to directly supervise it using ground truth labels as they cannot be collected in practice. However, there are some physics rules to regularize the velocity field. The objects are transported by the velocity field, and the velocity field is transported by itself according to some unobservable hidden forces. In order to keep the mass and appearance of the objects, the velocity field needs to be divergence-free and obeys the basic law of momentum conservation.

1) Mass Conservation: We enforce the appearance and volume density features, denoted as f_i both admit the conservation laws characterized by the divergence-free implicit velocity field $\mathbf{v}(\mathbf{x}, t)$:

$$\frac{\partial f_i}{\partial t} + \nabla \cdot (\mathbf{v} f_i) = \frac{\partial f_i}{\partial t} + \mathbf{v} \cdot \nabla f_i + f_i \nabla \cdot \mathbf{v} = \frac{Df_i}{Dt} + f_i \nabla \cdot \mathbf{v} = 0,$$

where $\frac{D}{Dt}$ is the material derivative. Intuitively, we assume the objects moved by the velocity field have a property of local rigidity, and the implicit velocity field should be incompressible (or in other words free of sources or sinks of mass). Then we can get the divergence-free constraint $\nabla \cdot \mathbf{v} = 0$.

2) Momentum Conservation: The momentum of a velocity field is defined to be $\rho \mathbf{v}$, per unit volume. Newton's second law of motion states that momentum is conserved by a mechanical system of masses if no forces act on the system. If $\mathbf{F}(\mathbf{x}, t)$ is the force acting on the velocity, per unit volume, then we immediately have

$$\rho \frac{D\mathbf{v}_i}{Dt} = \mathbf{F},$$

where we assume the implicit velocity field has uniform mass density ρ . As a result, the momentum conservation constraint could be derived as:

$$\frac{D\mathbf{v}_i}{Dt} = \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \frac{\mathbf{F}}{\rho} = \mathbf{a}. \quad (3.10)$$

Note that, more complex physics dynamics beyond the daily 3D scenarios are out of the scope of this chapter.

In this regard, our velocity field, *i.e.*, $\mathbf{v} = g_\Phi(\mathbf{p}, t)$, needs to firstly satisfy the following two constraints.

$$\nabla_{\mathbf{p}} \cdot \mathbf{v} = 0, \quad \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{p}} \mathbf{v} = \mathbf{a} \quad (3.11)$$

We simply turn these PDEs into the following two PINN losses [91] to optimize the velocity field. Here we use $\mathbf{v}(\mathbf{p}, t)$ as the network to avoid abuse of notation.

$$\begin{aligned} \ell_{divergence_free} &= \frac{1}{NM} \sum_{n=1}^N \sum_{m=1}^M \|\nabla_{\mathbf{p}_n} \cdot \mathbf{v}(\mathbf{p}_n, t_m)\| \\ \ell_{momentum} &= \frac{1}{NM} \sum_{n=1}^N \sum_{m=1}^M \left\| \frac{\partial \mathbf{v}(\mathbf{p}_n, t_m)}{\partial t_m} + \mathbf{v}(\mathbf{p}_n, t_m) \cdot \nabla_{\mathbf{p}_n} \mathbf{v}(\mathbf{p}_n, t_m) - \mathbf{a} \right\| \end{aligned} \quad (3.12)$$

where $\mathbf{p}_n$ is uniformly sampled in the whole 3D scene volume, and t_m is uniformly sampled from 0 to the interested maximum extrapolation time t_{max} , and $\mathbf{a}$ is the general acceleration term learned by another MLP-based network: $(x, y, z, t) \rightarrow \mathbf{a}$. Nevertheless, with such PINN losses, it is insufficient to optimize the velocity field itself, since there are infinitely many solutions. To tackle this, we introduce an additional interframe optimization strategy.

Interframe Optimization Strategy: Naturally, the 3D scene geometry and appearance encoded in the keyframe dynamic radiance field, once appropriately transported by the velocity field, should be able to render 2D images to match with the ground truth observations in interframes belonging to the remaining $(T - K)$ timestamps. To enforce such a constraint, the key challenge is to determine the color and density values for all 3D points at each interframe timestamp, such that the volume rendering equation can be applied to estimate RGB for each pixel at the interframe timestamp, after which the photometric loss can be adopted. To tackle this, we propose the following Algorithm 1.

Having the color and density values of all 3D points along ray $\mathbf{r}_i$ in the interframe of timestamp t_i , the appearance $\mathbf{C}(\mathbf{r}_i, t_i)$ is obtained by volume rendering of NeRF [76]. Then both networks f_Θ and g_Φ can be optimized by the following interframe

photometric loss.

$$\ell_{interframe} = ||\mathbf{C}(\mathbf{r}_i, t_i) - \bar{\mathbf{C}}(\mathbf{r}_i, t_i)||, \quad (3.13)$$

where $\bar{\mathbf{C}}(\mathbf{r}_i, t_i)$ is the ground truth color observation.

3.2.5 Joint Keyframe and Interframe Optimization

Technically, the keyframe dynamic radiance field f_{Θ} can be firstly trained by $\ell_{keyframe}$ only, and then the interframe velocity field g_{Φ} by $\ell_{divergence_free} + \ell_{momentum} + \ell_{interframe}$.

Nevertheless, we empirically find that simultaneously propagating errors of interframes back to the radiance field f_{Θ} helps achieve better performance overall. Therefore, we adopt the following joint keyframe and interframe strategy to optimize both networks.

$$f_{\Theta} \leftarrow (\ell_{keyframe} + \ell_{interframe}) \quad (3.14)$$

$$g_{\Phi} \leftarrow (\ell_{divergence_free} + \ell_{momentum} + \ell_{interframe}) \quad (3.15)$$

Algorithm 1 At a specific interframe timestamp t_i , given a light ray $\mathbf{r}_i$ with viewing angle (θ, ϕ) and S sample points $\{\mathbf{p}_1 \cdots \mathbf{p}_s \cdots \mathbf{p}_S\}$ along the ray, the objective of this algorithm is to determine the color and density values for the S points along $\mathbf{r}_i$, denoted as: $\{(\mathbf{c}_1, \sigma_1) \cdots (\mathbf{c}_s, \sigma_s) \cdots (\mathbf{c}_S, \sigma_S)\}$. In the meantime, we also have the keyframe dynamic radiance field f_Θ and velocity field g_Φ .

Input:

- The ray direction (θ, ϕ) , the interframe timestamp t_i , the S sample points on the ray $\{\mathbf{p}_1 \cdots \mathbf{p}_s \cdots \mathbf{p}_S\}$;
- The K keyframe timestamps $\{t_1 \cdots t_k \cdots t_K\}$;
- The initialized and ongoing training networks: f_Θ and g_Φ ;

Output:

- The color and density values for S sample points along the ray: $\{(\mathbf{c}_1, \sigma_1) \cdots (\mathbf{c}_s, \sigma_s) \cdots (\mathbf{c}_S, \sigma_S)\}$;

Preliminary step:

- Find the nearest keyframe timestamp $\hat{t}_k$ for the interframe timestamp t_i :

$$\hat{t}_k = \arg \min_{t_k} |t_k - t_i|$$

for $\mathbf{p}_s$ in $\{\mathbf{p}_1 \cdots \mathbf{p}_s \cdots \mathbf{p}_S\}$ do

- Transport $\mathbf{p}_s$ to its corresponding point $\mathbf{p}'_s$ at its nearest keyframe timestamp $\hat{t}_k$, according to its velocity field. The position of $\mathbf{p}'_s$ can be obtained by:

$$\mathbf{p}'_s = \mathbf{p}_s + \int_{t_i}^{\hat{t}_k} g_\Phi(\mathbf{p}_s(t), t) dt, \quad \text{Runge-Kutta 2 solver [15] is applied in our implementation.}$$

- Retrieve the volume density σ'_s and view-agnostic color feature vector $\mathbf{e}'_s$ for point $\mathbf{p}'_s$:

$$(\sigma'_s, \mathbf{e}'_s) \leftarrow f_\Theta(\mathbf{p}'_s, \hat{t}_k), \quad \text{Note: HexPlane [10] can output a view-agnostic color feature } \mathbf{e}'_s.$$

- Assign the retrieved features of $\mathbf{p}'_s$ to the original point $\mathbf{p}_s$: $(\sigma_s, \mathbf{e}_s) \leftarrow (\sigma'_s, \mathbf{e}'_s)$.
- Obtain the color $\mathbf{c}_s$ for point $\mathbf{p}_s$:

$$\mathbf{c}_s \leftarrow \tilde{f}_\Theta(\mathbf{e}_s, \theta, \phi), \quad \text{Note: } \tilde{f}_\Theta \text{ is a subnetwork of HexPlane [10] as detailed in Sec 3.2.2.}$$

- Output $(\mathbf{c}_s, \sigma_s)$ for point $\mathbf{p}_s$.

After the above *for loop*, we obtain all color and density values for S sample points.

3.3 Experiments

Datasets: Our framework primarily focuses on learning meaningful physical velocity fields for dynamic 3D scenes, instead of simply fitting video frames. Although there are a number of dynamic 3D scene datasets in the literature, they are mainly collected for the popular task of novel view synthesis within the training time period, *i.e.*, interpolation in time dimension. Besides, the underlying motions of these scenes tend to be chaotic, and estimating their future motions or transferring their motions are hardly meaningful or entertaining in practice. In this regard, we introduce two new synthetic datasets: 1) Dynamic Object dataset, and 2) Dynamic Indoor Scene dataset.

1) Dynamic Object Dataset: This dataset consists of 6 distinct 3D objects, each of which displays a unique motion pattern, including either rigid or deformable movements in 3D space. These 3D objects and their realistic motions are all designed by unknown external practitioners from SketchFab, and we purchased their Licenses and will make them available for free use in the community.

For each 3D object, we collect RGB images at 15 different viewing angles over 1 second, where each viewing angle has 60 frames captured. We reserve the first 45 frames at randomly picked 12 viewing angles as the training split, *i.e.*, 540 frames, while leaving the 45 frames at the remaining 3 viewing angles for testing interpolation ability, *i.e.*, 135 frames for novel view synthesis within the training time period, and keeping the last 15 frames at all 15 viewing angles for evaluating future frame extrapolation, *i.e.*, 225 frames. Details of the 6 dynamic objects are:

- **Falling Ball:** A basketball is falling freely through a hoop. The basketball is accelerated by gravity, which should be learned by the model.
- **Telescope:** A telescope is given, whose upper part is rotating while the lower part remains static.
- **Fan:** An antique fan is given. The outer part of the fan is static, while the inner

fan is rotating. The embedded motion makes this scene difficult.

- **Bat:** A bat is flapping its wings. Since the wing is extremely thin, and the motion is in a great extent, it's hard to reconstruct the motion.
- **Shark:** A shark is shaking its tail left and right.
- **Whale:** A whale is flapping its tail up and down.

2) *Dynamic Indoor Scene Dataset:* We also collect another synthetic dynamic 3D dataset, which includes 4 indoor scenes with multiple complex 3D objects undergoing different rigid body motions. There are about 4 objects such as tables or chairs in each 3D scene. Basically, such an indoor dataset aims to simulate potential scenarios for robotics or VR applications to understand dynamic 3D scenes.

Since the indoor scene is significantly more challenging, for each 3D scene, we collect RGB images at 30 viewing angles over 1 second, where each viewing angle has 60 frames captured. Similarly, we reserve the first 45 frames at randomly picked 25 viewing angles as the training split, *i.e.*, 1125 frames, while leaving the 45 frames at the remaining 5 viewing angles for testing interpolation ability, *i.e.*, 225 frames, and keeping the last 15 frames at all 30 viewing angles for evaluating future frame extrapolation, *i.e.*, 450 frames. The ground truth object segmentation masks are also collected for evaluating the semantic decomposition capability in Section 3.3.2.

Details of the 4 scenes are:

- **Gnome House:** 3 objects are moving outwards from the center of a house: a gnome, a sofa, and a treasure chest.
- **Chessboard:** 5 objects are moving towards each other on a chessboard: a horse statue, a marble bust, and a china vase from one side, a wooden elephant and a brass vase from the other side.

- **Factory:** 5 objects are moving towards nearly the same direction, among which 2 objects are rotating at the same time: three barrels with different appearance, a cardboard box, and a compost bag.
- **Dining Table:** 4 objects are falling down onto a dining table from the surrounding air: an apple, a cake, a croissant, and a lime.

While existing dynamic 3D scene modeling techniques and the commonly used datasets in literature are mainly designed for novel view rendering/interpolation within the training time period, rather than for extrapolation beyond the training time period, we evaluate our method on two selected scenes from a real-world dataset: NVIDIA Dynamic Scene[139]. It captures real-world dynamic scenes by a static camera rig with 12 cameras. For each scene, we clip 60 frames with reasonable and predictable motions. We reserve the first 46 frames at randomly picked 11 cameras as the training split, *i.e.*, 506 frames, while leaving the 46 frames at the remaining 1 camera for testing interpolation ability, *i.e.*, 46 frames for novel view synthesis within the training time period, and keeping the last 14 frames at all 12 cameras for evaluating future frame extrapolation, *i.e.*, 168 frames.

Baselines: We carefully choose three representative groups of methods as our baselines: 1) dense frame optimization method T-NeRF [86] and NSFF [59], 2) canonical frame optimization methods D-NeRF [86] and TiNeuVox [26], 3) PINN methods T-NeRF_{PINN} and HexPlane_{PINN}. Both methods are adapted by us via integrating a separate velocity field supervised by the same PINN losses as ours.

Metrics: For evaluating both interpolation and future frame extrapolation and motion transfer, the standard metrics **PSNR**, **SSIM**, and **LPIPS** scores are reported across testing views. For evaluating semantic decomposition, the Average Precision (**AP**), Panoptic Quality (**PQ**) and **F1** scores with an IoU threshold of 0.5, together with the mean Intersection over Union (**mIoU**) scores are reported.

Table 3.1: Quantitative results for both novel view interpolation and future frame extrapolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset.

	Dynamic Object Dataset						Dynamic Indoor Scene Dataset					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
T-NeRF[86]	13.163	0.709	0.353	13.818	0.739	0.324	24.944	0.742	0.336	22.242	0.700	0.363
D-NeRF[86]	14.158	0.697	0.352	14.660	0.737	0.312	25.380	0.766	0.300	20.791	0.692	0.349
TiNeuVox[26]	<u>27.988</u>	<u>0.960</u>	0.063	19.612	0.940	0.073	<u>29.982</u>	<u>0.864</u>	<u>0.213</u>	21.029	0.770	<u>0.281</u>
T-NeRF _{PINN}	15.286	0.794	0.293	16.189	0.835	0.230	16.250	0.441	0.638	17.290	0.477	0.618
HexPlane _{PINN}	27.042	0.958	<u>0.057</u>	<u>21.419</u>	<u>0.946</u>	<u>0.067</u>	25.215	0.763	0.389	23.091	0.742	0.401
NSFF[59]	-	-	-	-	-	-	29.365	0.829	0.278	<u>24.163</u>	<u>0.795</u>	0.289
NVFi(Ours)	29.027	0.970	0.039	27.594	0.972	0.036	30.675	0.877	0.211	29.745	0.876	0.204

Table 3.2: Quantitative results on the NVIDIA Dynamic Scene dataset.

	Truck						Skating					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
TiNeuVox[26]	<u>27.230</u>	0.846	0.229	24.887	<u>0.848</u>	0.209	29.377	0.889	0.202	<u>24.224</u>	<u>0.878</u>	<u>0.220</u>
HexPlane _{PINN}	25.494	0.768	0.337	<u>24.991</u>	0.768	0.325	24.447	<u>0.867</u>	<u>0.225</u>	23.955	0.868	0.232
NVFi(Ours)	27.276	<u>0.840</u>	<u>0.235</u>	28.269	0.855	<u>0.220</u>	<u>26.999</u>	0.848	0.227	28.654	0.896	0.208

3.3.1 Evaluation of Future Frame Extrapolation

We first evaluate the extrapolation capability of our method. Particularly, our method and 5 baselines except NSFF are trained from scratch on each of the 6 objects in Dynamic Object dataset, and all methods on each of the 4 scenes in Dynamic Indoor Scene dataset, all in a scene-specific fashion. In total, $(6 \times 6) + (7 \times 4) = 64$ models are trained for comparison. The final keyframe number K is set as 16 in our method for Dynamic Object Dataset and 4 for Dynamic Indoor Scene Dataset. For NVIDIA Dynamic Scene dataset, we evaluate our model and 2 baselines with comparable results on our own datasets, where the final keyframe number K is set as 4 in our

method. In order to adaptively adjust the number of keyframe number according to the reconstruction performance, we use a coarse-to-fine pipeline to reconstruct the keyframe radiance field, *i.e.* we begin with a small keyframe number K and gradually increase it till the reconstruction achieving promising results. And the final keyframe number listed above is chosen the best based on our observation during training.

Analysis: Table 3.1 compares all methods regarding the quality of view synthesis for future frame extrapolation. The view synthesis for interpolation is also included for comparison. It can be seen that: 1) our NVFi achieves significantly better results than all baselines on both dynamic datasets, particularly on the critical task of future frame extrapolation, although the strong baseline TiNeuVox shows excellent performance for interpolation. 2) Naïvely adding physics priors into an existing dynamic NeRF tends to be inferior as shown by T-NeRF_{PINN} and HexPlane_{PINN}. This clearly verifies the effectiveness of our special design of the joint keyframe and interframe optimization strategy. More qualitative results are in Figure 3.7(a). Table 3.2 compares our method and two best baselines. It can be seen that, even if our NVFi only gets comparable performance in interpolation, it still achieves the best performance in extrapolation without any performance drop compared with interpolation thanks to the accurate motion predictions. Qualitative results for NVIDIA Dynamic Scene are in Figure 3.3. We also test the performance of our NVFi on a synthetic smoke scene to demonstrate our ability in modeling fluid-like cases. It can be seen that, although our method cannot predict the unseen diffusion behavior without specific priors, our method can still make right predictions for momentum motion.

3.3.2 Evaluation of 3D Semantic Scene Decomposition

Having the well-trained keyframe dynamic radiance field f_{Θ} and interframe velocity field g_{Φ} for each 3D scene in the Dynamic Indoor Scene dataset in Section 3.3.1, naturally, all individual 3D objects such as chairs and tables undergoing different

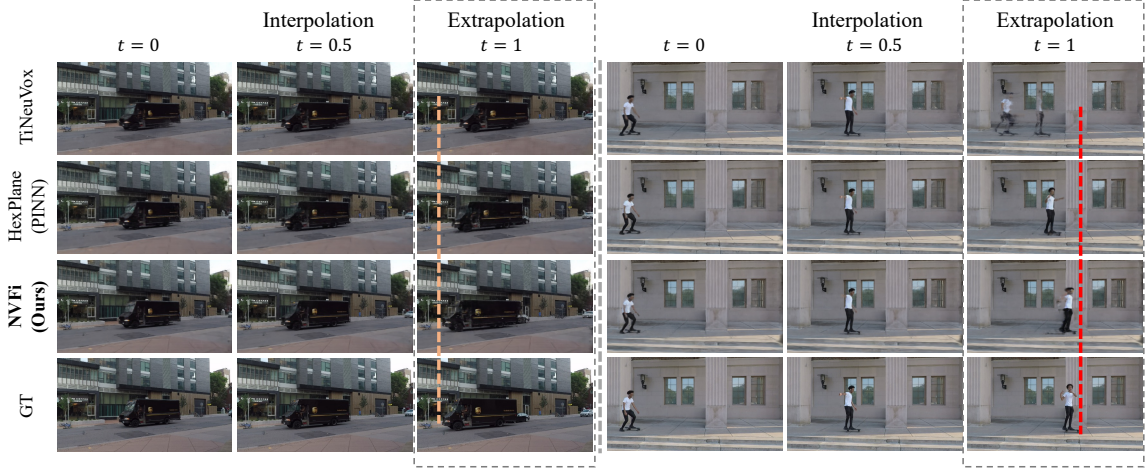


Figure 3.3: Qualitative results on NVIDIA Dynamic Scene dataset.

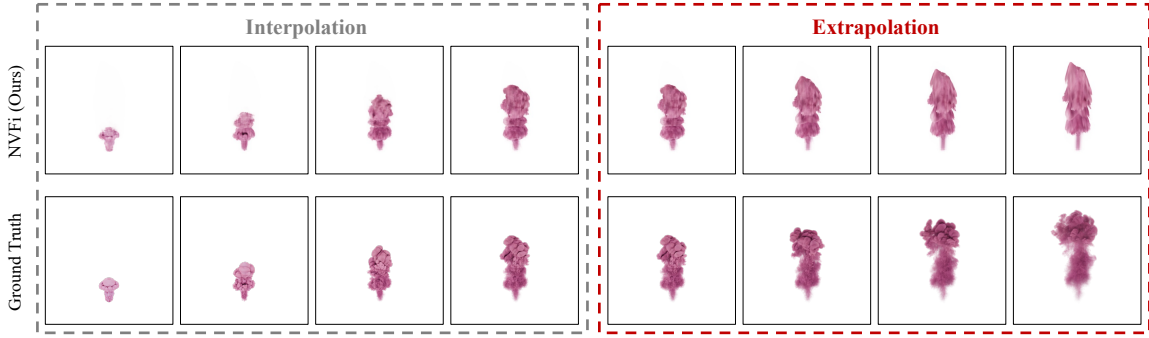


Figure 3.4: Qualitative results on synthetic smoke scene.

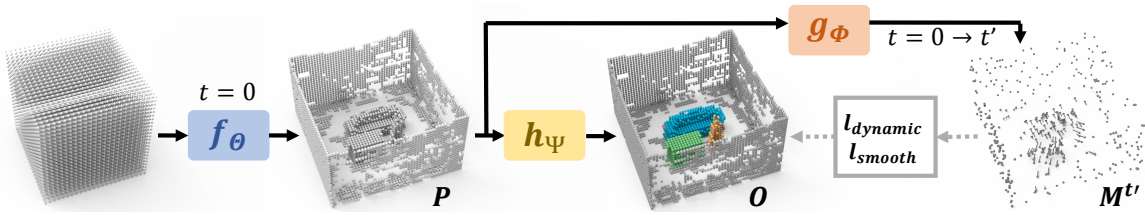


Figure 3.5: The optimization pipeline of our method for scene decomposition. The solid black lines represent forward inference while the dashed gray lines represent backward supervision.

movements are supposed to be automatically discovered, segmented, and tracked without needing any extra object annotations as supervision signals.

In order to achieve this desirable unsupervised object decomposition objective, a naïve strategy is to query the velocity values of dense 3D points in space, followed by a velocity clustering module to group points into objects. However, such a strategy fundamentally fails to recognize object shapes but just identifies similar motions, thereby tracking objects is also infeasible.

A more elegant strategy is to directly learn an object code $\mathbf{o}_p$ (usually one-hot) for each 3D point p within the entire 3D scene volume at the initial timestamp $t = 0$, without retraining any neural layer of the well-trained networks f_Θ and g_Φ but just using them. This would clearly allow all dynamic 3D objects in the scene to be segmented and tracked over time. To this end, we simply introduce a simple 4-layer MLP as the semantic scene decomposition network h_Ψ . It takes a 3D point p as input, directly regressing its object code $\mathbf{o}_p$, *i.e.*, $\mathbf{o}_p = h_\Psi(p)$ which is optimized using the following steps. Figure 3.5 illustrates the optimization pipeline of our method.

- First, given the well-trained keyframe dynamic radiance field f_Θ , we uniformly sample dense 3D points at timestamp $t = 0$, obtaining their density values. Only 3D points with sufficiently large densities are kept as valid points for the subsequent steps, obtaining a non-empty point set $\mathbf{P} \in \mathbb{R}^{N \times 3}$ with N points at $t = 0$.
- Second, the valid 3D points are fed into our new object segmentation network h_Ψ (initialized but yet to be trained), obtaining their corresponding object codes $\mathbf{O} \in (0, 1)^{N \times K}$, where K is a predefined number of objects that the network is expected to predict in maximum.
- Third, the valid points are transported to their correspondences at a random timestamp t' using the well-trained velocity field g_Φ . Thus, we can compute the per-point motion vectors $\mathbf{M}^{t'} \in \mathbb{R}^{N \times 3}$ as $\mathbf{M}^{t'} = \mathbf{P}^{t'} - \mathbf{P}$.
- Lastly, given per-point motions, we employ the dynamic rigid consistency and spatial smoothness losses proposed in OGC [102] to optimize the object segmentation network h_Ψ .

1) Dynamic rigid consistency: For the k^{th} object, We first retrieve its (soft) binary mask $\mathbf{O}^k$, and feed the tuple $\{\mathbf{P}, \mathbf{P}^{t'}, \mathbf{O}^k\}$ into the weighted-Kabsch algorithm to estimate its transformation matrix $\mathbf{T}_k \in \mathbb{R}^{4 \times 4}$ belonging to $SE(3)$ group. Then the dynamic loss is defined as:

$$\ell_{dynamic} = \frac{1}{N} \sum_{\mathbf{p} \in \mathbf{P}} \left\| \left(\sum_{k=1}^K o_{\mathbf{p}}^k \cdot (\mathbf{T}_k \circ \mathbf{p}) \right) - (\mathbf{p} + \mathbf{m}^{t'}) \right\|_2 \quad (3.16)$$

where $o_{\mathbf{p}}^k$ represents the probability of being assigned to the k^{th} object for a specific point $\mathbf{p}$, and $\mathbf{m}^{t'} \in \mathbb{R}^3$ represents the motion vector of $\mathbf{p}$ from timestamp 0 to t' . The operation $\circ$ applies the rigid transformation to the point. This loss serves to discriminate objects with different motions.

2) Spatial smoothness: We first search H nearest neighboring points for each point $\mathbf{p}$ in $\mathbf{P}$. Then the smoothness loss is defined as:

$$\ell_{smooth} = \frac{1}{N} \sum_{\mathbf{p} \in \mathbf{P}} \left(\frac{1}{H} \sum_{h=1}^H \|\mathbf{o}_{\mathbf{p}} - \mathbf{o}_{\mathbf{p}_h}\|_1 \right) \quad (3.17)$$

where $\mathbf{o}_{\mathbf{p}} \in (0, 1)^K$ represents the object assignment of center point $\mathbf{p}$, and $\mathbf{o}_{\mathbf{p}_h} \in (0, 1)^K$ represents the object assignment of its h^{th} neighbour point. This loss servers to avoid the over-segmentation issues. We refer readers to [102] for more details.

Once the object segmentation network h_{Ψ} is learned, all dynamic objects at time $t = 0$ are segmented. With the aid of the well-trained velocity field, in any subsequent timestamps, all these identified objects can be naturally tracked. In addition, with the aid of well-trained keyframe dynamic radiance field f_{Θ} , we simply use the accumulated weights computed in volume rendering to combine point object codes along a given light ray, thus rendering accurate object segmentation 2D masks for any camera poses at any given timestamps.

We evaluate the semantic scene decomposition ability on the Dynamic Indoor Scene dataset. In particular, we render all 2D object segmentation masks from our network at the 30 viewing angles over 60 frames for all scenes, *i.e.*, 7200 2D masks, and then

Table 3.3: Quantitative results of scene decomposition on Synthetic Indoor Scene dataset.

	AP $\uparrow$	PQ $\uparrow$	F1 $\uparrow$	Pre $\uparrow$	Rec $\uparrow$	mIoU $\uparrow$
Mask2Former[18]	65.37	<u>73.14</u>	78.29	94.83	68.88	<u>64.42</u>
D-NeRF[86]	57.26	46.15	59.02	56.55	62.94	46.58
NVFi(Ours, $\Delta t = 0.5$)	<u>75.82</u>	63.34	<u>80.59</u>	80.78	<u>81.11</u>	57.05
NVFi(Ours, $\Delta t = 0.06$)	91.21	78.74	93.75	<u>93.76</u>	93.74	67.64

evaluate them against ground truth masks. For a fair comparison, we also train a similar object segmentation network for the baseline D-NeRF [86] at time $t = 0$, using its learned deformation vectors as supervision signals and tracking signals. Note that, the deformation vectors are converted back as motion vectors. In addition, we include an image-based object segmentation method, the powerful Mask2Former [18] pre-trained on COCO [62] dataset, as a fully-supervised baseline.

Analysis: As shown in Table 3.3, we can see that: 1) Our object segmentation performance is superior to D-NeRF [86], essentially because our velocity field learns better scene dynamics than the deformation field, thus enabling the object segmentation network to be better optimized. 2) We also clearly surpass the powerful pre-trained Mask2Former [18] on all metrics. The reasons are two-fold. First, we fundamentally rely on motion patterns rather than appearances to discover objects, thus being able to generalize to unseen object types ("Genome") or scenes ("Factory") better than Mask2Former [18]. Second, our learned object field inherently leverages multi-view consistency, thus allowing the segmentation of partially occluded objects. Figure 3.7(b) shows qualitative results.

3.3.3 Evaluation of Motion Transfer

We further demonstrate the ability of our model to transfer a well-trained velocity field to another separately trained static scene. All objects in the new scene are expected

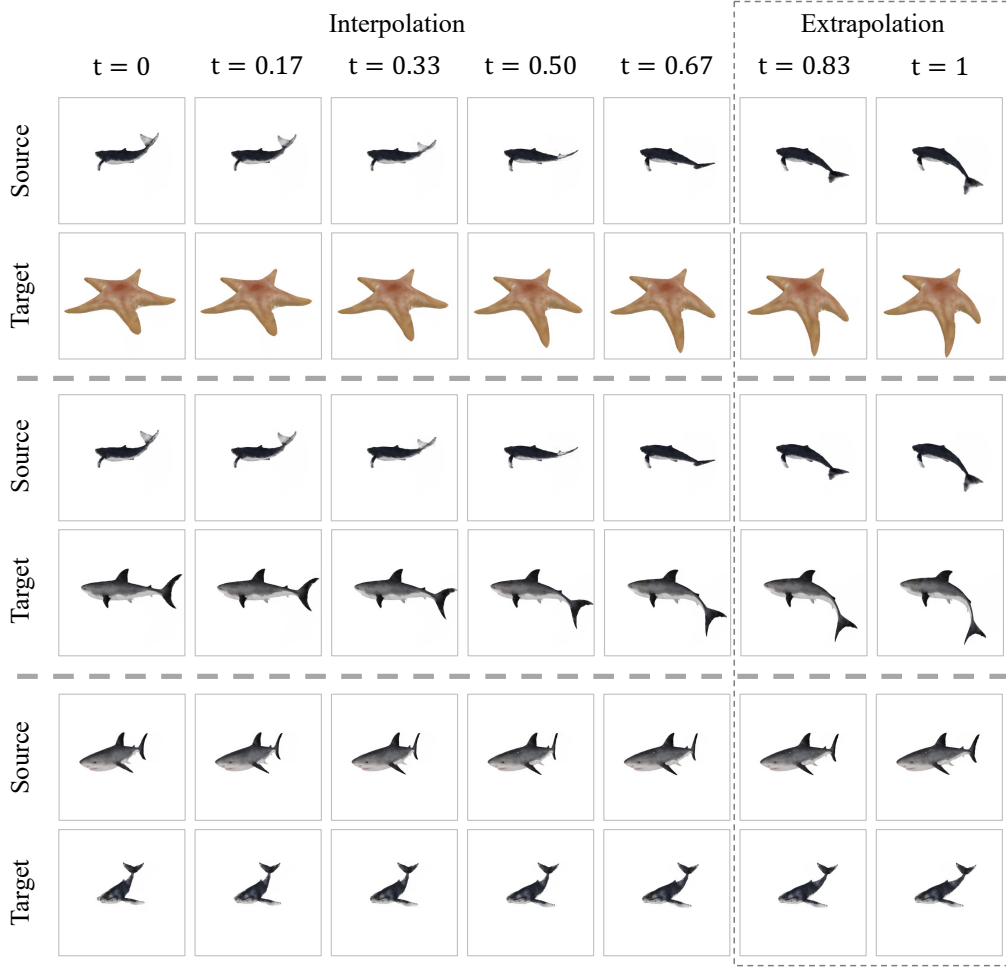


Figure 3.6: Qualitative results for objectwise motion transfer.

to undergo the same dynamics as learned in the velocity field. The more accurate the learned velocity field, the more realistic and entertaining the new 3D scene will be, after applying the learned dynamics.

In order to evaluate the performance, we create a new 3D scene, called *Gnome-new*, being similar to the scene *Gnome* in our Dynamic Indoor Scene dataset. We apply the same dynamics of *Gnome* on *Gnome-new*, recording $30 \times 60 = 1800$ frames as its ground truth observations. To explicitly show the advantage of our learned disentangled velocity field, we separately train a static TensorRF [12] model for *Gnome-new* only using its frames at time $t = 0$. Note that, any other NeRF variants are also applicable here. We then pick up the well-trained velocity field of *Gnome* in Section

3.3.1, after which we directly apply the learned velocity field on the newly trained static TensorRF model, rendering 30×60 frames for a comparison with the ground truth images. Similarly, we apply the deformation field learned by D-NeRF in Section 3.3.1 in the same transferring pipeline, rendering 2D images for comparison.

Analysis: Figure 3.7(c) shows that our method clearly keeps the geometry and appearance of the new static object, thanks to the accurately learned disentangled velocity field, whereas D-NeRF [86] fails to do so. However, we observe that in Table 3.4 our advantage is not

Table 3.4: Quantitative results of motion transfer on Synthetic Indoor Scene dataset.

	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
D-NeRF [86]	16.124	0.327	0.550
NVFi(Ours)	16.178	0.334	0.551

so significant. This is because the static reconstruction of *Gnome-new* lacks supervision signals for ground planes occluded by objects, leading to rendering artifacts in these regions when objects are moved away by motion transfer. Our strong ability of motion transfer is further validated by additional experiments on our Dynamic Object Dataset in Figure 3.6.

3.3.4 Ablation Study

We take the following ablation studies to validate the design of our method:

- (1) **Without Joint Optimization:** We only use $\ell_{keyframe}$ to train the keyframe dynamic radiance field f_{Θ} , followed by the $(\ell_{divergence_free} + \ell_{momentum} + \ell_{interframe})$ to separately train the velocity field g_{Φ} .
- (2) **Removing Physics Constraints:** The PINN losses $(\ell_{divergence_free} + \ell_{momentum})$ are removed to train g_{Φ} .
- (3) **Choice of Keyframe Number K :** We set the keyframe number K as 8 and 32, while we choose $K = 16$ in main experiments.

(4) **Reducing the Number of Cameras:** we reduce the number of cameras used in our Dynamic Object datasets to half of the number, *i.e.*, 6 cameras, and one quarter of the number, *i.e.*, 3 cameras.

Table 3.5: Quantitative results of ablation studies on Dynamic Object dataset.

Joint	Physics	#K	Interpolation			Extrapolation		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✓	✓	16	29.027	0.970	0.039	27.594	<u>0.972</u>	0.036
✗	✓	16	27.525	0.959	0.057	24.792	0.955	0.059
✓	✗	16	<u>29.109</u>	0.970	0.039	25.537	0.968	0.040
✓	✓	8	29.355	0.970	0.041	<u>27.490</u>	0.974	0.036
✓	✓	32	28.922	0.969	0.042	24.902	0.964	0.037

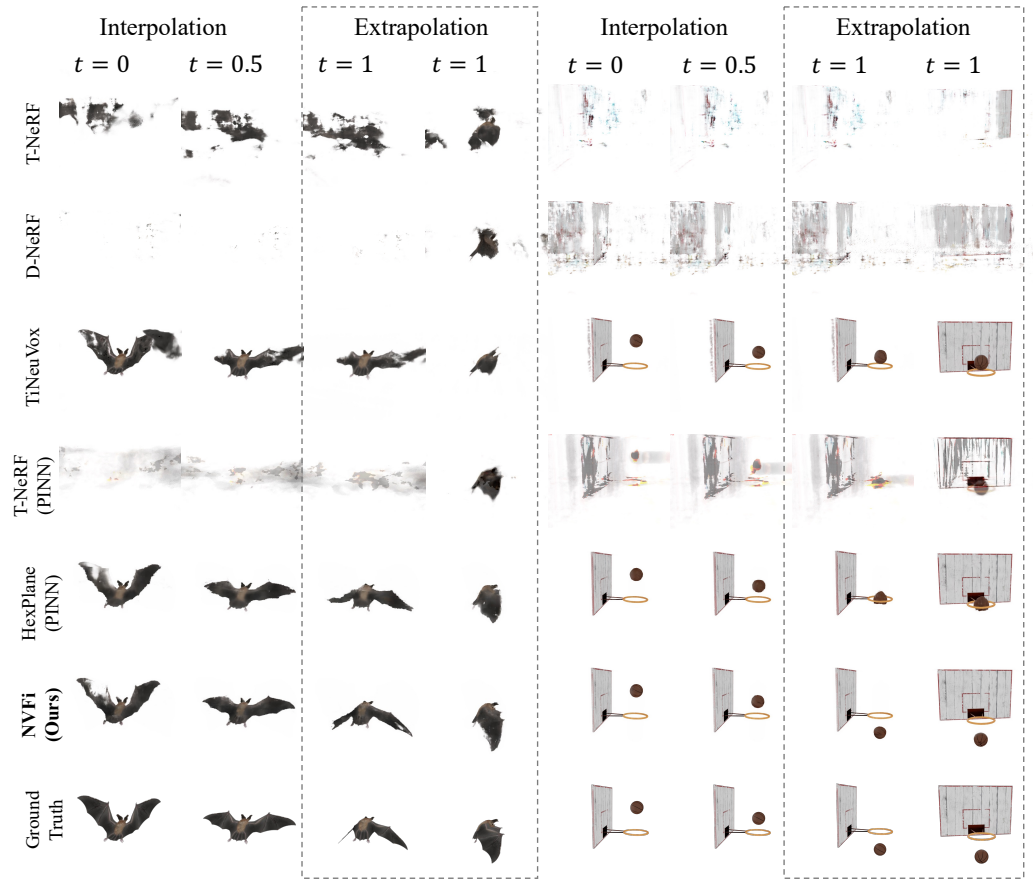
Table 3.6: Quantitative results of ablation studies on Dynamic Object dataset.

Number of Cameras	Interpolation			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
12	29.027	0.970	0.039	27.594	0.972	0.036
6	<u>25.689</u>	<u>0.954</u>	<u>0.051</u>	<u>25.114</u>	<u>0.959</u>	<u>0.122</u>
3	21.460	0.912	0.088	21.370	0.917	0.084

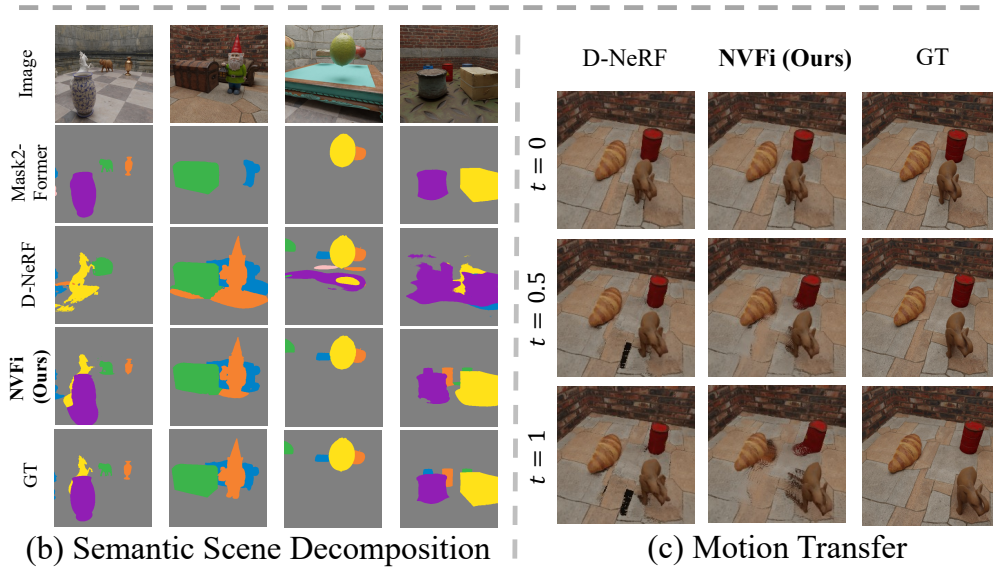
Table 3.5 and Table 3.6 show the ablation results on our Dynamic Object dataset, and we are more interested in the performances for future frame extrapolation. We can see that: 1) The joint keyframe and interframe optimization strategy is critical to enable our method to learn accurate velocity field as well as dynamic radiance field. 2) Once the keyframe number K becomes larger, the extrapolation capability clearly drops, validating that the dense supervision is inferior to help learn physics velocity overall. 3) The extremely sparse camera views are unlikely to capture sufficient visual information for physical motion learning.

3.4 Conclusion

In this chapter, we extend the appealing radiance field to represent dynamic 3D scenes from multi-view videos. Unlike most of the existing methods which focus on novel view synthesis within the training time period, our method learns to disentangle the physical velocity field from the geometry and appearance of 3D scenes by jointly optimizing two neural networks: the keyframe dynamic radiance field and the interframe velocity field. Extensive experiments on three dynamic datasets demonstrate that our framework learns accurate velocity, enabling successful applications in future frame extrapolation, semantic scene decomposition, and motion transfer.



(a) Future Frame Extrapolation



(b) Semantic Scene Decomposition

(c) Motion Transfer

Figure 3.7: Qualitative results of baselines and our method on the three tasks.

Chapter 4

Efficient Physics Learning via Divergence-free Velocity Fields

4.1 Introduction

Obtaining an accurate physical model for complex dynamic 3D environments is a key enabler for many emerging applications, including robot planning, embodied interaction, and dynamic 3D/4D generation. Represented by NeRF [76], 3DGS [49] and their variants [86, 7, 60, 137, 125, 50], recent advancements in 3D learning have demonstrated an unprecedented level of fidelity in modeling dynamic 3D scenes, achieving excellent performance in novel view interpolation and/or 3D shape reconstruction. However, the majority of these approaches lack the core ability of future prediction, essentially because they fail to effectively learn physical properties of 3D space.

To learn physical properties, one line of works introduces various governing PDEs, commonly known as PINN [91], into the learning process of dynamic 3D scenes. While achieving encouraging results in modeling 3D physics such as velocity and viscosity [21, 53, 57, 121], the integration of PINN losses often compromises the accuracy of learned physics in boundary regions, leading to blurred predictions for future frames.

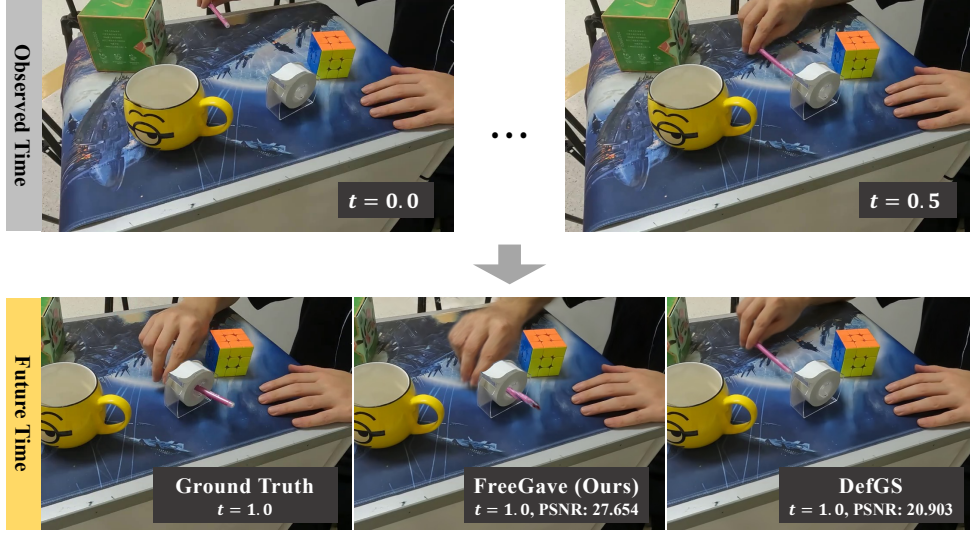


Figure 4.1: Given video frames of a real-world dynamic scene, our FreeGave can learn the underlying physics and accurately predict the future motion of a purple pen going through the hole of a tape, while existing baselines cannot.

Another line of works explicitly incorporates various physics models into neural networks [42, 124, 151, 9], showing impressive results in learning physical properties and further transferring to simulation. Nevertheless, these methods are usually limited to specific types of objects or motions, as the encoded physics models are often less generic than PINN constraints as used by NVFi in Chapter 3, necessitating additional boundary conditions such as object masks.

In this chapter, we aim to present a simple and general method for learning physics of complex dynamic 3D scenes using only multi-view videos, without needing any PINN losses or extra priors such as object types or masks, which are often needed by existing methods. Among various physical properties of a dynamic 3D scene, we follow NVFi [53] as shown in Chapter 3 and focus on learning 3D velocity as it directly governs 3D dynamics and future predictions.

Given video frames containing an unknown number of objects/parts exhibiting diverse motions, at a single glance, it is extremely challenging to learn velocities of all

(unknown) objects or parts in 3D space, due to the lack of physics and semantics in raw pixels. Nevertheless, the whole dynamic 3D scene can be simply regarded as a collection of many 3D rigid particles with their own sizes, orientations, and motion patterns. The dynamics of each particle are actually governed by its hidden physical parameters, such as mass and forces. Thus, our challenge of learning complex 3D dynamics can be naturally framed as learning dynamics of these particles in general.

With this motivation, we propose an elegant pipeline to simultaneously learn 3D geometry, appearance, and velocity of dynamic scenes. Treating each 3D Gaussian kernel as a rigid particle, we extend the successful 3DGS [49] to learn an additional physics code for each 3D Gaussian. With such a code, the 3D velocity of each Gaussian is then learned from visual pixels without relying on PINN losses. Particularly, our pipeline comprises three major components: 1) a **canonical 3D representation module** which adopts a vanilla 3DGS to learn the dynamic 3D scene geometry and appearance at a canonical timestamp; 2) a **neural divergence-free Gaussian velocity module** to estimate per-Gaussian velocity based on the introduced latent physics code; and 3) a **deformation-aided optimization module** to drive the velocity to be physically meaningful.

The core to our pipeline lies in the latter two modules. Basically, given video frames of a dynamic 3D scene, our neural divergence-free **Gaussian velocity** module together with the optimization strategy explicitly drives the per-Gaussian physics code to describe the corresponding motion patterns in latent space. This ensures that the estimated velocities adhere to fundamental physics rules, ultimately enabling precise predictions of future frames. Figure 4.1 shows qualitative results of a challenging dynamic scene.

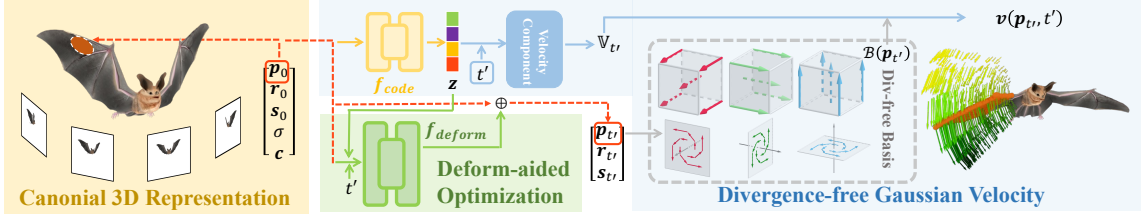


Figure 4.2: The leftmost block shows the canonical 3D Gaussians module, the middle bottom block shows the deformation-aided optimization module and the right block shows our divergence-free Gaussian velocity module.

Our method is named **FreeGave** and our contributions are:

- We introduce a general framework for learning physics of dynamic scenes solely from RGB videos, eliminating the need for any object priors, such as types and masks.
- We propose to learn per-Gaussian velocities from latent physics codes, guided by a carefully designed divergence-free ingredient instead of inefficient PINN losses.
- We showcase superior results in future frame extrapolation on three existing datasets, and a newly collected real-world dataset featuring highly challenging dynamics.

4.2 FreeGave

Given dynamic multi-view RGB videos with known camera poses and intrinsics over time, all frames are denoted as $\left\{ \{ \mathbf{I}_0^1 \cdots \mathbf{I}_0^n \cdots \mathbf{I}_0^N \} \cdots \{ \mathbf{I}_T^1 \cdots \mathbf{I}_T^n \cdots \mathbf{I}_T^N \} \right\}$, where N represents the total number of cameras and T the greatest timestamp in training. Our pipeline is designed to simultaneously learn 3D geometry, appearance, and velocity information of the captured dynamic scene. As shown in Figure 4.2, the first component - canonical 3D representation module - aims to learn a set of 3D Gaussian kernels to represent the 3D scene geometry and appearance in a canonical space, which is elaborated in Section 4.2.1. The second component - neural divergence-free Gaussian

velocity module - attempts to learn the 3D velocity of each Gaussian kernel, which is detailed in Section 4.2.2, while the third module - deformation-aided optimization - is discussed in Section 4.2.3.

4.2.1 Canonical 3D Representation Module

Following the vanilla 3DGS [49], this module aims to learn a set of 3D Gaussian kernels $\mathbf{P}_0$ to represent the canonical scene geometry and appearance at time $t = 0$. Each kernel is parameterized by a 3D position $\mathbf{p}_0$, covariance matrix converted from quaternion $\mathbf{r}_0$ and scaling $\mathbf{s}_0$, opacity σ , and color $\mathbf{c}$ converted from spherical harmonics (SH). Following prior works of dynamic Gaussian [125, 137], we assume the opacity σ and color $\mathbf{c}$ of each Gaussian are not changed over time, but constantly associated and transported with the kernel.

We exactly follow 3DGS to train canonical kernels. Having the N images at time $t = 0$, we first initialize all canonical kernels $\mathbf{P}_0$ randomly or based on sparse point clouds created by SfM [96], and then project kernels into camera space, rendering 2D images and optimizing Gaussian parameters via ℓ_1 and ℓ_{ssim} losses as used in 3DGS:

$$\underbrace{\{\cdot \cdot (\mathbf{p}_0, \mathbf{r}_0, \mathbf{s}_0, \sigma, \mathbf{c}) \cdot \cdot\}}_{\mathbf{P}_0} \xrightleftharpoons[\ell_1 + \ell_{ssim}]{\text{render}} \{\mathbf{I}_0^1 \cdot \cdot \mathbf{I}_0^n \cdot \cdot \mathbf{I}_0^N\} \quad (4.1)$$

4.2.2 Divergence-free Gaussian Velocity Module

This module aims to model the velocity of each Gaussian kernel. A naïve solution is to implement a simple network similar to our prior work NVFi [53] (Chapter 3) to take each Gaussian as input and directly regress the corresponding 3D velocity, followed by commonly-used PINN losses to drive the estimated velocity to be physically meaningful, *i.e.*, divergence-free for volume preserving (or mass conservation). However, there are two key limitations. First, it is extremely hard to learn distinct 3D velocities for neighboring Gaussians exhibiting radically different motions, as a neural network

just models a continuous and smooth function. Second, PINN losses are inefficient in learning physics priors as they involve dense sampling in both spatial and temporal dimensions during training. To this end, we tackle the problem of Gaussian velocity learning by introducing the following new technique ingredients.

Physics Code: For each canonical Gaussian, we propose to learn a vector $\mathbf{z}$, which is regarded as a latent physics code to generally describe the underlying physical information possibly including mass, forces, *etc.*. Basically, this code aims to explain the motion type of a Gaussian kernel over a whole period of time. Therefore, the code $\mathbf{z}$ is shared across all timestamps. In particular, we feed the position of each canonical Gaussian into an MLP-based network f_{code} , directly predicting its physics code $\mathbf{z}$:

$$\mathbf{z} = f_{code}(\mathbf{p}_0), \quad \mathbf{z} \in \mathcal{R}^L \quad (4.2)$$

Technically, we may assign a learnable code to each canonical Gaussian independently. Nevertheless, such a strategy, albeit flexible, involves too many parameters to learn and we empirically find it is inferior as verified in our ablation study. Now, all canonical Gaussians $\mathbf{P}_0$ are as follows:

$$\mathbf{P}_0 \leftarrow \{ \cdots (\mathbf{p}_0, \mathbf{r}_0, \mathbf{s}_0, \sigma, \mathbf{c}, \mathbf{z}) \cdots \} \quad (4.3)$$

Divergence-free Velocity: Having the physics code of a Gaussian kernel, our goal is to estimate its velocity field over the whole time period, without relying on the inefficient PINN losses. However, the estimated velocity field for each Gaussian must satisfy the basic divergence-free property. Otherwise, the extrapolated future frames will not be physically meaningful.

To this end, we treat each canonical Gaussian $\mathbf{p}_0$ as a rigid particle with its size and orientation, whose future motion is governed by a divergence-free velocity field $\mathbf{v}(\mathbf{p}_t, t)$, where $\mathbf{p}_t$ indicates the Gaussian position at time t . To learn the divergence-free velocity field $\mathbf{v}(\mathbf{p}_t, t)$ for a rigid particle, we propose to disentangle $\mathbf{v}(\mathbf{p}_t, t)$ into six basic velocity components: $\mathbb{V}_t \leftarrow [v_t^x, v_t^y, v_t^z, w_t^x, w_t^y, w_t^z]$ and a basis matrix $\mathcal{B}(\mathbf{p}_t)$,

where $v_t^x/v_t^y/v_t^z$ represent the linear velocities along $x/y/z$ axes respectively, and $w_t^z/w_t^y/w_t^x$ represent the angular velocities of the rigid particle around $z/y/x$ axes respectively. Mathematically, the velocity field $\mathbf{v}(\mathbf{p}_t, t)$ is decomposed as follows:

$$\mathbf{v}(\mathbf{p}_t, t) = \mathbb{V}_t \cdot \mathcal{B}(\mathbf{p}_t), \quad \mathbb{V}_t \in \mathcal{R}^{1 \times 6} \quad \mathcal{B}(\mathbf{p}_t) \in \mathcal{R}^{6 \times 3} \quad (4.4)$$

where $\mathcal{B}(\mathbf{p}_t)$ is a set of basis vectors defined as follows:

$$\mathcal{B}(\mathbf{p}_t) = \begin{bmatrix} 1 & 0 & 0 & -p_t^y & p_t^z & 0 \\ 0 & 1 & 0 & p_t^x & 0 & -p_t^z \\ 0 & 0 & 1 & 0 & -p_t^x & p_t^y \end{bmatrix}^T \quad (4.5)$$

where $p_t^x/p_t^y/p_t^z$ represent the values of $\mathbf{p}_t$ at $x/y/z$ axes. The left three columns correspond to the linear velocities $v_t^x/v_t^y/v_t^z$, whereas the right three columns correspond to the angular velocities $w_t^z/w_t^y/w_t^x$. Note that, to guarantee the divergence-free property of $\mathbf{v}(\mathbf{p}_t, t)$, we will design $\mathbb{V}_t$ to be irrelevant to $\mathbf{p}_t$.

For any specific rigid particle at time t , we can easily collect its canonical position $\mathbf{p}_0$ and physics code $\mathbf{z} = f_{code}(\mathbf{p}_0)$. To learn $\mathbb{V}_t$, a naïve solution is to feed $\mathbf{z}$ and t into an MLP-based network, directly decoding a 6D vector. Nevertheless, we empirically find that this is too flexible and results in inferior performance as shown in our ablation study. To tackle this issue, we opt for a simple design illustrated in Figure 4.3.

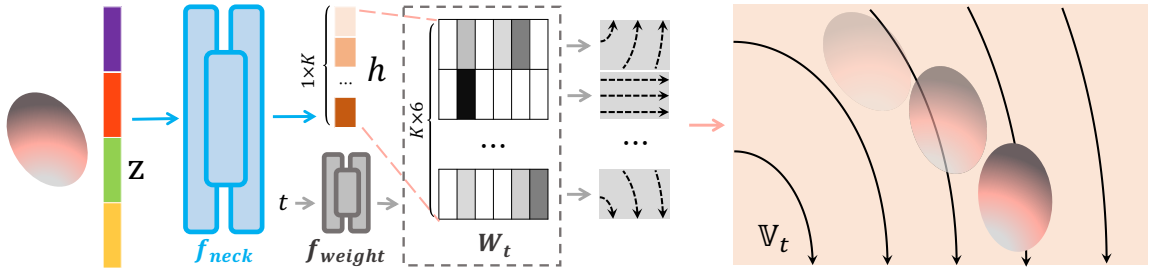


Figure 4.3: An illustration of the network architecture to learn basic velocity component $\mathbb{V}_t$. Each Gaussian kernel is assigned to different learned motion patterns.

In particular, we first feed the physics code $\mathbf{z}$ into several MLP layers f_{neck} , obtaining a bottleneck vector $\mathbf{h} \in \mathcal{R}^K$, where K is a relatively small hyperparameter. In parallel, we feed the timestamp t into another several MLP layers f_{weight} , obtaining a $K \times 6$ vector. This is followed by a reshape operation and we get a matrix $\mathbf{W}_t \in \mathcal{R}^{K \times 6}$. The basic velocity component $\mathbb{V}_t$ is obtained by a matrix multiplication as follows:

$$\mathbb{V}_t = \mathbf{h} \cdot \mathbf{W}_t = f_{neck}(\mathbf{z}) \cdot f_{weight}(t) \quad (4.6)$$

Basically, our insight behind this simple design is that: the physics code $\mathbf{z}$ tends to be decoded into K types of motion patterns by f_{neck} , whereas f_{weight} tends to generate a set of weights to select the decoded motion patterns at a particular timestamp t . Each motion pattern serves as a motion agent, whose value is simply zero for those static patterns. Since there could be not only one static pattern, the weights matrix f_{weight} tends to be sparse. Empirically, such a simple architecture achieves superior results, and we leave more sophisticated designs for future exploration.

Proof of Divergence-free Property: In order to prove the divergence-free property, we just need to show $\nabla_{\mathbf{p}_t} \cdot \mathbf{v}(\mathbf{p}_t, t) = 0$. Since $\mathbb{V}_t$ is totally irrelevant to $\mathbf{p}_t$, we only need to show each basis vector function follows $\nabla_{\mathbf{p}_t} \cdot \mathcal{B}^k(\mathbf{p}_t) = 0$, *i.e.*:

$$\nabla_{\mathbf{p}_t} \cdot \mathbf{v}(\mathbf{p}_t, t) = \nabla_{\mathbf{p}_t} \cdot (\mathbb{V}_t \cdot \mathcal{B}(\mathbf{p}_t)) = \sum_{k=1}^6 \mathbb{V}_t^k \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^k(\mathbf{p}_t) = 0.$$

Next, we show that each of the basis vector functions is divergence-free:

$$\begin{aligned} \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^1(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = \frac{\partial 1}{\partial p_t^x} = 0; \\ \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^2(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} = \frac{\partial 1}{\partial p_t^y} = 0; \\ \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^3(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = \frac{\partial 1}{\partial p_t^z} = 0; \\ \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^4(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} -p_t^y & p_t^x & 0 \end{bmatrix} = \frac{\partial -p_t^y}{\partial p_t^x} + \frac{\partial p_t^x}{\partial p_t^y} = 0; \\ \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^5(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} p_t^z & 0 & -p_t^x \end{bmatrix} = \frac{\partial p_t^z}{\partial p_t^x} + \frac{\partial -p_t^x}{\partial p_t^z} = 0; \\ \nabla_{\mathbf{p}_t} \cdot \mathcal{B}^6(\mathbf{p}_t) &= \nabla_{\mathbf{p}_t} \cdot \begin{bmatrix} 0 & -p_t^z & p_t^y \end{bmatrix} = \frac{\partial -p_t^z}{\partial p_t^y} + \frac{\partial p_t^y}{\partial p_t^z} = 0. \end{aligned}$$

4.2.3 Deformation-aided Optimization

Regarding each Gaussian kernel, we need to effectively optimize its physics code $\mathbf{z}$ and basic velocity components $\mathbb{V}_t$, such that the corresponding per-Gaussian velocity field $\mathbf{v}(\mathbf{p}_t, t)$ is physically meaningful. However, this is particularly challenging because there are only multi-view images at sparse timestamps as supervision without any other priors. We empirically find that directly training all networks is hard to converge as shown in our ablation study.

In this work, we introduce an auxiliary deformation field to aid the optimization process of our pipeline. In particular, we feed each canonical Gaussian kernel $\mathbf{p}_0$, timestamp t , and its physics code $\mathbf{z}$ into an MLP-based network f_{deform} , predicting the corresponding position displacement $\delta\mathbf{p}$, and the change of quaternion $\delta\mathbf{r}$ and scaling $\delta\mathbf{s}$ from timestamp 0 to t . Naturally, all Gaussians $\mathbf{P}_t$ at time t can be computed by transporting all canonical Gaussians as follows. The deformation operation $\circ$ follows [137], representing quaternion multiplication.

$$\begin{aligned}
 (\delta\mathbf{p}, \delta\mathbf{r}, \delta\mathbf{s}) &= f_{deform}(\mathbf{p}_0, t, \mathbf{z}), \quad \mathbf{z} = f_{code}(\mathbf{p}_0) \\
 \mathbf{p}_t &= \mathbf{p}_0 + \delta\mathbf{p}, \quad \mathbf{r}_t = \mathbf{r}_0 \circ \delta\mathbf{r}, \quad \mathbf{s}_t = \mathbf{s}_0 \cdot \delta\mathbf{s} \\
 \mathbf{P}_t &\leftarrow \{\cdots (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdots\}
 \end{aligned} \tag{4.7}$$

With our design of physics code $\mathbf{z}$, basic velocity components $\mathbb{V}_t$, and the deformation field f_{deform} , we now discuss how to link and train them together, ultimately driving the learned per-Gaussian velocity field to be truly learned.

First, given training images $\{\{\mathbf{I}_0^1 \cdots \mathbf{I}_0^N\} \cdots \{\mathbf{I}_T^1 \cdots \mathbf{I}_T^N\}\}$ and the canonical Gaussians $\mathbf{P}_0$, we sample a timestamp t from training set and another timestamp t' , where $\Delta t = t - t'$ is predefined to be small enough. We can obtain Gaussians $\mathbf{P}_{t'}$ via the auxiliary deformation field f_{deform} (Eq 4.7):

$$\mathbf{P}_{t'} \leftarrow \{\cdots (\mathbf{p}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}, \mathbf{z}) \cdots\} \tag{4.8}$$

Second, we feed the set of Gaussians $\mathbf{P}_{t'}$ at time t' and the sampled interval Δt into our interleaved mid-point Algorithm 2, predicting the set of Gaussians $\mathbf{P}_t$ at time t transported by our per-Gaussian velocity field.

Algorithm 2 Interleaved mid-point method.

Input: (1) Gaussians at t' : $\{\cdot \cdot (\mathbf{p}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}, \mathbf{z}) \cdot \cdot\}$, and (2) Δt

Output: Gaussians at t , *i.e.*, at $(t' + \Delta t)$: $\{\cdot \cdot (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdot \cdot\}$

- $\mathbf{v}(\mathbf{p}_{t'}, t) \leftarrow f_{neck}(\mathbf{z}) \cdot f_{weight}(t') \cdot \mathcal{B}(\mathbf{p}_{t'})$, (ref to Eqs 4.4 / 4.5 / 4.6)
- $\mathbf{p}_{mid} \leftarrow \mathbf{p}_{t'} + \frac{\Delta t}{2} \mathbf{v}(\mathbf{p}_{t'}, t)$
- $\mathbf{v}_{mid} \leftarrow f_{neck}(\mathbf{z}) \cdot f_{weight}(t' + \frac{\Delta t}{2}) \cdot \mathcal{B}(\mathbf{p}_{mid})$
- $\mathbf{p}_t \leftarrow \mathbf{p}_{t'} + \Delta t \mathbf{v}_{mid}$
- Convert quaternion to rotation: $\mathbf{R}_{t'} \leftarrow \mathbf{r}_{t'}$
- $\mathbf{R}_t \leftarrow (\mathbf{I} + \Delta t \frac{\partial \mathbf{v}_{mid}}{\partial \mathbf{p}_{mid}}) \mathbf{R}_{t'}$, (following the approximation in [128])
- Convert rotation to quaternion: $\mathbf{r}_t \leftarrow \mathbf{R}_t$
- Assign $\mathbf{s}_{t'}$ to $\mathbf{s}_t$: $\mathbf{s}_t \leftarrow \mathbf{s}_{t'}$, (each Gaussian is a rigid particle with its size fixed over time, ensuring the divergence-free property)

Return: All gaussians at t : $\{\cdot \cdot (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdot \cdot\}$

Third, having the set of Gaussians $\mathbf{P}_t$ transported from t' by our per-Gaussian velocity field, we project them to 2D space, rendering RGBs supervised by the total N training views $\{\mathbf{I}_t^1 \cdot \cdot \mathbf{I}_t^N\}$ at time t via the following standard 3DGS losses. Notably, similar as standard 3DGS reconstruction, when an object or an object part is observed only in the later timestamp but not in the canonical timestamp, the corresponding Gaussians will be densified according to the supervision at later timestamps.

$$\{\mathbf{P}_0, f_{code}, f_{neck}, f_{weight}, f_{deform}\} \leftarrow (\ell_1 + \ell_{sim}) \quad (4.9)$$

4.3 Experiments

Datasets: Our method focuses on learning physical information from dynamic videos for accurate future frame predictions, rather than fitting training views for interpolation. In this regard, the closest work to us is NVFi [53], and we mainly evaluate our method on its two synthetic dynamic datasets: **1) Dynamic Object Dataset** which consists of 6 dynamic objects undergoing different rigid or deformable motions; and **2) Dynamic Indoor Scene Dataset** which comprises 4 complex indoor scenes with 3 to 6 objects exhibiting challenging motions. For the sake of motion diversity, we also evaluate another synthetic **3) ParticleNeRF Dataset**[1], which includes various motions from robot manipulation to harmonic oscillation. All datasets have test splits for *novel view interpolation* and *future frame extrapolation*.

More details about Dynamic Object Dataset and Dynamic Indoor Scene Dataset are in Section 3.3. For ParticleNeRF Dataset, it includes 6 challenging dynamic objects.

- **Object #1: Robot:** a robot arm waving from one side to another side.
- **Object #2: Robot Task:** a robot arm putting a box onto a sliding platform.
- **Object #3: Wheel:** a constant rotating wheel.
- **Object #4: Spring:** a box tied on a spring, which undergoes a harmonic oscillation motion. The training and the test observation period in total form a whole oscillation period.
- **Object #5: Pendulums:** two swing pendulums, each undergoing harmonic oscillation motion. The training and the test observation period in total form a whole oscillation period.
- **Object #6: Cloth:** a flat cloth being folded.

Each scene contains 40 viewing angles. For Object #1 & #2, we choose the first

53 frames from 36 selected viewing angles as the training split, *i.e.* 1908 frames in total, and the first 53 frames from the other 4 viewing angles for evaluating novel view interpolation within the training time period, *i.e.*, 212 frames in total. All the remaining 17 frames from 40 viewing angles are used to evaluate future frame extrapolation, *i.e.*, 680 frames.

For Object #3 & #4 & #5 & #6, the first 104 frames from 36 selected viewing angles are used as the training split, *i.e.*, 3744 frames in total, and the first 104 frames from the other 4 viewing angles are used for evaluating novel view interpolation within the training time period, *i.e.*, 416 frames in total. All the remaining 36 frames from 40 viewing angles are used to evaluate future frame extrapolation, *i.e.*, 1440 frames.

While NVFi [53] is evaluated on a real-world dataset, it just has two simple scenes selected from NVIDIA Dynamic Scene Dataset [139] which is not primarily collected for physics learning. Since this field of study is still in its infancy and there is lack of real-world datasets in high-quality, we opt to collect a new real-world dataset particularly for benchmarking 3D physics learning, which is the first of its kind to the best of our knowledge and will be made public.

4) FreeGave-GoPro Dataset: We capture 6 dynamic scenes with 20 GoPro cameras (GoPro Hero 10 Black). All cameras are rigged on tripods and placed around an operation platform, as illustrated in Figure 4.4. The 6 dynamic scenes consist of particularly challenging physical motions such as a moving object is tightly beside a static object, or a thin object is moving through a tiny hole of another object.

We use voice to synchronize all videos and use COLMAP [96] on the first frames of all 20 views to estimate camera poses. Since all cameras are fixed under a stabilizing mode, the camera poses do not change over time. For each dynamic scene, we select 89 frames from each view, and resize images to be a resolution of 960×540 . We reserve the first 67 frames at 17 picked viewing angles as the training split, *i.e.*, 1139 frames, while leaving the 67 frames at the remaining 3 viewing angles for evaluating

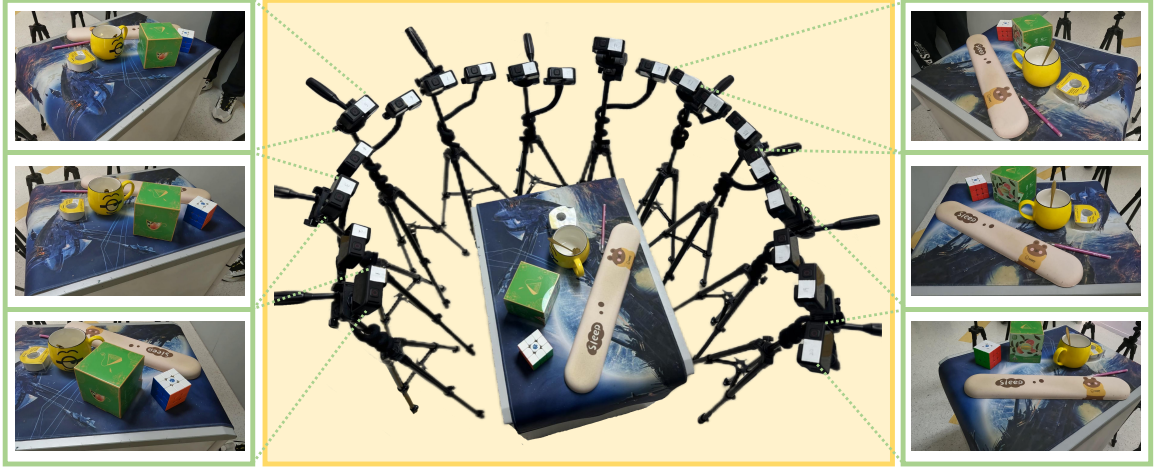


Figure 4.4: The camera settings for our dataset collection.

novel view interpolation within the training time period, *i.e.*, 201 frames. We keep the last 22 frames at all 20 viewing angles for evaluating *future frame extrapolation*, *i.e.*, 440 frames in total. The details of each of the scenes are listed below:

- **Scenes #1/#2: Pen & Tape.** There is a person holding a pen and trying to pass it through the hole of a static tape. The difference between these two scenes is that there are more static objects in the second scene, introducing more visual occlusions and requiring more accurate separation between moving areas and static areas. The difficulty lies in the motion of one object which is going to penetrate through another in future.
- **Scene #3: Box.** This scene contains a drawer-like box, and a person is trying to close it. The difficulty lies in a tight combination of the moving part and the static part of the box, especially in future.
- **Scene #4: Hammer.** This scene contains a hammer moving on the topside of a box. The difficulty lies in the direct contact of moving objects and static objects, which requires sharp separation of diverse motion patterns in order to keep right static/moving states in future.

- **Scene #5: Collision.** This scene contains a cube and a cup moving towards each other. The difficulty is the different directions of two motions. It is hard to keep the shapes of these two objects in future.
- **Scene #6: Wrist Rest.** A person is trying to bend a wrist rest. The difficulty is that the object is deformable and the motion is not rigid or part-wise rigid.

Baselines: We choose the following baselines: **1) NVFi** [53]. It is the closest work to us, but relies on PINN losses to learn physics priors whereas we directly build a divergence-free velocity field for each rigid particle. **2) T-NeRF** [86], **3) D-NeRF** [86], **4) NSFF** [59], and **5) TiNeuVox** [26]. All these baselines adopt NeRF framework as the backbone, thus being short in 3D reconstruction compared to 3DGS. As our method adopts 3DGS, for a fair comparison, we also include the following two baselines. **6) DefGS** [137]. This recent method Deformable 3D Gaussians is excellent to model dynamic 3D scenes for novel view interpolation, but lacks the ability for future prediction. In this regard, we build the most comparable baseline to us **7) DefGS_{nvfi}** by combining DefGS with the velocity field proposed by NVFi. It shows strong ability in novel view synthesis (inherited from DefGS) and extrapolation ability (inherited from NVFi framework). To be fair, we combine the original deformation field of DefGS with exactly the same velocity field of NVFi. The same deformation-aided optimization as our FreeGave is used to train DefGS_{nvfi}.

Metrics: The standard metrics **PSNR**, **SSIM**, and **LPIPS** are reported for RGB view synthesis in two tasks: novel view interpolation and future frame extrapolation.

Implementation Details: We implement our networks by MLPs, and the detailed configurations are as follows:

- f_{code} : This network includes 4 MLP layers and each hidden layer has a size of 128 neurons followed by ReLU activation. In addition, we make a positional encoding for input positions with degree of 8. And we set the dimension of the

physics code L as 32 for Dynamic Indoor Scene dataset and 16 for the others.

- f_{neck} : This network has MLP layers as $L \mapsto 4L \mapsto 4L \mapsto K$. We choose the bottleneck vector dimension K as 32 for Dynamic Indoor Scene dataset and 16 for other two datasets.
- f_{weight} : This network includes 5 MLP layers and each hidden layer has a size of 128 neurons. In addition, we make a degree of 8 positional encoding for input positions, and we add a ResNet connection on the 3rd layer. The output vector has a dimension of $K * 6$ and is reshaped as a matrix of $K \times 6$.
- f_{deform} : For two synthetic datasets, this network includes 6 MLP layers and each hidden layer has a size of 128 neurons, while it includes 8 MLP layers and each hidden layer has a size of 256 neurons for the challenging FreeGave-GoPro dataset. In addition, we also make a degree of 8 positional encoding for input positions, and the physics code $\mathbf{z}$ is concatenated to the encoded position. We also add a ResNet connection on the third layer.

4.3.1 Evaluation for Future Frame Extrapolation

We evaluate our method on all four datasets for both *novel view interpolation* and *future frame extrapolation*. All baselines are evaluated on the Dynamic Object and Dynamic Indoor Scene Dataset. We select 4 representative baselines to compare on our new FreeGave-GoPro Dataset, including **TiNeuVox**, **NVFi**, **DefGS** and **DefGS_{nvfi}**.

Results & Analysis: From Tables 4.1&4.2, it can be seen that: 1) Our FreeGave achieves the best performance on all datasets, clearly surpassing all baselines by a large margin on the challenging task of future frame extrapolation, showing that our method can indeed learn the underlying physical information, thus predicting accurate future motions. 2) Our method also achieves better scores on the task of novel view interpolation, notably outperforming the strong baselines DefGS and DefGS_{nvfi}.

Table 4.1: Quantitative results of all methods for both novel view interpolation and future frame extrapolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset, and future frame extrapolation on ParticleNeRF Dataset. Bold numbers indicate the best performance.

	Dynamic Object Dataset						Dynamic Indoor Scene Dataset						ParticleNeRF Dataset		
	Interpolation			Extrapolation			Interpolation			Extrapolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
T-NeRF [86]	13.163	0.709	0.353	13.818	0.739	0.324	24.944	0.742	0.336	22.242	0.700	0.363	-	-	-
D-NeRF [86]	14.158	0.697	0.352	14.660	0.737	0.312	25.380	0.766	0.300	20.791	0.692	0.349	-	-	-
T-NeRF _{PINN}	15.286	0.794	0.293	16.189	0.835	0.230	16.250	0.441	0.638	17.290	0.477	0.618	-	-	-
HexPlane _{PINN}	27.042	0.958	0.057	21.419	0.946	0.067	25.215	0.763	0.389	23.091	0.742	0.401	-	-	-
NSFF [59]	-	-	-	-	-	-	29.365	0.829	0.278	24.163	0.795	0.289	-	-	-
TiNeuVox [26]	27.988	0.960	0.063	19.612	0.940	0.073	29.982	0.864	0.213	21.029	0.770	0.281	20.407	0.888	0.102
NVFi [53]	29.027	0.970	0.039	27.594	0.972	0.036	<u>30.675</u>	0.877	0.211	29.745	0.876	0.204	18.173	0.867	0.119
DefGS [137]	<u>37.865</u>	<u>0.994</u>	<u>0.007</u>	19.849	0.949	0.045	29.926	<u>0.916</u>	<u>0.130</u>	21.380	0.819	0.188	19.205	0.900	0.083
DefGS _{nvfi}	37.316	<u>0.994</u>	0.008	<u>28.749</u>	<u>0.984</u>	<u>0.013</u>	30.170	0.915	0.133	<u>31.096</u>	<u>0.945</u>	<u>0.077</u>	<u>22.730</u>	<u>0.931</u>	<u>0.050</u>
FreeGave (Ours)	39.393	0.995	0.005	31.987	0.990	0.007	32.287	0.930	0.092	35.019	0.966	0.051	26.657	0.956	0.030

Table 4.2: Quantitative results for both novel view interpolation and future frame extrapolation on FreeGave-GoPro Dataset.

	FreeGave-GoPro Dataset					
	Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
TiNeuVox [26]	19.026	0.740	0.319	20.600	0.760	0.292
NVFi [53]	18.947	0.739	0.322	22.747	0.756	0.393
DefGS [137]	<u>28.408</u>	0.920	0.091	21.344	0.858	0.170
DefGS _{nvfi}	28.217	0.920	<u>0.092</u>	<u>26.431</u>	<u>0.903</u>	<u>0.124</u>
FreeGave (Ours)	28.451	0.920	0.091	28.094	0.914	0.112

We hypothesize that the effective incorporation of physics priors allows our method to fully utilize all Gaussian kernels for representing 3D geometry and appearance. As shown in Figure 4.6, 4.5, 4.7, 4.8, 4.9, 4.10, our method makes promising predictions: 1) in dynamic multi-object scenes, such as the pen-through-tape scene, where only our method succeeds while others all fail; 2) in scenes with complex motions, such as the harmonic oscillations in pendulums and spring scenes, where only our method captures the right periodic pattern with the shape being kept while others all fail.

Though not primarily collected for physics learning, we also evaluate our model on two simple scenes from NVIDIA Dynamic Scene Dataset [139] selected by NVFi [53]. The quantitative results are shown in Table 4.3.

Table 4.3: Results on NVIDIA Dynamic Scene Dataset.

	Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
TiNeuVox	28.304	0.868	0.216	24.556	0.863	0.215
NVFi	27.138	0.844	0.231	<u>28.462</u>	0.876	0.214
DefGS	26.662	<u>0.893</u>	<u>0.127</u>	24.240	0.895	0.140
DefGS _{nvfi}	26.972	0.890	0.128	27.529	<u>0.927</u>	<u>0.102</u>
FreeGave (Ours)	<u>27.345</u>	0.896	0.097	29.005	0.933	0.072

Analysis of Computational Costs: We calculate the average time and memory consumption in training, the average speed and memory consumption in test, and model sizes for most baselines in Table 4.4. We can see that: 1) our method has a clear advantage over the strong baseline DefGS_{nvfi} in terms of time and memory cost in training, thanks to our new and more efficient divergence-free velocity module over the PINN loss; 2) our method is generally better or on par with other baselines (TiNeuVox / NVFi / DefGS) in computation cost of training and test, but our method demonstrates significantly better extrapolation results (as shown in Tables 4.1&4.2).

Table 4.4: The average time (hours) and GPU memory (GB) cost for training, the inference speed (fps), GPU memory (GB), and model size (MB) for test on all three datasets.

	Dynamic Object Dataset					Dynamic Indoor Scene Dataset					FreeGave-GoPro Dataset				
	Training		Test			Training		Test			Training		Test		
	Time $\downarrow$	Mem $\downarrow$	fps $\uparrow$	Mem $\downarrow$	Size $\downarrow$	Time $\downarrow$	Mem $\downarrow$	fps $\uparrow$	Mem $\downarrow$	Size $\downarrow$	Time $\downarrow$	Mem $\downarrow$	fps $\uparrow$	Mem $\downarrow$	Size $\downarrow$
TiNeuVox	0.5	8.0	0.40	2.3	<u>49.8</u>	0.6	<u>8.2</u>	0.51	<u>3.7</u>	49.9	0.6	<u>9.4</u>	0.16	<u>4.7</u>	49.6
NVFi	2.2	22.6	0.11	16.5	114.7	2.3	21.5	0.34	16.8	107.9	2.3	23.3	0.03	23.1	121.4
DefGS	<u>0.8</u>	<u>6.4</u>	<u>19.40</u>	<u>5.0</u>	53.1	<u>0.8</u>	5.9	21.9	4.0	98.1	<u>1.3</u>	8.4	3.22	3.4	<u>88.5</u>
DefGS _{nvfi}	2.1	22.7	13.59	5.2	54.5	6.0	26.4	10.98	4.1	101.2	8.0	32.6	2.90	5.8	92.1
FreeGave (Ours)	<u>0.8</u>	5.7	19.70	4.0	27.0	1.5	10.4	<u>13.60</u>	3.1	<u>55.8</u>	1.9	16.1	<u>3.03</u>	8.1	115.7



Figure 4.5: Qualitative results for future frame extrapolation on ParticleNeRF Dataset.

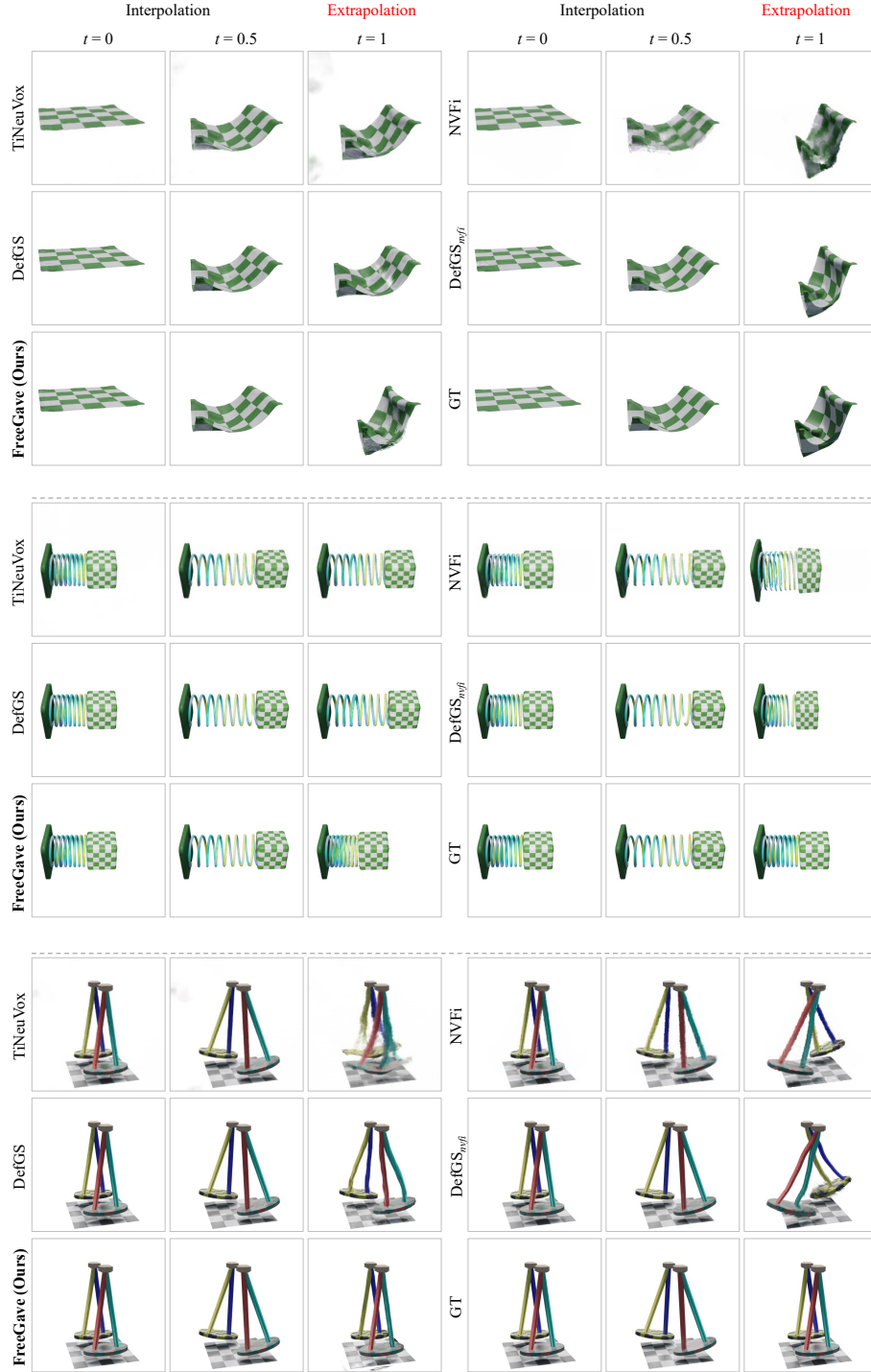


Figure 4.6: Qualitative results for future frame extrapolation on ParticleNeRF Dataset.

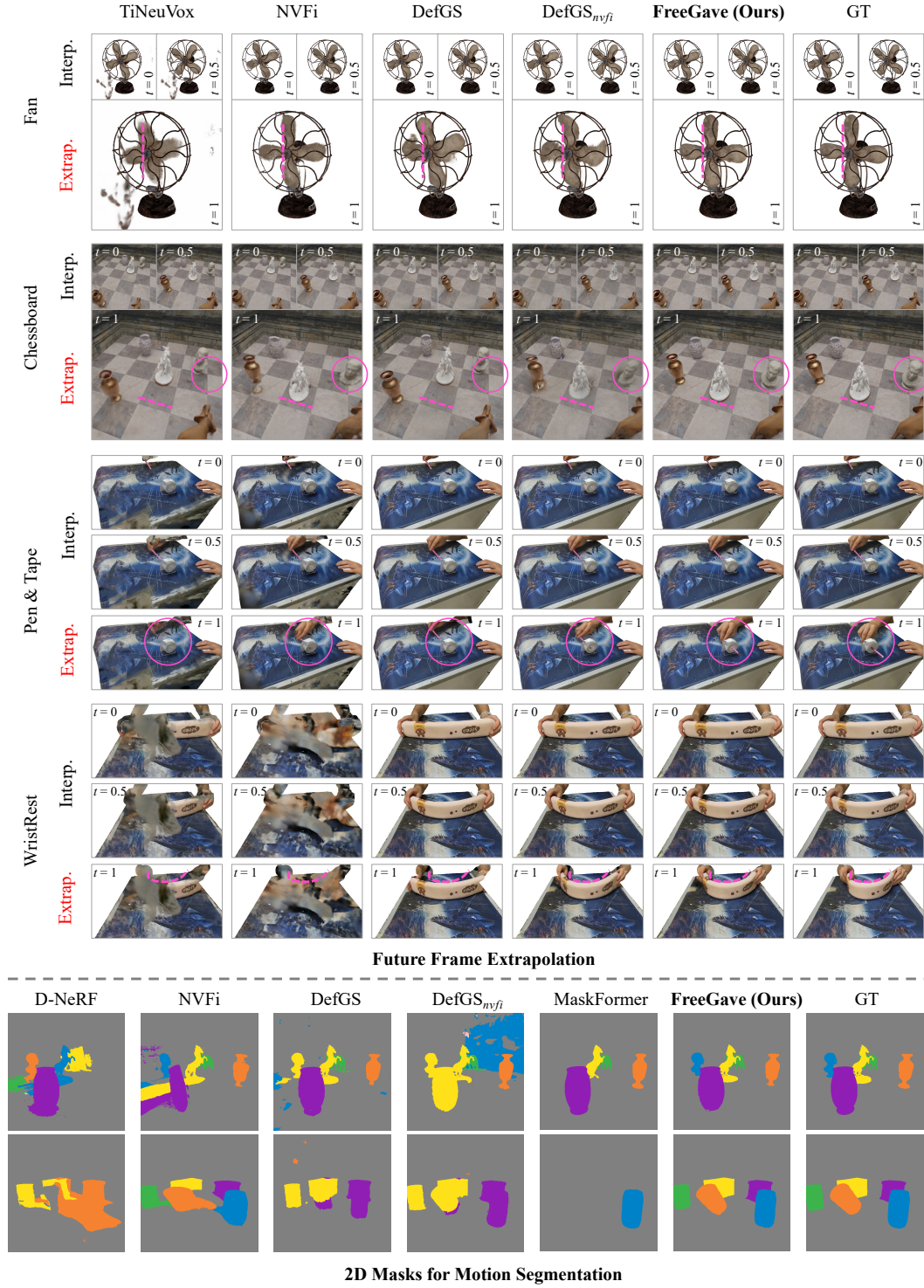


Figure 4.7: Qualitative results of our method and baselines for future frame extrapolation and unsupervised motion segmentation.

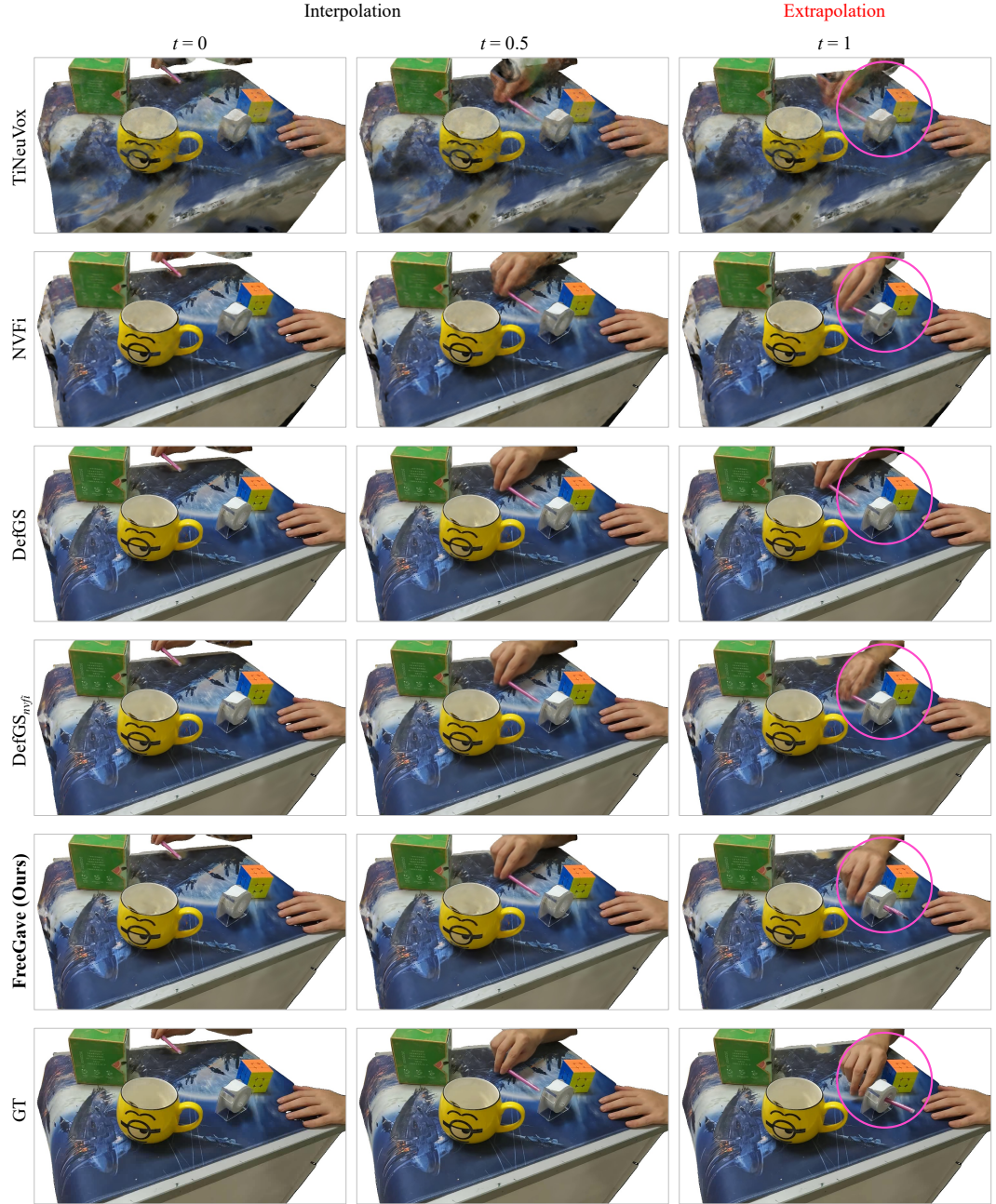


Figure 4.8: Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.



Figure 4.9: Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.

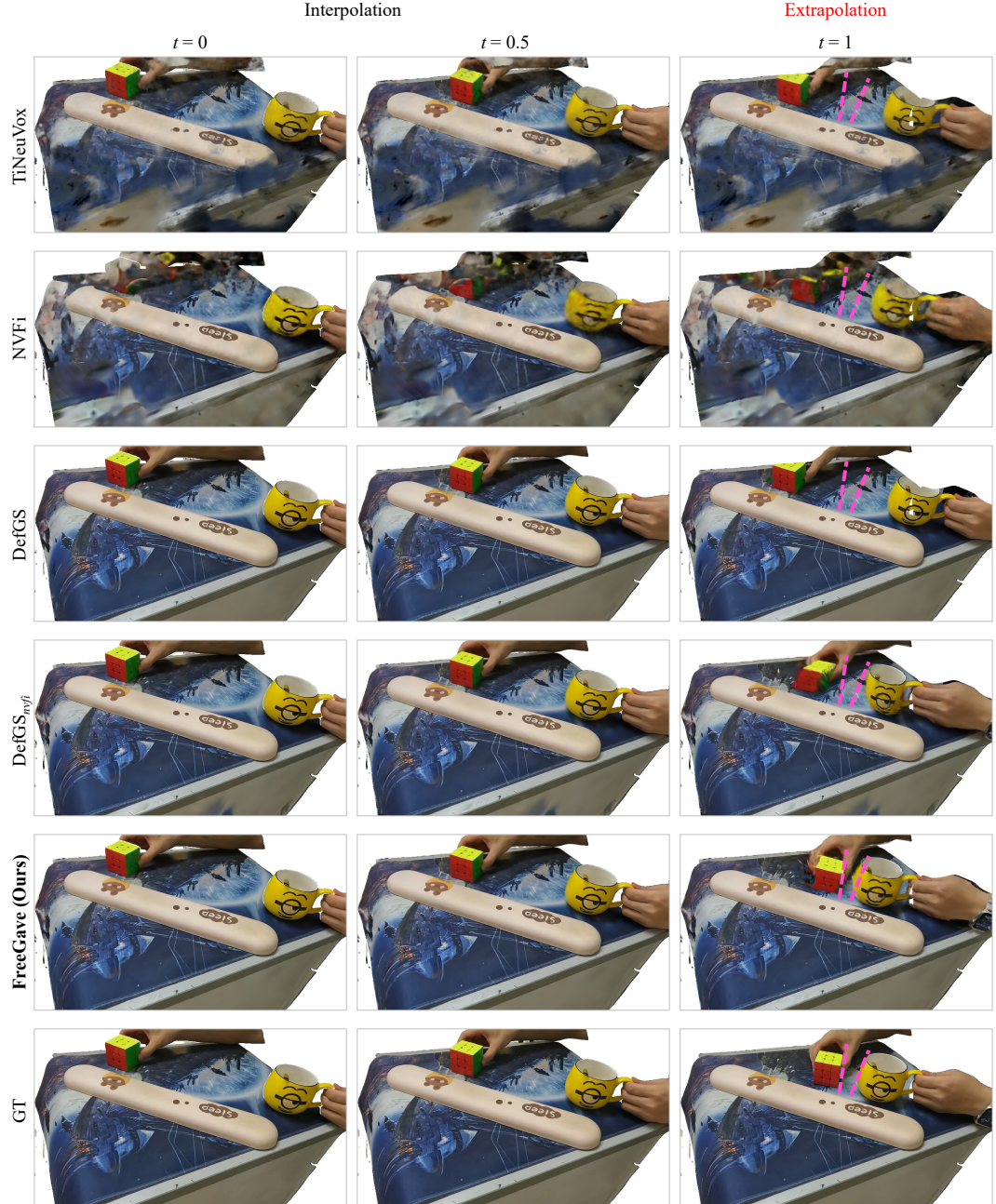


Figure 4.10: Qualitative results of our method and baselines for future frame extrapolation on FreeGave-GoPro Dataset.

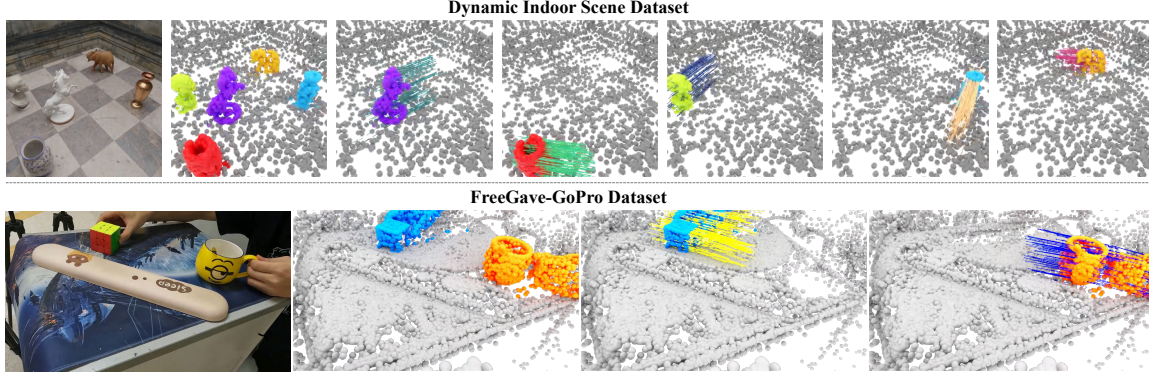


Figure 4.11: Qualitative results of decomposed motion patterns (colored objects) and their corresponding velocity fields (colored trajectories).

4.3.2 Analysis of Physics Code and Motion Patterns

Our divergence-free Gaussian velocity module is designed to learn the velocity field for each Gaussian particle. With our design of bottleneck vector decomposition, our model can naturally group Gaussian kernels according to the learned motion patterns. To validate this nice property of bottleneck vector, we conduct the following experiments.

For each Gaussian, we query its learned bottleneck vectors $\mathbf{h} = f_{neck}(\mathbf{z})$, and query its learned velocity by uniformly sampling 10 timestamps to form a velocity trajectory $\{\mathbf{v}_0, \dots, \mathbf{v}_{10}\}$. Then, we apply K-means [72] to group the learned bottleneck vectors $\mathbf{h}$ into C groups. We color the velocity trajectory based on the velocity direction. As shown in Figure 4.11, we can see that Gaussian kernels in the same group share the same velocity trajectory, while Gaussian kernels in different groups have totally different motion trajectories.

We further quantitatively evaluate the objectness after grouping our learned physics codes on Dynamic Indoor Scene dataset. In particular, we follow [138] to render the group masks as 2D object segmentation masks for all 30 views over 60 timestamps on all 4 scenes, *i.e.*, 7200 images in total. We compare our performance with **D-NeRF**, **NVFi**, **DefGS** and **DefGS_{nvfi}**. We follow NVFi as shown in Section 3.3.2 to get

Table 4.5: Quantitative results of scene decomposition on the Synthetic Indoor Scene Dataset.

	AP $\uparrow$	PQ $\uparrow$	F1 $\uparrow$	Pre $\uparrow$	Rec $\uparrow$	mIoU $\uparrow$
Mask2Former [18]	65.37	73.14	78.29	<u>94.83</u>	68.88	64.42
D-NeRF [86]	57.26	46.15	59.02	56.55	62.94	46.58
NVFi [53]	<u>91.21</u>	<u>78.74</u>	<u>93.75</u>	93.76	<u>93.74</u>	<u>67.64</u>
DefGS [137]	51.73	57.60	66.43	63.21	70.07	54.46
DefGS _{nvfi}	55.26	62.75	69.83	69.39	72.91	56.82
FreeGave (Ours)	99.75	96.27	99.99	99.99	99.93	80.52

the segmentation details of D-NeRF and NVFi. In addition, we include a powerful image-based object segmentation method, Mask2Former [18] pre-trained on COCO [62] dataset, as a fully-supervised baseline.

For the 3DGS based baselines, we also adopt OGC [102, 103] to segment Gaussians based on scene flows calculated from their learned deformation fields. Given a well-trained DefGS or DefGS_{nvfi} model with N canonical Gaussian kernels, we first assign learnable per-Gaussian object codes $\mathbf{O} \in (0, 1)^{N \times K}$ to all Gaussian kernels, where K is the maximum number of objects that is expected to appear in the scene. After that, we query the position displacements for Gaussians kernels from the well-trained deformation field at time 0 and t respectively, thus obtaining the Gaussians $\mathbf{P}_0$ at time 0 and $\mathbf{P}_t$ at time t . Then the per-Gaussian scene flows $\mathbf{M}_t$ from time 0 to t is calculated as $\mathbf{M}_t = \mathbf{P}_t - \mathbf{P}_0$. Lastly, two losses proposed in OGC [102] are computed on the learnable object codes. The details of the two losses are in Section 3.3.2.

As shown in Table 4.5 and Figure 4.7 and 4.12, we can see that our method achieves nearly perfect object segmentation results on all metrics, clearly outperforming all baselines. This shows that our learned physics codes truly represent object physical motion patterns which can be easily used to identify individual objects accordingly without needing any human labels to train a separate segmentation network.

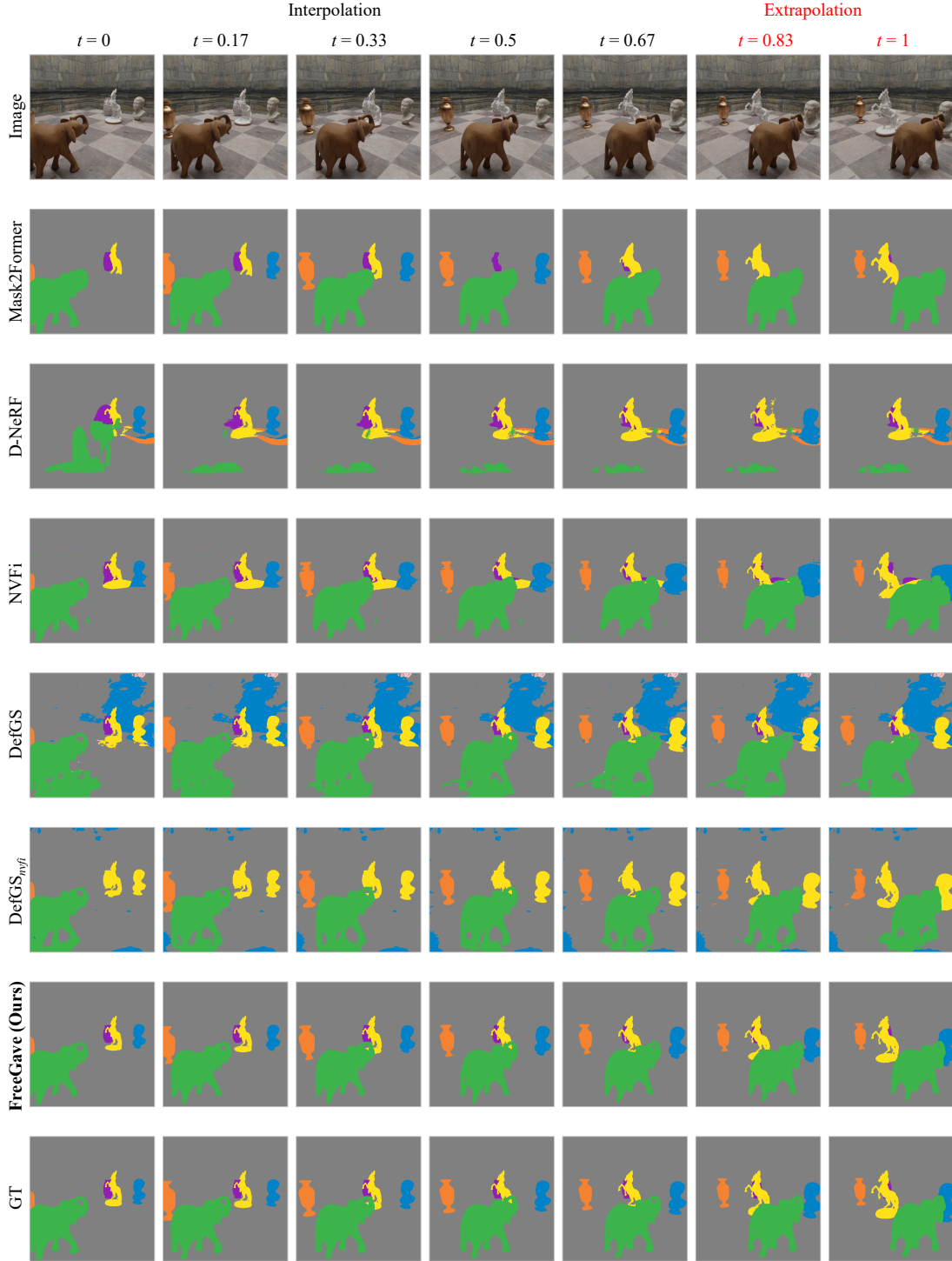


Figure 4.12: Qualitative results for unsupervised motion segmentation on “Chessboard” of Dynamic Indoor Scene Dataset.

4.3.3 Ablation Study

Our framework comprises 3DGS as the scene representation backbone and the neural divergence-free Gaussian velocity module with a deformation-aided optimization module. We conduct the following groups of ablation experiments to validate the choices of our method. All ablation experiments are conducted on the Dynamic Objects dataset.

- (1) **Replacing f_{code} by a learnable code $\mathbf{z}$ in Eq 4.2:** Instead of feeding the position of each canonical Gaussian to f_{code} to learn its physics code $\mathbf{z}$, we assign a learnable $\mathbf{z} \in \mathcal{R}^L$ for each Gaussian respectively.
- (2) **Removing divergence-free parametrization:** Instead of decomposing the velocity field by divergence-free parametrization in Eqs 4.4/4.5/4.6, we directly define the velocity field as $\mathbf{v}(\mathbf{p}_t, t) = f_{neck}(\mathbf{z}) \cdot f_{motion}(\mathbf{p}_t, t)$, where $f_{motion} : \mathcal{R}^4 \mapsto \mathcal{R}^K$ is an MLP-based network. Notably, now f_{motion} is based on $\mathbf{p}_t$, so it is not divergence-free.
- (3) **Removing bottleneck vector decomposition for velocity components in Eq 4.6:** Instead of decomposing velocity components network $\mathbb{V}_t$ by bottleneck vector in Eq 4.6, we directly define $\mathbb{V}_t$ as a single MLP: $\mathbb{V}_t = \text{MLP}(\mathbf{z}, t)$.
- (4) **Different choices of motion patterns K :** We compare 3 choices for the number of motion patterns in f_{neck} / f_{weight} : $\{8, 16, 32\}$, while $K=16$ in our main experiments.
- (5) **Removing the auxiliary deformation field f_{deform} :** To validate the necessity of our deformation-aided optimization scheme, we additionally train our velocity field without the deformation field. Particularly, we use Algorithm 2 to directly transport canonical Gaussians from $t = 0$ to target time t . Since this requires autoregressive integration of all Gaussians by several steps, the memory cost accumulates rapidly, so we choose $\Delta t = \frac{t}{2}$ to control the memory cost same as our main setting.
- (6) **Removing the physics code from deformation field f_{deform} :** Instead of feeding the physics code into the deformation field, we compare with the deformation

Table 4.6: Complete ablation study results on both Dynamic Object Dataset and Dynamic Indoor Scene Dataset

					Dynamic Object Dataset					
					Interpolation			Extrapolation		
	Code $\mathbf{z}$	f_{deform}	$\mathbf{v}(\mathbf{p}_t, t)$	K	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
(1)	learnable	full	full	16	38.722	<u>0.995</u>	<u>0.005</u>	25.961	0.975	0.025
(2)	field	full	w/o $\mathcal{B}(\mathbf{p}_t)$	16	39.126	<u>0.995</u>	<u>0.005</u>	29.400	0.986	0.010
(3)	field	full	w/o decomp	16	39.111	<u>0.995</u>	<u>0.005</u>	29.432	0.985	<u>0.009</u>
(4)	field	full	full	8	<u>39.324</u>	0.996	0.004	30.972	<u>0.989</u>	<u>0.009</u>
(4)	field	full	full	32	39.318	0.996	0.004	31.438	0.990	0.007
(5)	field	$\mathbf{x}$	full	16	20.974	0.945	0.068	17.927	0.922	0.088
(6)	field	w/o $\mathbf{z}$	full	16	39.151	<u>0.995</u>	<u>0.005</u>	31.217	0.988	<u>0.009</u>
(7)	field	w/o $\delta \mathbf{s}$	full	16	39.191	<u>0.995</u>	<u>0.005</u>	<u>31.704</u>	0.990	0.007
FreeGave	field	full	full	16	39.393	<u>0.995</u>	<u>0.005</u>	31.987	0.990	0.007
					Dynamic Indoor Scene Dataset					
					Interpolation			Extrapolation		
	Code $\mathbf{z}$	f_{deform}	$\mathbf{v}(\mathbf{p}_t, t)$	K	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
(1)	learnable	full	full	16	32.343	0.930	0.091	30.444	0.933	0.087
(2)	field	full	w/o $\mathcal{B}(\mathbf{p}_t)$	16	31.471	0.921	0.108	32.316	0.944	0.077
(3)	field	full	w/o decomp	16	31.707	0.921	0.106	31.204	0.943	0.071
(4)	field	full	full	8	32.005	<u>0.929</u>	0.093	34.159	0.962	0.053
(4)	field	full	full	16	31.996	<u>0.929</u>	<u>0.092</u>	<u>34.716</u>	<u>0.965</u>	0.051
(5)	field	$\mathbf{x}$	full	16	-	-	-	-	-	-
(6)	field	w/o $\mathbf{z}$	full	16	31.603	0.921	0.107	33.408	0.955	0.067
(7)	field	w/o $\delta \mathbf{s}$	full	16	32.094	<u>0.929</u>	<u>0.092</u>	34.504	0.964	<u>0.052</u>
FreeGave	field	full	full	32	<u>32.287</u>	0.930	<u>0.092</u>	35.019	0.966	0.051

field same as DefGS [137], *i.e.*, $(\delta \mathbf{p}, \delta \mathbf{r}, \delta \mathbf{s}) = f_{deform}(\mathbf{p}_0, t)$.

(7) Removing the scaling deformation from deformation field: We remove the scaling deformation from the output of f_{deform} , forcing the motion of each Gaussian to be volume-preserving all time, *i.e.*, $(\delta \mathbf{p}, \delta \mathbf{r}) = f_{deform}(\mathbf{p}_0, t, \mathbf{z})$. Gaussian scales are not updated all time.

Results & Analysis: From Table 4.6, we can see that: 1) Removing deformation field is the most influential to the performance. Since autoregressive integration is needed in this case, in order to fit the memory cost, we have to use a large integration time interval, as large as half of the target time. This incurs instability and noise

in training, making the network totally unable to converge. The noise also leads to large gradients for all Gaussian kernels, resulting in unreasonable densification and ultimately making the network totally collapse on Dynamic Indoor Scene Dataset. 2) Replacing the physics code network by assigning a learnable physics code to each Gaussian kernel shows the next greatest impact on performance, demonstrating that the local continuity property of an MLP is helpful for network converging. 3) Either removing divergence-free parametrization or removing motion pattern decomposition diminishes the performance, due to the drastically increase in the degree of freedom and complexity of velocity field. 4) Feeding physics codes as input to the deformation field can enhance the performance. 5) Too few motion patterns decomposition hurts the performance, while over-parametrization is not sensitive.

4.4 Conclusion

Our method extends 3D Gaussian splatting to simultaneously learn 3D scene geometry, appearance, and the critical physics information just from multi-view videos, without needing any object priors. By introducing a physics code and a divergence-free velocity field for each Gaussian kernel, our method can effectively learn precise physical motion patterns without needing the inefficient PINN losses. Extensive experiments on three public datasets and our newly captured challenging real-world dataset demonstrate extraordinary performance of our method for future frame prediction and object motion decomposition. It would be interesting to apply our method in 4D generation where 3D physics learning is crucial to generate plausible motions.

Chapter 5

Translation-Rotation Dynamics System for Physics Learning

5.1 Introduction

Regarding our daily dynamic 3D scenes such as falling balls, rotating fans, and folding chairs, precisely modeling their geometry, appearance, and physical properties, and further enabling controllable prediction of their future states are crucial for emerging applications in robotics, mixed reality, and embodied AI. With the advancement of recent 3D representations such as NeRF [76] and 3DGS [49], a plethora of works [86, 137, 125] have been proposed to model various dynamic 3D scenes, achieving excellent performance in interpolating novel views within the observed time. However, they often fail to extrapolate future frames, essentially because they do not learn the underlying physics priors of 3D scenes.

To learn physics priors, existing methods mainly consist of two categories: 1) physics-informed neural network (PINN) based methods [91] which integrate the governing partial differential equations (PDEs) into loss functions to drive neural networks to learn physically plausible dynamic 3D scenes such as floating smoke [90] and

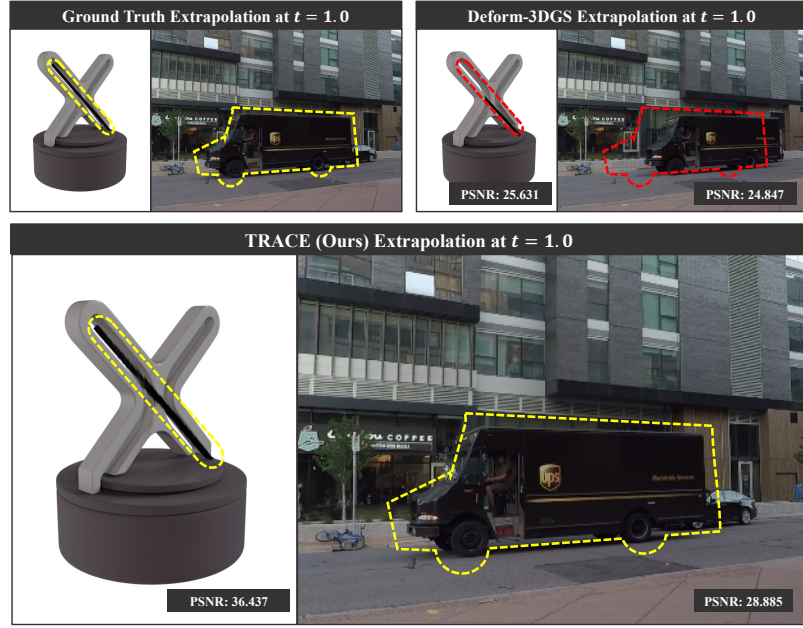


Figure 5.1: Given video frames of a real-world dynamic scene, our TRACE can learn the underlying physics and accurately predict the future motion.

simple moving objects [57]. While demonstrating promising results in modeling 3D geometry and physics such as velocity and viscosity, these methods usually need boundary constraints such as accurate object/foreground masks which may not always be available in practice. In addition, adding PINN losses is not a free lunch, but significantly sacrifices the efficiency in training and accuracy at boundary regions. 2) Simulation based methods [42, 151, 124] which encode various physics systems into neural networks to model elastic objects, fluids, *etc.*. Thanks to explicit physics priors, these methods obtain impressive results in physical properties learning but are often limited to specific object types, lacking generality for complex dynamic scenes.

More recent works like NVFi [53] in Chapter 3 and FreeGave [55] in Chapter 4 have made progress in physics-aware modeling, but they still rely on implicit neural fitting for motion and acceleration patterns, requiring complex neural ODE integration for future inference and failing to enable direct control over physical parameters, such as acceleration (2^{nd} order) or jerk (3^{rd} order).

In this chapter, we aim to introduce a new framework to model dynamic 3D scenes just from multi-view RGB videos, without needing any additional human labels such as object types or masks, ultimately being able to explicitly recover core physical parameters such as accelerations to enable more controllable and efficient future frame prediction. Among various physical properties of a dynamic 3D scene, we focus on learning not just velocity, as NVFi and FreeGave, but the underlying dynamics that generate velocity, as these directly govern the predictability and controllability of the motion. However, to accurately learn such dynamics from RGB videos is extremely challenging, essentially due to the lack of sufficient physics constraints from raw color pixels. This problem is even harder when multiple objects or parts are undergoing rather different motion patterns. For example, regarding two adjacent objects moving in opposite directions in 3D space, the velocity of neighboring 3D surface points at the intersection region tends to have particularly distinct patterns. This means that the latent representation of per-point dynamics in 3D space could be discrete in nature. This discreteness in per-point dynamics suggests that modeling independent particle-level physics is more effective, where each 3D point can be regarded as a rigid particle with its own size, orientation, and unique motion parameters. If its size is zero, the rigid particle degenerates to a point.

With this insight, for each rigid particle in space, we propose to learn an independent dynamics system that includes a complete set of physical parameters to govern its motion over time. According to the laws of classical mechanics, for a specific rigid particle traversing 3D space over time, its motion can always be regarded as a rotational movement about a rotation center which has its own translation. In this regard, we choose to learn a translation rotation dynamics system for each rigid particle, allowing its future motion to be derived accordingly. Alongside learning the core dynamics, we must also model the geometry and appearance of 3D scenes. In this chapter, we naturally choose 3D Gaussian Splatting (3DGS) [49] as the representation, thanks to its unprecedented fidelity in reconstruction and its particle (a Gaussian kernel) based

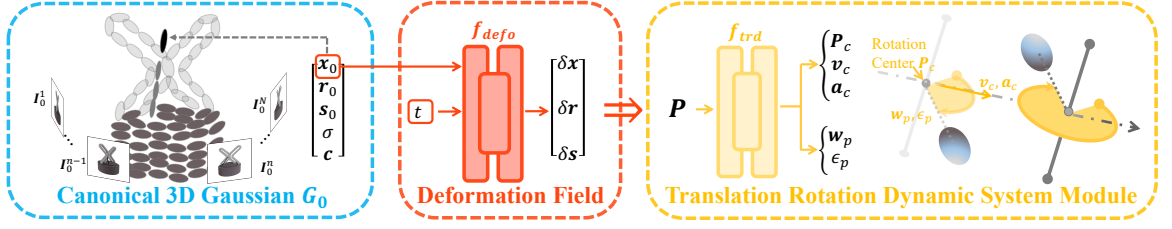


Figure 5.2: The leftmost block shows the canonical 3D Gaussians module, the middle block shows the deformation field, and the rightmost block shows our translation rotation dynamics system module.

representation in nature, which shares the basic concept of our defined rigid particle.

Our framework consists of two major components: 1) a **3D scene representation module** to learn dynamic scene geometry and appearance at a canonical timestamp, which is implemented by a vanilla 3DGS [49], though other variants can be adopted as well; 2) a **translation rotation dynamics system module** to learn a full set of physical parameters for each input rigid particle, which is just realized by multilayer perceptrons (MLPs). Based on these parameters, velocity and higher-order parameters are derived directly via mechanical laws, eliminating the need for additional physics priors such as PINN [91] in training and neural ODE integration in inference.

The key to our framework is the second module which simply regards each 3D Gaussian kernel as a rigid particle and takes it as input into MLPs. Nevertheless, we empirically find that it is hard to optimize this module due to the inaccuracy and instability of Gaussian kernels regressed at early training epochs. To tackle this issue, we simply train an auxiliary deformation field in parallel with our second module using an existing work such as [137] and [125]. Our pipeline is illustrated in Figure 5.2.

Different from current works for modeling dynamic scenes, including NeRF-based methods, *e.g.*, D-NeRF[86]/ TiNeuVox[26]/ HexPlane[10], and 3DGS-based methods such as DefGS [137] and 4DGS [125], our core novelty is the introduced translation rotation dynamics system with an effective optimization strategy, allowing us to

truly learn physical parameters, ultimately achieving future frame extrapolation. By comparison, all those existing methods fail to do so, as verified in Tables 5.1.

By leveraging 3D Gaussians to model scene geometry and appearance, our method, named **TRACE**, learns **t**ranslation **r**otation **d**ynamics systems for estimating accurate physical dynamics. Figure 5.1 shows our qualitative results. Our contributions are:

- We introduce a new framework to model motion physics of complex dynamic 3D scenes, without needing prior knowledge of object shapes, types, or masks.
- We propose to learn a translation rotation dynamics system for each 3D rigid particle, thus allowing direct derivation of velocity and acceleration via mechanical laws rather than implicit neural fitting from additional physics constraints in training.
- We demonstrate superior results in future frame extrapolation on three existing datasets, and one newly collected synthetic dataset with challenging dynamics.

5.2 TRACE

Our framework mainly comprises two modules with an auxiliary deformation field to model 3D geometry, appearance, and physics. Given dynamic multi-view RGB videos with known camera poses and intrinsics, the 3D scene representation module aims to learn a set of 3D Gaussians to model the geometry and appearance in a canonical space. The auxiliary deformation field is designed to predict the translation and distortion of each Gaussian given the current training time t . For these two components, we simply follow existing works [49, 137] which are briefly elaborated in Section 5.2.1. Notably, the deformation field alone cannot extrapolate frames beyond the training time due to the lack of physics. Our core module of the translation rotation dynamics system aims to learn a set of physical parameters for each 3D rigid particle, governing its motion dynamics evolution over time, which is detailed in Section 5.2.2.

5.2.1 Preliminary

For the input multi-view RGB videos, T represents the greatest timestamp in training and N the total number of cameras. For training stability, we first use all frames $\{\mathbf{I}_0^1 \cdots \mathbf{I}_0^n \cdots \mathbf{I}_0^N\}$ at time $t = 0$ to train a reasonable static 3DGS model as an initialization of the 3D scene geometry and appearance, and then use the remaining frames to jointly optimize our translation and rotation dynamics system and the auxiliary deformation field.

Canonical 3D Gaussians: Following the vanilla 3DGS [49], we employ a set of learnable 3D Gaussian kernels $\mathbf{G}_0$ to represent the canonical scene geometry and appearance at $t = 0$. Each kernel is parameterized by a 3D position $\mathbf{x}_0$, covariance matrix obtained from quaternion $\mathbf{r}_0$, scaling $\mathbf{s}_0$, opacity σ , and color $\mathbf{c}$ computed from spherical harmonics. Following prior works [137, 125], we assume the opacity σ and color $\mathbf{c}$ of each Gaussian will not be updated, but constantly associated with the kernel and transported over time.

Given the N images at timestamp $t = 0$, we either initialize all canonical 3D Gaussian kernels $\mathbf{G}_0$ randomly or based on sparse points created by SfM [96]. To train all kernels, we exactly follow 3DGS [49] by 1) projecting Gaussian kernels into camera space, 2) rendering the projected kernels into image space, and 3) optimizing all kernel parameters via ℓ_1 and ℓ_{sim} losses used in 3DGS as follows.

$$\underbrace{\{\cdot \cdot (\mathbf{x}_0, \mathbf{r}_0, \mathbf{s}_0, \sigma, \mathbf{c}) \cdot \cdot\}}_{\mathbf{G}_0} \xrightleftharpoons[\ell_1 + \ell_{sim}]{\text{project+render}} \{\mathbf{I}_0^1 \cdot \cdot \mathbf{I}_0^N\} \quad (5.1)$$

Auxiliary Deformation Field: To aid the learning of our translation and rotation dynamics system, we leverage an existing deformation field [137], but we are also amenable to other deformable Gaussian methods such as 4DGS [125], as demonstrated in our experiments in Section 5.3.1. In particular, the 3D position $\mathbf{x}_0$ of each canonical Gaussian kernel and the current timestamp t are fed into an MLP-based deformation network, denoted as f_{defo} , directly predicting the corresponding position displacement

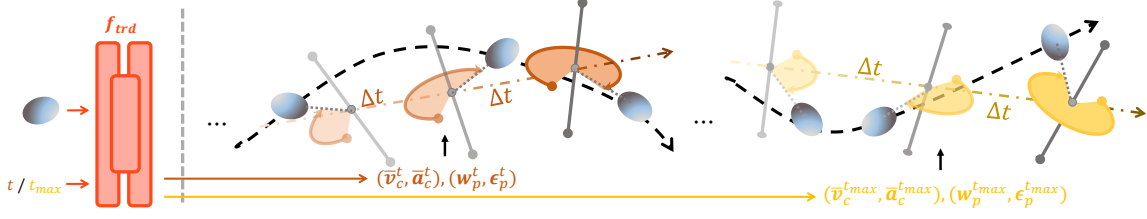


Figure 5.3: The proposed translation rotation dynamics system for a specific rigid particle. The rigid particle will be driven by its learned physical parameters over time, forming a trajectory in 3D space.

$\delta \mathbf{x}$, and the change of quaternion $\delta \mathbf{r}$ and scaling $\delta \mathbf{s}$ from timestamp 0 to t . All Gaussians $\mathbf{G}_t$ at time t can be easily computed as follows, where the operation $\circ$ follows [137], which represents the quaternion multiplication.

$$\mathbf{x}_t = \mathbf{x}_0 + \delta \mathbf{x}, \mathbf{r}_t = \mathbf{r}_0 \circ \delta \mathbf{r}, \mathbf{s}_t = \mathbf{s}_0 \cdot \delta \mathbf{s}, \sigma, \mathbf{c} \quad (5.2)$$

All these deformed Gaussians will be projected and optimized by visual images at timestamp t in a later stage, as clarified in Section 5.2.3, where the deformation net f_{defo} will be optimized from scratch.

5.2.2 Translation Rotation Dynamics System

This module aims to learn physical parameters that govern the motion of 3D scenes. However, the dynamics of an entire space are extremely complex. Here, we simplify this problem and formulate it into just learning per rigid particle dynamics, where we treat each (canonical or deformed) Gaussian kernel as a rigid particle with size and orientation. According to the laws of classical mechanics, for a specific rigid particle $\mathbf{P} \in \mathcal{R}^3$, its motion in a 3D world coordinate system can be regarded as a rotational movement about a rotation center which has its own translation. To this end, we aim to learn the following two groups of physical parameters for each 3D rigid particle $\mathbf{P}$:

- Group #1 - Rotation Center Parameters including: 1) the center's position $\mathbf{P}_c \in \mathcal{R}^3$,

- 2) the center's velocity $\mathbf{v}_c \in \mathcal{R}^3$, and 3) acceleration $\mathbf{a}_c \in \mathcal{R}^3$ in world coordinate.
- Group #2 - Rigid Particle Rotational Parameters including: 1) the rigid particle's rotation vector $\mathbf{w}_p \in \mathcal{R}^3$ with regard to its center $\mathbf{P}_c$, and 2) the rigid particle's angular acceleration $\boldsymbol{\epsilon}_p \in \mathcal{R}^3$.

As illustrated in Figure 5.3, our translation rotation dynamics system module, denoted as f_{trd} , takes a rigid particle $\mathbf{P}$ and a time t as input, directly predicting the physical parameters of its rotation center and its own rotational information. Then, this rigid particle will be driven by its learned physical parameters, forming its motion dynamics, as illustrated by the trajectory in Figure 5.3. The composite velocity for the rigid particle $\mathbf{P}$ at time t is derived as follows.

$$\mathbf{v}_p^t = \mathbf{w}_p^t \times (\mathbf{P} - \mathbf{P}_c^t) + \mathbf{v}_c^t = \mathbf{w}_p^t \times \mathbf{P} + (\mathbf{v}_c^t - \mathbf{w}_p^t \times \mathbf{P}_c^t) \quad (5.3)$$

Nevertheless, the rightmost block of above equation shows that the estimation of the center's velocity $\mathbf{v}_c^t$ and center's position $\mathbf{P}_c^t$ is compounded. Thus, instead of separately learning these two parameters, we turn to learn an equivalent (compounded) center velocity $\bar{\mathbf{v}}_c^t$. Similarly, we instead learn an equivalent center acceleration $\bar{\mathbf{a}}_c^t$:

$$\bar{\mathbf{v}}_c^t = \mathbf{v}_c^t - \mathbf{w}_p^t \times \mathbf{P}_c^t, \quad \bar{\mathbf{a}}_c^t = \mathbf{a}_c^t - \boldsymbol{\epsilon}_p^t \times \mathbf{P}_c^t. \quad (5.4)$$

Proof of Parameters Equivalence: By definition, the 1st-order equivalence is naturally obeyed by the definition. To be specific, it can be shown as:

$$\bar{\mathbf{v}}_p^t = \mathbf{w}_p^t \times \mathbf{P} + \bar{\mathbf{v}}_c^t = \mathbf{v}_p^t$$

Now let's prove the 2st-order equivalence:

$$\begin{aligned}
\bar{\mathbf{v}}_p^{t+dt} &= (\mathbf{w}_p^t + dt \boldsymbol{\epsilon}_p^t) \times \mathbf{P} + \bar{\mathbf{v}}_c^t + dt \bar{\mathbf{a}}_c^t \\
&= (\mathbf{w}_p^t + dt \boldsymbol{\epsilon}_p^t) \times \mathbf{P} + \mathbf{v}_c^t - \mathbf{w}_p^t \times \mathbf{P}_c^t + dt (\mathbf{a}_c^t - \boldsymbol{\epsilon}_p^t \times \mathbf{P}_c^t) \\
&= (\mathbf{w}_p^t + dt \boldsymbol{\epsilon}_p^t) \times \mathbf{P} - (\mathbf{w}_p^t + dt \boldsymbol{\epsilon}_p^t) \times \mathbf{P}_c^t + \mathbf{v}_c^t + dt \mathbf{a}_c^t \\
&= (\mathbf{w}_p^t + dt \boldsymbol{\epsilon}_p^t) \times (\mathbf{P} - \mathbf{P}_c^t) + (\mathbf{v}_c^t + dt \mathbf{a}_c^t) \\
&= \mathbf{v}_p^{t+dt}.
\end{aligned}$$

This f_{trd} module is defined as follows, which is implemented by a simple 8×256 MLPs with 8 layers in total and 256 hidden sizes for each layer. In addition, an 8-degree positional embedding is applied onto the input 3D position $\mathbf{x}$, and ReLU is chosen as the activation function.

$$\{(\bar{\mathbf{v}}_c^t, \bar{\mathbf{a}}_c^t), (\mathbf{w}_p^t, \boldsymbol{\epsilon}_p^t)\} = f_{trd}(\mathbf{P}, t) \quad (5.5)$$

Notably, for an input rigid particle $\mathbf{P}$, the elegance of this module f_{trd} is that it only needs to learn this full translation rotation dynamics system at one specific time t , and that particle's future motion will be governed by the learned dynamics system according to the laws of mechanics.

Due to the complex change of direction of rotation vector $\mathbf{w}_p$ driven by the angular acceleration $\boldsymbol{\epsilon}_p$ for a particle, we use Runge-Kutta 2nd-order method to numerically calculate the dynamics to derive the future parameters, as detailed in Algorithm 3.

Instead of using 2nd-order updating scheme, we can also extend to higher orders or reduce to lower orders with regard to future time. Intuitively, a higher order relationship is expected to capture extremely complex dynamics such as a rolling ball suddenly breaking up into pieces due to unknown explosives inside, whereas a much lower order relationship tends to only capture static or constant speed scenes, thus being oversimplified. In this chapter, we opt for the above 2nd-order scheme to update dynamics parameters for two primary reasons:

- In many applications such as robot manipulation, the need for future prediction usually involves a relatively short interval, *e.g.*, in milliseconds. In this case, a 2^{nd} -order update is usually sufficient to achieve decent approximations. Additionally, a simple sliding window based approach can be applied to continuously predict future frames given the newest visual observations from sensors.
- In our daily life, the majority of common physical movements such as rolling balls or moving cars can be generally described by a 2^{nd} -order relationship. In fact, both Newton’s First and Second Law of Motion can be captured. Notably, since the whole 3D scene comprises a large number of rigid particles, each has up to 2^{nd} -order dynamics. Therefore, the compounded dynamics for the entire 3D scene can be rather complex, including various deformations and translations in our daily lives.

Nevertheless, it is still interesting yet non-trivial to learn much higher-order relationships, and we leave it for future exploration.

5.2.3 Training

With our translation rotation dynamics module and the auxiliary deformation field, we now discuss how to train them together, such that physical parameters can be truly learned.

First, for two timestamps t' and t , where t is usually sampled from the training set and $\Delta t = t - t'$ is predefined to be small enough, we can easily obtain Gaussians $G_{t'}$ from the deformation field f_{defo} based on Equation 5.2.

Second, having our translation rotation dynamics module f_{trd} at hand, we naturally regard the transportation of all kernels from t' to t is governed by the corresponding physical parameters estimated by f_{trd} at time t' as defined in Equation 5.5. Then we use the Runge-Kutta 2^{nd} -order (RK2) method to numerically derive the future physical parameters. Thus we can easily compute the kernel’s orientation $\mathbf{r}_t$ from $\mathbf{r}_{t'}$

Algorithm 3 Runge-Kutta 2nd-order (RK2) method

Input: (1) Gaussians at t' : $\{\cdot \cdot (\mathbf{x}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}) \cdot \cdot\}$, and (2) Δt **Output:** Gaussians at $t = (t' + \Delta t)$: $\{\cdot \cdot (\mathbf{x}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}) \cdot \cdot\}$

- $\{(\bar{\mathbf{v}}_c^{t'}, \bar{\mathbf{a}}_c^{t'}), (\mathbf{w}_p^{t'}, \boldsymbol{\epsilon}_p^{t'})\} \leftarrow f_{trd}(\mathbf{x}_{t'}, t')$, (ref to Eq 5.5)
 - $\bar{\mathbf{v}}_c^{mid} \leftarrow \bar{\mathbf{v}}_c^{t'} + \frac{\Delta t}{2} \bar{\mathbf{a}}_c^{t'}$
 - $\mathbf{w}_p^{mid} \leftarrow \mathbf{w}_p^{t'} + \frac{\Delta t}{2} \boldsymbol{\epsilon}_p^{t'}$
 - $\mathbf{x}_t \leftarrow \mathbf{x}_{t'} + \Delta t(\bar{\mathbf{v}}_c^{mid} + \mathbf{w}_p^{mid} \times \mathbf{x}_{t'})$
 - Convert quaternion to rotation matrix: $\mathbf{R}_{t'} \leftarrow \mathbf{r}_{t'}$
 - Calculate the Cross-Product Matrix, denoted by $\mathbf{W}_p \in \mathcal{R}^{3 \times 3}$, for the normalized vector $\mathbf{w}_p^{mid} / \|\mathbf{w}_p^{mid}\|$. Details are provided below.
 - $\Delta \mathbf{R} \leftarrow \mathbf{I} + (\sin \Delta \theta) \mathbf{W}_p + (1 - \cos \Delta \theta) \mathbf{W}_p^2$, where $\Delta \theta = \Delta t \mathbf{w}_p^{mid}$, following Rodrigue's Formula [95]
 - $\mathbf{R}_t \leftarrow (\Delta \mathbf{R}) \mathbf{R}_{t'}$
 - Convert rotation matrix to quaternion: $\mathbf{r}_t \leftarrow \mathbf{R}_t$
 - Assign $\mathbf{s}_{t'}$ to $\mathbf{s}_t$: $\mathbf{s}_t \leftarrow \mathbf{s}_{t'}$, as each Gaussian is rigid with the same scale by definition
- Return:** All Gaussians at t : $\{\cdot \cdot (\mathbf{x}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}) \cdot \cdot\}$
-

and the kernel's position translation $\mathbf{x}_t$ from $\mathbf{x}_{t'}$. The kernel's translation consists of two parts: 1) the equivalent translation of its rotation center, and 2) the equivalent displacement caused by the kernel's rotation with regard to its center. In particular, the RK2 method is defined as:

Third, we render all the above Gaussians kernels at time t to 2D image space following 3DGS, comparing with the training images at time t . All parameters are trained with:

$$(\mathbf{G}_0, f_{defo}, f_{trd}) \leftarrow (\ell_1 + \ell_{ssim}) \quad (5.6)$$

Cross-Product Matrix: Here we first introduce the definition of Cross-Product Matrix. Given two vectors $\mathbf{k} = [k_1, k_2, k_3]^T \in \mathcal{R}^3$ and $\mathbf{v} = [v_1, v_2, v_3] \in \mathcal{R}^3$, one can calculate the cross product of these two vectors $\mathbf{k} \times \mathbf{v}$. Now we define the Cross-Product

Matrix of vector $\mathbf{k}$ as $\mathbf{K} \in \mathcal{R}^{3 \times 3}$, such that

$$\mathbf{K}\mathbf{v} = \mathbf{k} \times \mathbf{v} \quad (5.7)$$

$$= \begin{bmatrix} -k_3v_2 + k_2v_3 \\ k_3v_1 - k_1v_3 \\ -k_2v_1 + k_1v_2 \end{bmatrix} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \mathbf{v}. \quad (5.8)$$

Thereby we can get the closed-form for the Cross-Product Matrix of vector $\mathbf{k} = [k_1, k_2, k_3]^T$ as

$$\mathbf{K} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \quad (5.9)$$

Now we want to derive the cross-product matrix for vector $\mathbf{w}_p^{mid}/\|\mathbf{w}_p^{mid}\|$. We just need to plug the vector components into the above Equation 5.9.

5.3 Experiments

Datasets: Our method is designed to learn meaningful physical information of 3D dynamic scenes, aiming at accurately predicting future motions, instead of just fitting observed video frames. In this regard, the closest work to us is the recent NVFi [53]. Following NVFi, we primarily evaluate our method on its three dynamic datasets: **1) Dynamic Object dataset** consisting of 6 dynamic objects. Each object displays a unique motion pattern belonging to either rigid or deformable movement. **2) Dynamic Indoor Scene dataset** with 4 complex indoor scenes. Each scene has multiple objects undergoing different rigid body motions. **3) NVIDIA Dynamic Scene dataset** [139]. It consists of two chosen real-world dynamic 3D scenes.

Upon a closer look at the above three datasets, we find that their dynamics captured are relatively simple. In our daily life, the majority of objects and scenes have multiple parts undergoing radically different motions over time, showing challenging physical

patterns to learn. To further evaluate the effectiveness of our design, we collect a new synthetic dataset, named **4) Dynamic Multipart dataset**.

Our new synthetic dataset comprises 4 objects. Each has 2 to 5 distinct motion patterns on different object parts. Following [53], for each object, we collect RGBs at 15 different viewing angles, where each viewing angle has 60 frames captured. We reserve the first 46 frames at randomly picked 12 viewing angles as the training split, *i.e.*, 552 frames, while leaving the 46 frames at the remaining 3 viewing angles for testing *interpolation* ability, *i.e.*, 138 frames for novel view synthesis within the training time period, and keeping the last 14 frames at all 15 viewing angles for evaluating future frame *extrapolation*, *i.e.*, 210 frames. Details of the 4 dynamic objects are:

- **Foldingchair:** A folding chair is given. This chair is composed of three parts. The whole motion is unfolding this chair, so all three parts are undergoing different rotating motions.
- **Hypoerbolic Slot:** This is an extremely hard case, where a stick is rotating through a hypoerbolic slot. Note that, only the stick in this hyperbolic shape is dynamic, this introduces more challenges in motion extrapolation.
- **Satellite:** This object is a satellite with two wing doors opening and one main door opening, all rotating in different directions.
- **Stove:** A home stove is given. The motion is mainly closing its top cover plate.

Baselines: We select the following baselines: **1) NVFi** [53]: This is the closest work to us, but differs from us in two folds. First, NVFi relies on PINN losses to learn physics priors, but we directly learn physical parameters. Second, NVFi adopts NeRF as a backbone, being short in 3D scene geometry and appearance modeling, but our method is amenable to and adopts the powerful 3DGS in nature. **2) T-NeRF** [86]. **3) D-NeRF** [86]. **4) TiNeuVox** [26]. The latter four methods are based on NeRF and designed for novel view interpolation. Therefore they are expected to be rather

weak for future frame extrapolation. For a fair and extensive comparison, we also include the following two baselines. **5) DefGS** [137], and **6) 4DGS** [125]. Both very recent deformable 3D Gaussians methods are particularly strong to model dynamic 3D scenes for novel view synthesis using 3DGS as a backbone. **7) DefGS_{nvfi}**. We build this baseline by combining DefGS with the velocity field proposed by NVFi. This baseline has the powerful 3DGS as a backbone as well as the current state-of-the-art NVFi learning strategy. It is trained with exactly the same settings as our method. To demonstrate the flexibility of our framework, we also adopt 4DGS as our auxiliary deformation field, denoted as TRACE_{4dgs}.

Metrics: We report **PSNR/ SSIM/ LPIPS** scores for RGB synthesis in both novel view interpolation and future frame extrapolation.

5.3.1 Main Results of Future Frame Extrapolation

All methods are trained in a scene-specific fashion. As a common practice, the actual time length in all datasets is normalized to be one time unit, where the first ($0 \sim 0.7$) time unit is training time and the remaining is test time. In training and test, we set Δt to be 2 divided by the training set frame rate. To extrapolate future frames, the time t' for our auxiliary deformation field f_{defo} is set as 0.7. The future frames are progressively calculated by Algorithm 3 with the same time interval Δt .

Our primary goal is to extrapolate meaningful future frames as a continuum of the last training observations. In our evaluation, we follow the second step in Section 5.2.3 to extrapolate future frames from the last observed timestamp of training frames. We also compare novel view (past) interpolation with baselines, but this is less important.

Results & Analysis: Table 5.1 and Table 5.2 compare novel view interpolation and future frame extrapolation of all methods on four datasets. Figure 5.4 shows qualitative results on four datasets. We also provide qualitative results of extrapolation for longer time in Figure 5.6. We can see that:

Table 5.1: Quantitative results of all methods for both future frame extrapolation and novel view interpolation on Dynamic Object and Dynamic Indoor Scene Dataset.

	Dynamic Object Dataset						Dynamic Indoor Scene Dataset					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
T-NeRF[86]	13.163	0.709	0.353	13.818	0.739	0.324	24.944	0.742	0.336	22.242	0.700	0.363
D-NeRF[86]	14.158	0.697	0.352	14.660	0.737	0.312	25.380	0.766	0.300	20.791	0.692	0.349
TiNeuVox[26]	27.988	0.960	0.063	19.612	0.940	0.073	29.982	0.864	0.213	21.029	0.770	0.281
T-NeRF _{PINN}	15.286	0.794	0.293	16.189	0.835	0.230	16.250	0.441	0.638	17.290	0.477	0.618
HexPlane _{PINN}	27.042	0.958	0.057	21.419	0.946	0.067	25.215	0.763	0.389	23.091	0.742	0.401
NSFF[59]	-	-	-	-	-	-	29.365	0.829	0.278	24.163	0.795	0.289
NVFi[53]	29.027	0.970	0.039	27.594	0.972	0.036	<u>30.675</u>	0.877	0.211	29.745	0.876	0.204
DefGS[137]	<u>37.865</u>	<u>0.994</u>	<u>0.007</u>	19.849	0.949	0.045	29.926	<u>0.916</u>	<u>0.130</u>	21.380	0.819	0.188
DefGS _{nvfi}	37.316	<u>0.994</u>	0.008	28.749	<u>0.984</u>	<u>0.013</u>	30.170	0.915	0.133	<u>31.096</u>	<u>0.945</u>	<u>0.077</u>
4DGS[125]	37.285	0.986	0.020	20.354	0.950	0.052	29.381	0.889	0.212	21.107	0.793	0.274
FreeGave	39.393	0.995	0.005	31.987	0.990	0.007	32.287	0.930	0.092	35.019	0.966	0.054
TRACE_{Adgs} (Ours)	35.676	0.985	0.021	<u>30.327</u>	0.983	0.019	28.804	0.862	0.242	30.607	0.896	0.209
TRACE (Ours)	39.305	0.995	0.006	31.597	0.987	0.009	32.088	0.929	0.093	34.824	0.965	0.054

Table 5.2: Quantitative results of all methods for both future frame extrapolation and novel view interpolation on NVIDIA Dynamic Scene and Dynamic Multipart Dataset

	NVIDIA Dynamic Scene Dataset						Dynamic Multipart Dataset					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
T-NeRF[86]	23.078	0.684	0.355	21.120	0.707	0.358	9.833	0.567	0.550	10.064	0.576	0.537
D-NeRF[86]	22.827	0.711	0.309	20.633	0.709	0.327	13.279	0.747	0.378	13.344	0.767	0.340
TiNeuVox[26]	28.304	0.868	0.216	24.556	0.863	0.215	29.957	0.966	0.067	20.804	0.923	0.090
T-NeRF _{PINN}	18.443	0.597	0.439	17.975	0.605	0.428	-	-	-	-	-	-
HexPlane _{PINN}	24.971	0.818	0.281	24.473	0.818	0.279	-	-	-	-	-	-
NVFi[53]	<u>27.138</u>	0.844	0.231	<u>28.462</u>	0.876	0.214	27.516	0.960	0.052	25.235	0.955	0.046
DefGS[137]	26.662	<u>0.893</u>	<u>0.127</u>	24.240	0.895	0.140	34.635	0.990	0.019	20.664	0.930	0.067
DefGS _{nvfi}	26.972	0.890	0.128	27.529	<u>0.927</u>	<u>0.102</u>	34.637	0.990	0.018	28.455	0.979	0.017
4DGS[125]	19.411	0.462	0.532	22.510	0.703	0.408	37.021	0.992	<u>0.014</u>	20.564	0.935	0.067
TRACE_{Adgs} (Ours)	18.618	0.432	0.553	22.554	0.721	0.390	<u>35.993</u>	<u>0.991</u>	0.016	<u>32.317</u>	<u>0.985</u>	<u>0.016</u>
TRACE (Ours)	26.861	0.912	0.089	29.341	0.933	0.074	35.768	<u>0.991</u>	0.011	33.481	0.990	0.007

- Compared with NeRF and 3DGS based dynamic scene modeling methods such as T-NeRF/ D-NeRF/ TiNeuVox/ DefGS/ 4DGS, both versions of our method achieve about 10 points higher on PSNR for future frame extrapolation. This means that, without explicitly learning physical information like us, these methods completely fail to predict the future, highlighting the core value of our method. Compared to

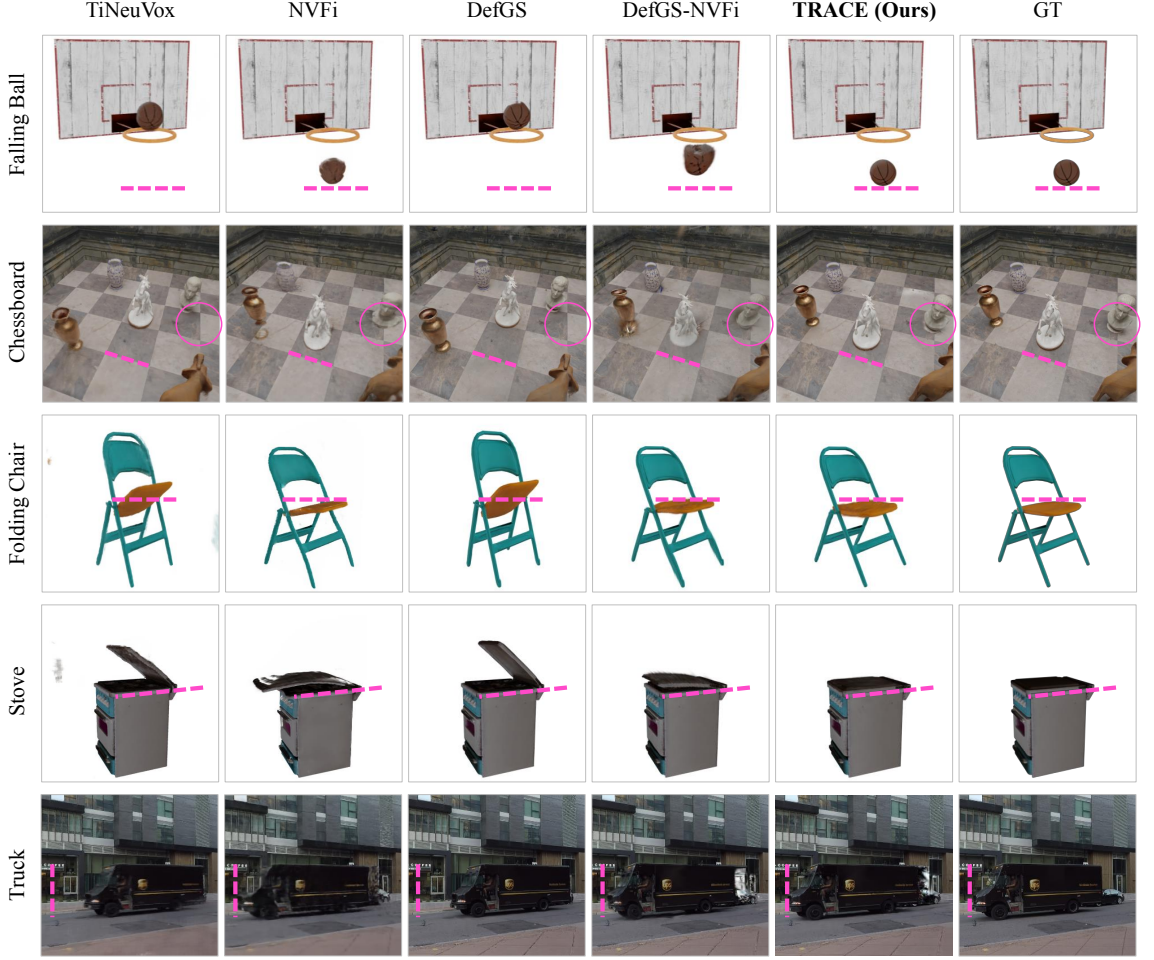


Figure 5.4: Qualitative results of RGB view synthesis for future frame extrapolation on four datasets.

FreeGave, TRACE achieves comparable performance but with explicit parameters.

- Compared with the closest and also strongest baselines NVFi/ DefGS_{nvfi}, our method is still constantly better than them on all datasets for future frame extrapolation. Notably, on the Dynamic Indoor Scene dataset and our newly collected Dynamic Multipart dataset, there are much more complex motion dynamics such as different objects or parts moving in distinct directions, but our best results are constantly about 3 points higher on PSNR than them. Fundamentally, this is because both NVFi and DefGS_{nvfi} rely on PINN losses as soft constraints to

incorporate physics priors, whereas we directly integrate hard physics by learning translation rotation parameters, thus being more effective in learning dynamics.

- Our framework is indeed amenable to existing deformation fields such as DefGS and 4DGS, and both versions achieve superior results for future frame extrapolation on most datasets.
- Lastly, our method can make reasonable future predictions for longer time as shown in Figure 5.6. The motion is generally right and continuous, while some appearing areas which are never observed such as the background in the Truck case may introduce some artifacts.

Analysis on Performance Change along Prediction Time Span We present the extrapolation results for per future timestamp in Figure 5.5. We can see that, though our method is clearly better than baselines, the performance of all methods decreases as the extrapolation goes further. To alleviate this issue, we advocate incorporating new observations in a continual learning manner, which is detailed in Section 5.3.2.

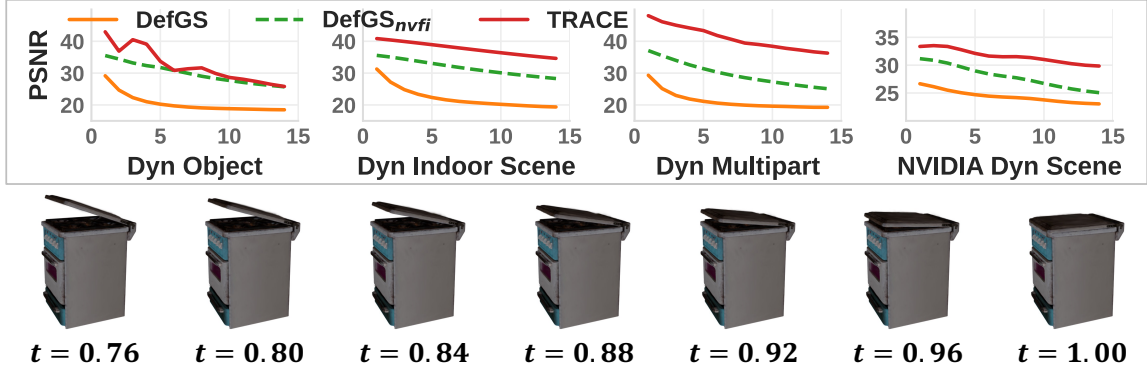


Figure 5.5: The top row shows extrapolation results over future timestamps and the bottom shows qualitative results of our extrapolation.

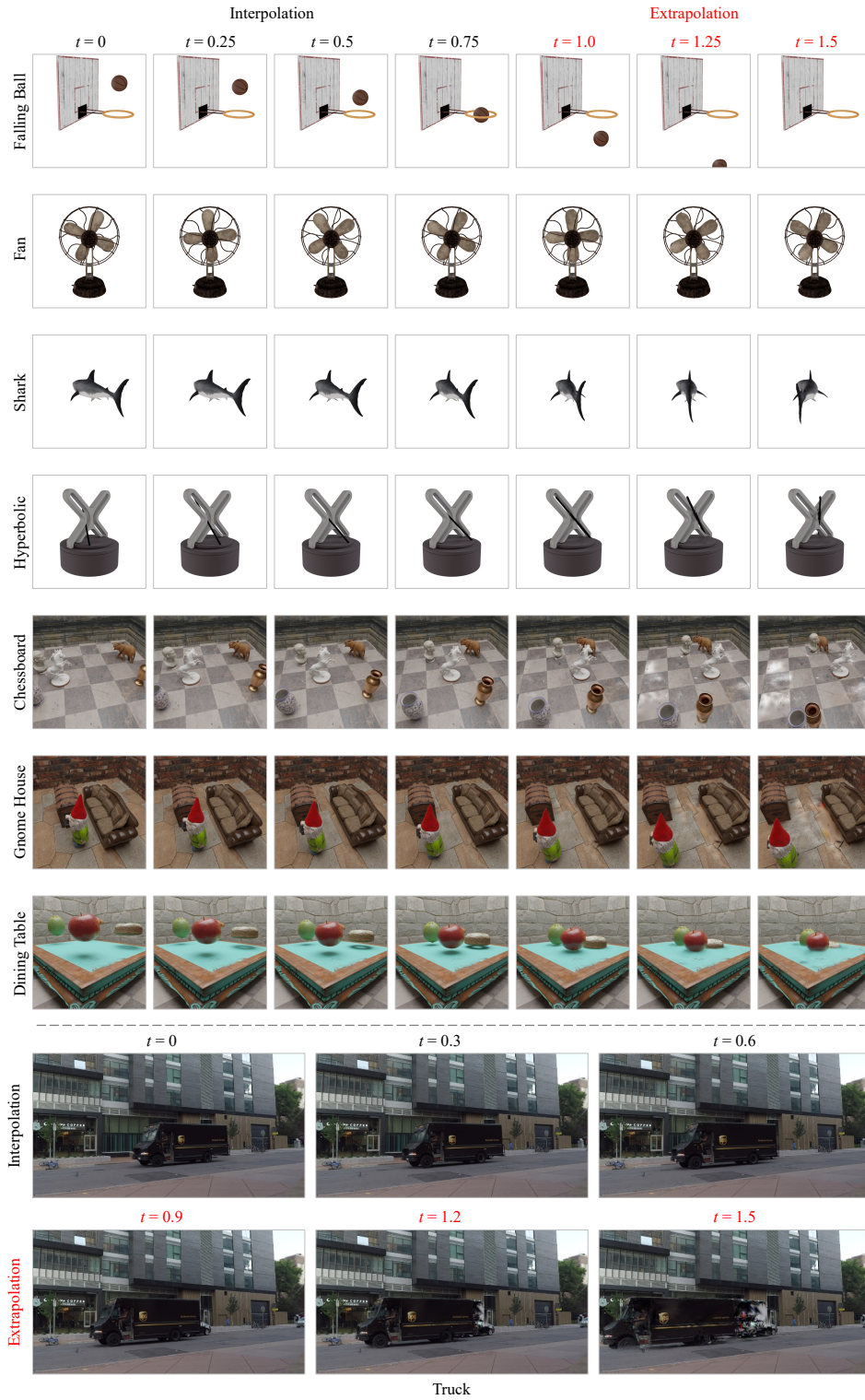


Figure 5.6: Qualitative results of RGB view synthesis for longer extrapolation from our method.

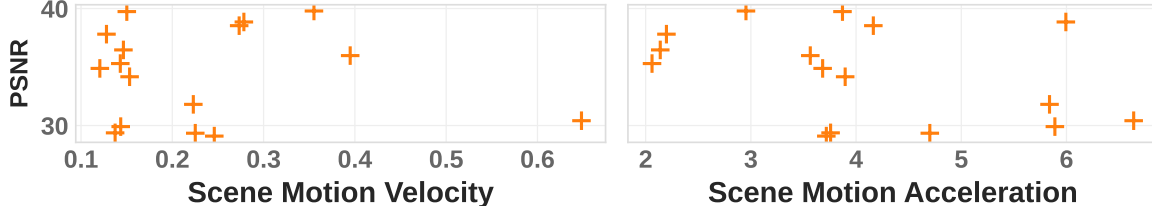


Figure 5.7: Extrapolation over different scene motion complexities.

Analysis on Performance against Motion Complexity For each scene, we calculate its scene motion complexity as follows: 1) we uniformly sample 40K points on each image of the first timestamp; 2) we use BootsTAPIR[23] to track these points over all timestamps and static points are removed; 3) we calculate optical flows for dynamic points between every two adjacent timestamps; 4) the optical flows are divided by the time interval and then normalized by image size, getting normalized image motion velocity which will be averaged out on the whole scene to obtain *Scene Motion Velocity*; 5) we further calculate the difference between two adjacent normalized image motion velocities, getting acceleration which will be averaged out on the whole scene to obtain *Scene Motion Acceleration*. Figure 5.7 shows the relationship where each cross represents one scene out of our 4 datasets. We can see that we achieve similar results over different levels of motion complexity, highlighting our generality.

5.3.2 Continual Learning

We include an additional continual learning experiment to show that our framework can easily adapt to new observations to learn rapidly changing dynamics. We test on **ParticleNeRF dataset** [1]. It has 6 dynamic objects. Each object undergoes various rigid motions from robot manipulation to harmonic oscillation or deformable motions such as cloth. The details for motions and time splits for each scene are as listed:

- **cloth:** This scene includes a piece of square cloth folded by external forces.

- **wheel:** This scene includes a rolling wheel with a constant angular velocity.
- **spring:** This scene includes a box tied onto a spring, and undergoing a harmonic oscillation motion.
- **robot:** This scene shows a robot arm rotating its poses.
- **robot-task:** This scene shows a robot arm putting a box onto a sliding platform.
- **pendulums:** This scene shows two pendulums and each of them undergoes harmonic oscillation motion.

For the first three scenes, we define the 140-th frame as virtual time 1.0, and for the latter three scenes we define the 70-th frame as virtual time 1.0. Particularly, we first feed frames of $(0 \sim 0.15)$ time unit to train the network, and evaluate future frame extrapolation on $(0.15 \sim 0.30)$. Next, we include $(0.15 \sim 0.30)$ to continue training, and then evaluate future frame extrapolation on $(0.30 \sim 0.45)$. We keep adding a time unit of 0.15 till training on $(0 \sim 0.75)$, and extrapolate on $(0.75 \sim 0.9)$.

We compare our extrapolation performance with DefGS and DefGS_{*nvfi*}. Table 5.4 and Table 5.4 show quantitative results. It can be seen that DefGS and DefGS_{*nvfi*} struggle in making right future predictions based on novel observations, while our model can stably adapt to new observations and achieve excellent future frame predictions. This means that even though the 3D scene dynamics are radically and rapidly changing over time, our model can continue capturing those dynamics given new observations. We believe this would be a key enabler for robot perception and manipulation in highly dynamic environments. We show qualitative results in Figure 5.10.

We also conduct experiments on the Panoptic Sports Dataset [44, 99, 43, 71]. It has 6 realistic non-rigid scenes like human interactions, where each scene has 31 viewpoints and each viewpoint has 150 timestamps/frames in total. This dataset is naturally suitable for our continual learning setting and we evaluate our method as

Table 5.3: Quantitative interpolation results (PSNR) of continual learning on ParticleNeRF Dataset.

Observed Time	0.15	0.30	0.45	0.60	0.75	Average
Extrap Till	0.30	0.45	0.60	0.75	0.90	
DefGS[137]	38.665	37.446	36.730	36.229	35.820	36.978
DefGS _{nvfi}	37.984	36.499	35.233	35.168	34.896	35.956
TRACE (Ours)	35.213	34.705	33.164	33.567	33.414	34.013

Table 5.4: Quantitative extrapolation results (PSNR) of continual learning on ParticleNeRF Dataset.

Observed Time	0.15	0.30	0.45	0.60	0.75	Average
Extrap Till	0.30	0.45	0.60	0.75	0.90	
DefGS[137]	19.856	18.652	20.141	20.813	19.131	19.718
DefGS _{nvfi}	26.105	25.987	27.122	26.051	25.640	26.181
TRACE (Ours)	26.731	27.623	27.168	28.422	27.777	27.544

follows: 1) the first 15 timestamps are selected for initial training; 2) for every later round, we load one more timestamp into training set and evaluate future prediction on the future 10 timestamps; 3) we keep training till 135th timestamp. Quantitative results are presented in Figure 5.8 and qualitative results are shown in Figure 5.9. We can see that our method succeeds in extremely challenging non-rigid scenes with sudden motions and non-constant forces. By design, our method is also flexible to advanced hierarchical deformation fields, and Gaussians are traced in a Lagrangian representation, thus mass being preserved in nature.

5.3.3 Analysis of Dynamics Parameters

Our core translation rotation dynamics system module is designed to learn per rigid particle’s physical parameters. Ideally, for particles undergoing the same motion such as all surface points of a single rigid part, they should have the same or similar physical parameters. Given this, multiple dynamic objects or parts with distinct motions can

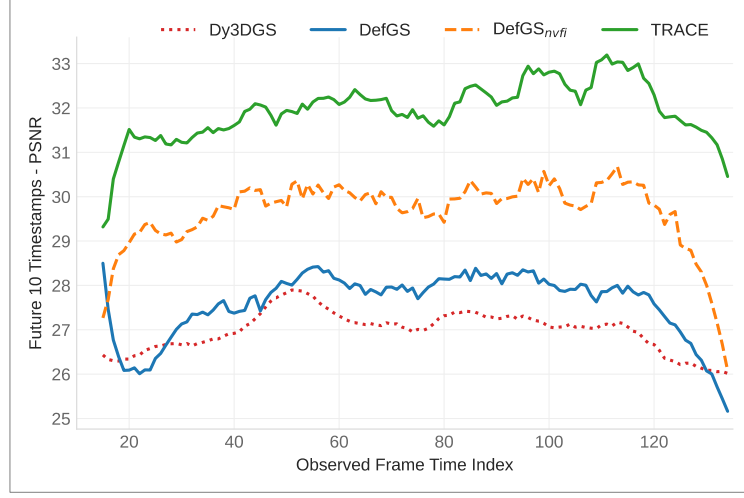


Figure 5.8: Quantitative Extrapolation Results (PSNR) of Continual Learning on Panoptic Sports Dataset.

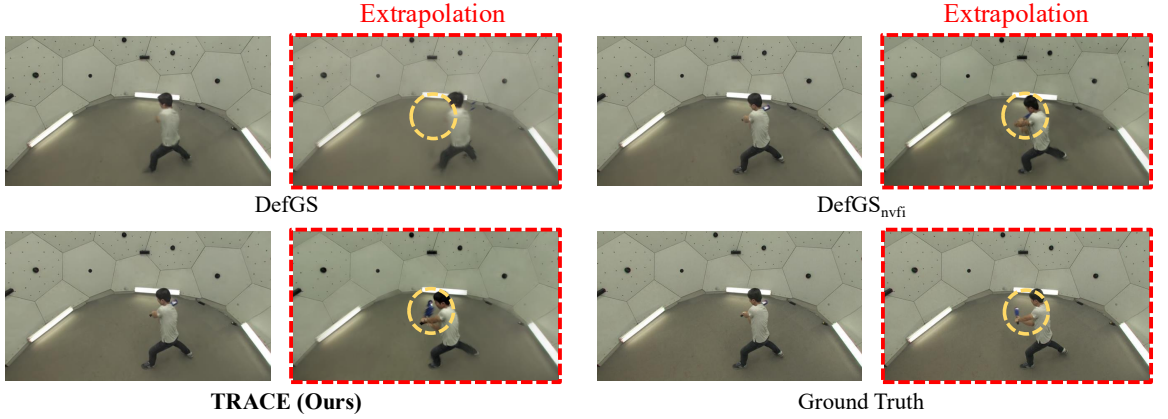


Figure 5.9: Qualitative Results of Continual Learning on Panoptic Sports Dataset.

be automatically segmented based on the similarity of learned physical parameters. By comparison, as shown in Chapter 3.3.2, the prior work NVFi [53] can hardly achieve this autonomous dynamic segmentation by its own design, unless an additional motion grouping method is applied. To further evaluate this nice property of our method, we conduct the following steps to analyze the learned dynamics parameters.

First, after training our method on the Dynamic Object dataset, Dynamic Multipart dataset, and Dynamic Indoor Scene dataset, for each dynamic scene, we have a

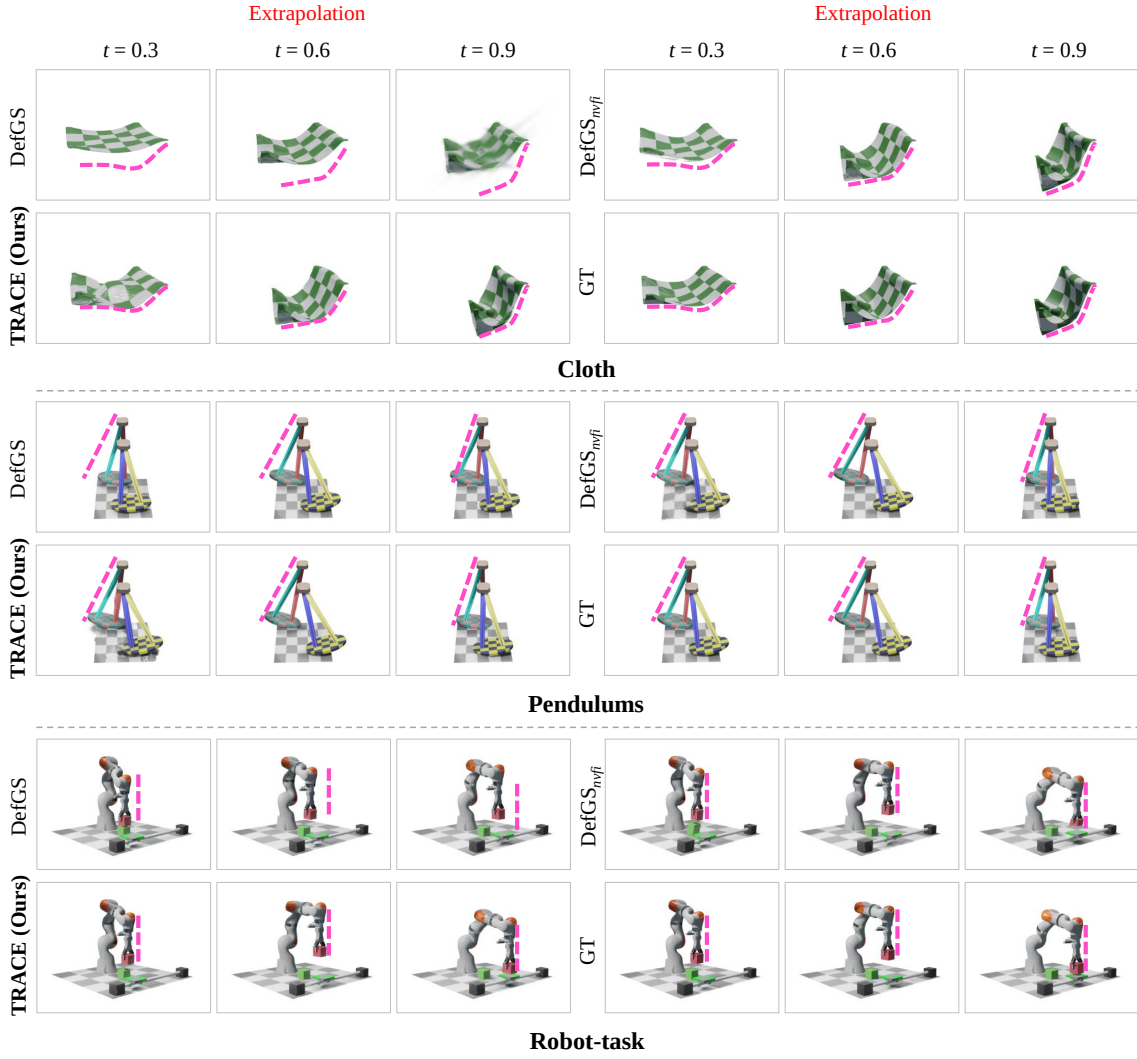


Figure 5.10: Qualitative extrapolation results for continual learning on ParticleNeRF dataset.

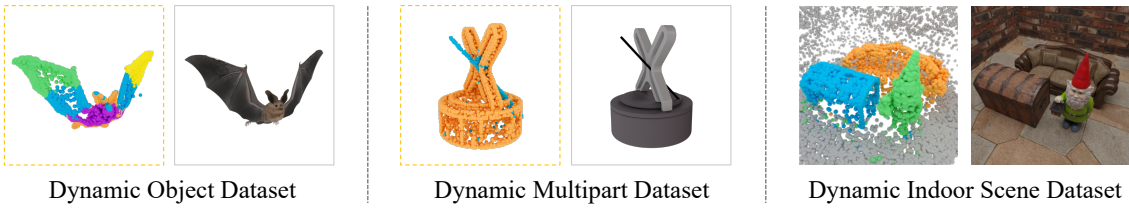


Figure 5.11: Clustering of translation rotation physics parameters.

set of well-trained canonical 3D Gaussians, an auxiliary deformation field, and our translation rotation dynamics parameters.

Then, we use the auxiliary deformation field f_{defo} to deform the canonical 3D Gaussians $\mathbf{G}_0$ to time $t = 0.7$, which is the maximum time the deformation field can query in our training. At this timestamp, the motions of different objects and parts normally achieve a steady state.

Lastly, we query all the learned physical parameters at this time, *i.e.*, $\{(\bar{\mathbf{v}}_c^t, \bar{\mathbf{a}}_c^t), (\mathbf{w}_p^t, \epsilon_p^t)\}$. We choose $(\|\bar{\mathbf{v}}_c^t\|, \bar{\mathbf{v}}_c^t, \|\mathbf{w}_p^t\|, \mathbf{w}_p^t/\|\mathbf{w}_p^t\|)$ as the features to cluster Gaussians via a simple K-means algorithm. As shown in Figure 5.11, all Gaussians can be grouped into physically meaningful objects or parts according to their actual motion patterns. More results are in Figures 5.13 and 5.14

We further quantitatively evaluate our motion grouping results on Dynamic Indoor Scene Dataset. In particular, we follow Gaussian Grouping [138] to render 2D object segmentation masks for all 30 views over 60 timestamps on all 4 scenes, *i.e.*, 7200 images in total. We compare with **D-NeRF**, **NVFi**, **DefGS** and **DefGS_{nvfi}**. We follow NVFi and FreeGave to obtain segmentation results of D-NeRF and NVFi. For the 3DGS-based baselines, we also adopt OGC [102] to segment Gaussians based on scene flows induced from their learned deformation fields. All implementation details can be found in Section 3.3.2 and 4.3.2. We also include a strong 2D object segmentation method, Mask2Former [18] pre-trained by human annotations on COCO [62] as a fully supervised baseline.

As shown in Table 5.5, our method achieves almost perfect object segmentation results on all metrics, significantly outperforming all baselines. This shows that our learned physical parameters correctly model object physical motion patterns and can be used to identify objects according to their motions, without needing any human annotations.

We show qualitative results of the rendered object masks in Figure 5.15.

Motion Segmentation for Real-world Scene: We select the basketball scene from Panoptic Sports Dataset [44, 99, 43, 71], and use manual prompts on Track-Anything [135] to get pseudo ground truth. We render 2D masks on all $31 \times 150 = 4650$ views

Table 5.5: Quantitative results of motion segmentation results on Dynamic Indoor Scene dataset.

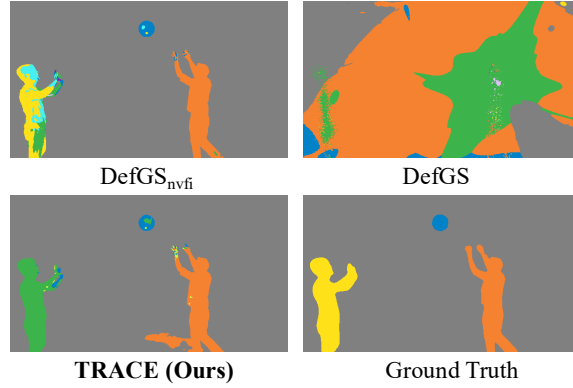
	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑
M2F[18]	65.37	73.14	78.29	<u>94.83</u>	68.88	64.42
D-NeRF[86]	57.26	46.15	59.02	56.55	62.94	46.58
NVFi[53]	<u>91.21</u>	<u>78.74</u>	<u>93.75</u>	93.76	<u>93.74</u>	<u>67.64</u>
DefGS[137]	51.73	57.60	66.43	63.21	70.07	54.46
DefGS _{nvfi}	55.26	62.75	69.83	69.39	72.91	56.82
TRACE (Ours)	95.82	93.28	97.90	96.21	99.86	79.55

for evaluation. Results are shown in Table 5.6 and Figure 5.12.

Table 5.6: Segmentation results on basketball scene of Panoptic Sports Dataset.

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑
DefGS	5.64	15.30	20.51	15.53	30.23	32.05
DefGS _{nvfi}	43.52	53.60	64.02	52.95	80.93	68.25
TRACE (Ours)	53.40	59.46	72.06	64.64	81.40	66.19

Figure 5.12: Qualitative results of segmentation.



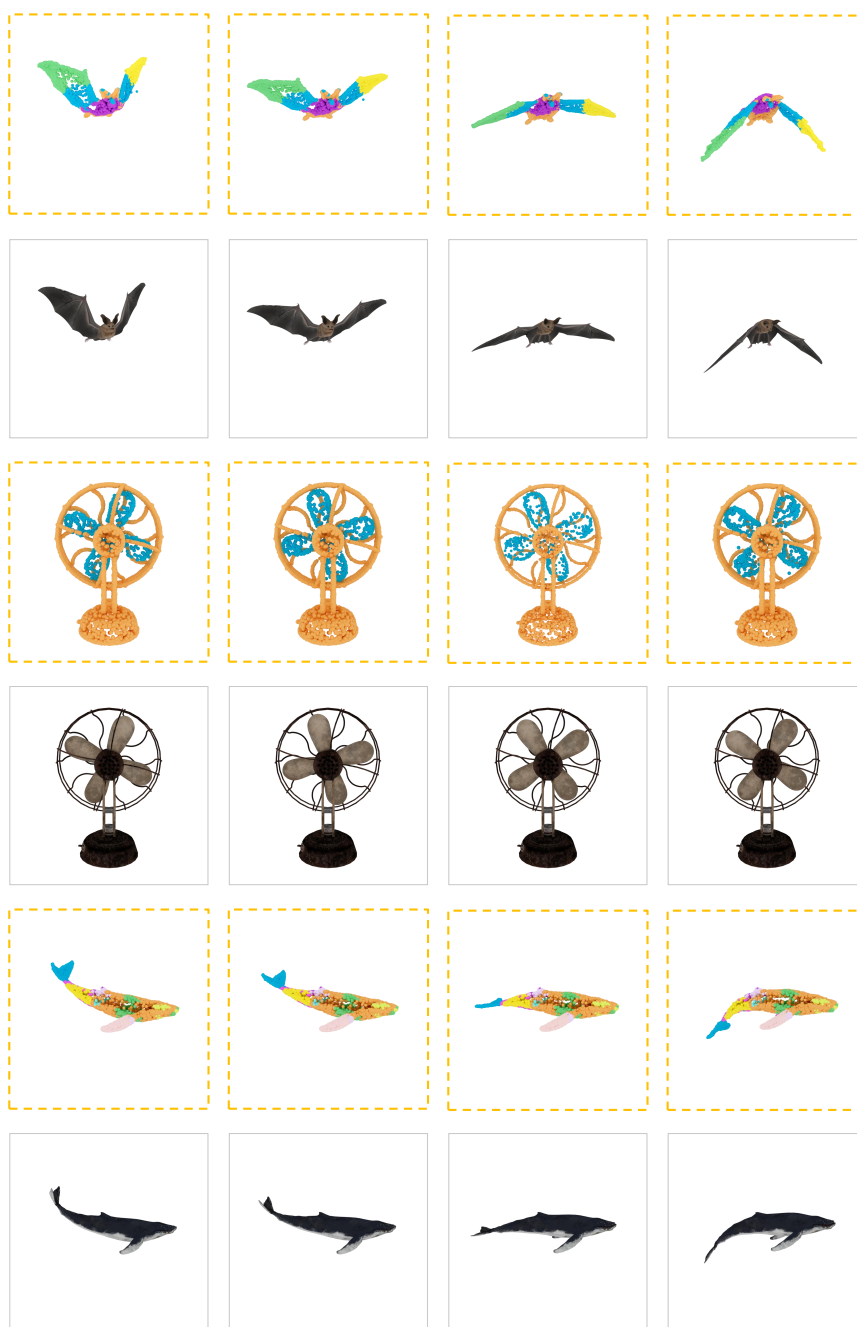


Figure 5.13: Qualitative results for Object/Part Segmentation on Dynamic Object dataset.

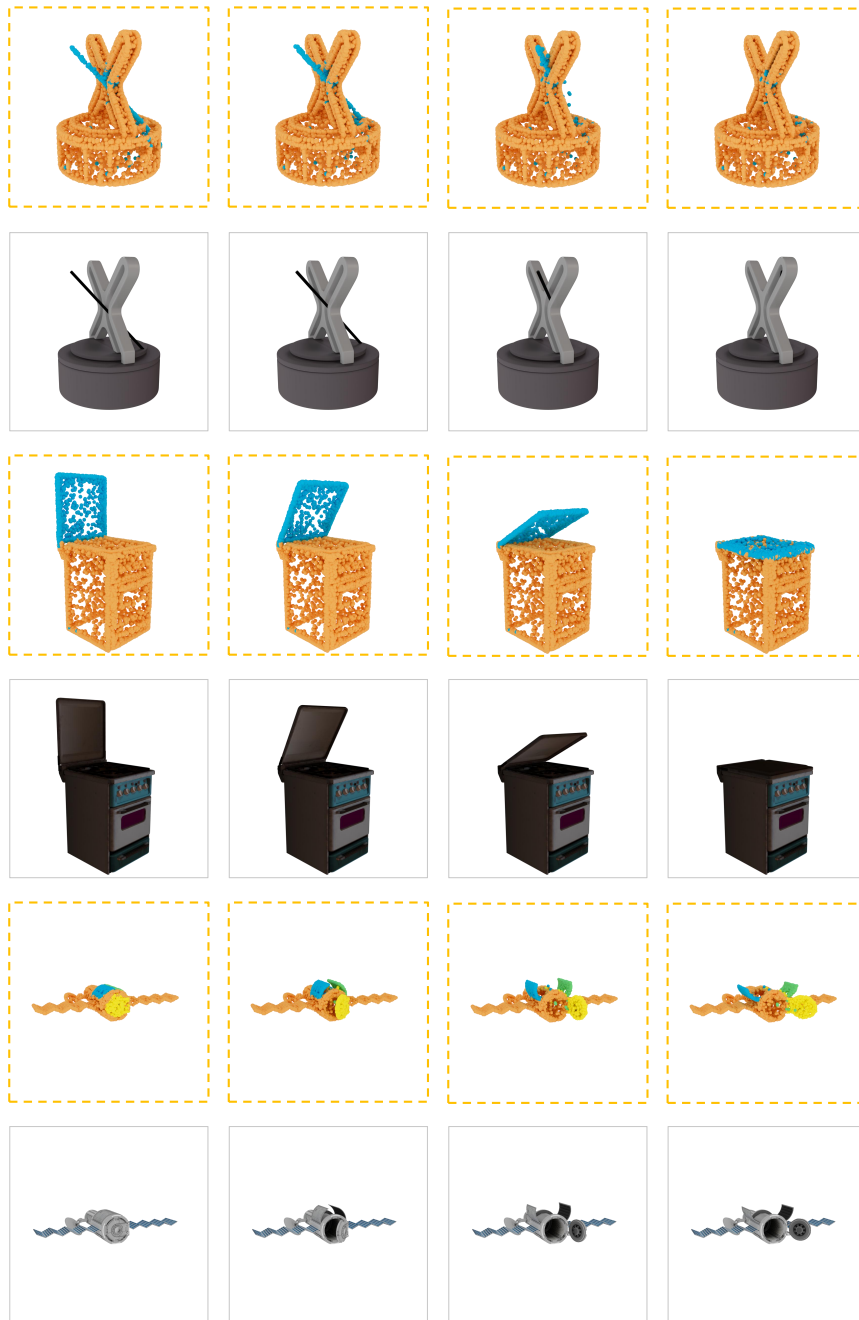


Figure 5.14: Qualitative results for Object/Part Segmentation on Dynamic Multipart dataset.

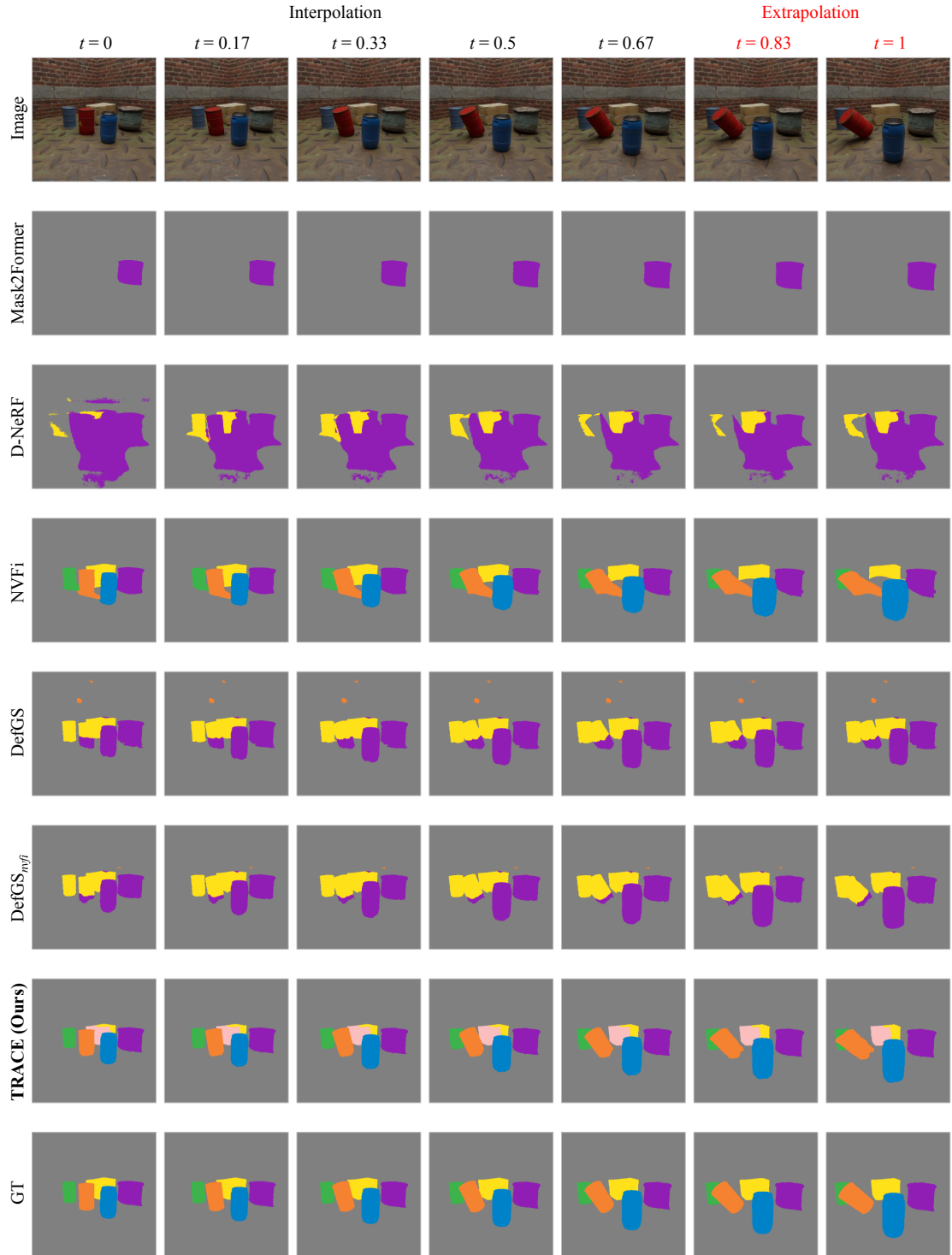


Figure 5.15: Qualitative results for object segmentation on “Factory” of Dynamic Indoor Scene dataset.

5.3.4 Ablation Study

To verify different choices of our method, we conduct the following ablations.

(1) Different choices of time difference Δt in training stage: Given the time interval between two consecutive frames in the training set as δt , we compare three choices of the time difference Δt in the training stage: $\{\delta t, 2\delta t, 3\delta t\}$. We choose $\Delta t = 2\delta t$ in our main experiments.

(2) Different choices of the order of physical dynamics: Our translation rotation dynamic systems choose to learn 2^{nd} -order dynamics in main experiments. We compare the choices for the 1^{st} order (no acceleration) and the 3^{rd} order (acceleration of acceleration).

(3) Removing the auxiliary deformation field f_{defo} : We feed Gaussian s at time $t' = 0$ directly into f_{trd} , and use the output physical parameters to progressively move s to a target future time t with $\Delta t = 2\delta t$.

(4) Removing the physics derivation: Instead of using the physics parameters queried at time t' to derive the future motion states at time $t > t'$, we query the physics parameters from $f_{trd}(\mathbf{P}, t)$ for future timestamps.

(5) Removing the equivalence parametrization: Instead of learning the equivalent velocity and acceleration for the centers as Equation 5.4, we directly learn the original parameters, implemented by a same MLP only with a modified output layer: $\{(\mathbf{P}_c^t, \mathbf{v}_c^t, \mathbf{a}_c^t), (\mathbf{w}_p^t, \epsilon_p^t)\} = f_{trd'}(\mathbf{P}, t')$.

Results & Analysis: Table 5.7 shows all ablation results for future frame extrapolation. To show the robustness of our hyperparameters selected, the first two groups of ablations are extensively conducted on three datasets. The remaining ablations are conducted on our new Dynamic Multipart dataset. It can be seen that: 1) The greatest impact is caused by the removal of the deformation field f_{defo} . While this deformation field itself is unable to learn physics, it significantly aids our core translation rotation

Table 5.7: Ablation studies on three datasets.

Interpolation													
					Dynamic Multipart			Dynamic Object			Dynamic Indoor Scene		
					PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
order	f_{defo}	physcis	equiv										
(1) δt	2	✓	✓	✓	35.370	0.991	0.018	39.305	0.995	0.006	32.184	0.928	0.121
(1) $2\delta t$	2	✓	✓	✓	35.768	0.991	0.011	38.518	0.995	0.005	32.088	0.929	0.093
(1) $3\delta t$	2	✓	✓	✓	35.666	0.991	0.011	37.718	0.994	0.005	32.044	0.929	0.094
(2) $2\delta t$	1	✓	✓	✓	35.369	0.991	0.018	37.859	0.994	0.006	32.250	0.927	0.122
(2) $2\delta t$	3	✓	✓	✓	35.832	0.991	0.011	38.669	0.995	0.004	32.111	0.929	0.093
(3) $2\delta t$	2	✗	✓	✓	18.906	0.895	0.133	-	-	-	-	-	-
(4) $2\delta t$	2	✓	✗	✓	35.526	0.992	0.011	-	-	-	-	-	-
(5) $2\delta t$	2	✓	✓	✗	35.225	0.991	0.011	-	-	-	-	-	-

Extrapolation													
					Dynamic Multipart			Dynamic Object			Dynamic Indoor Scene		
					PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
order	f_{defo}	physcis	equiv										
(1) δt	2	✓	✓	✓	32.852	0.989	0.010	31.597	0.987	0.009	33.831	0.958	0.078
(1) $2\delta t$	2	✓	✓	✓	33.481	0.990	0.007	31.511	0.988	0.006	34.824	0.965	0.054
(1) $3\delta t$	2	✓	✓	✓	33.003	0.989	0.008	29.787	0.983	0.011	34.778	0.965	0.054
(2) $2\delta t$	1	✓	✓	✓	33.125	0.989	0.010	28.522	0.984	0.012	34.576	0.963	0.076
(2) $2\delta t$	3	✓	✓	✓	33.312	0.989	0.008	31.044	0.987	0.007	34.086	0.961	0.056
(3) $2\delta t$	2	✗	✓	✓	19.206	0.907	0.120	-	-	-	-	-	-
(4) $2\delta t$	2	✓	✗	✓	29.602	0.983	0.012	-	-	-	-	-	-
(5) $2\delta t$	2	✓	✓	✗	29.986	0.985	0.012	-	-	-	-	-	-

dynamics system module to learn physical parameters given motion information. 2) The choice of time difference Δt is also important. Once it is as small as the interval between two consecutive frames, the performance drops, because the motion in short intervals is too subtle to be distinguished. For example, a rotation may be learned as a translation. However, if Δt is too large, the appearance fitting could be sacrificed, so the performance is slightly weaker. 3) If physical parameters are learned but not derived, the lack of physics consistency will influence motion learning.

5.4 Conclusion

In this chapter, we have demonstrated that complex motion dynamics—including core physical parameters such as accelerations—can be explicitly learned purely from

multi-view RGB videos without needing additional human labels such as object types and masks. This is achieved by a new generic framework that simultaneously models 3D scene geometry, appearance, and explicit physical parameters by extending the appealing 3DGS technique. In contrast to existing works, including PINN-based methods that rely on soft constraints and even our prior works NVFi and FreeGave that use implicit neural fitting for motion patterns, we directly learn a complete set of physical parameters (encompassing accelerations and rotation/translation properties) to govern the motion of each 3D rigid particle via our core translation-rotation dynamics system module. This explicit parameterization enables not only accurate future frame prediction but also more controllable manipulation of motion dynamics. Moreover, since our framework is purely trained by RGB supervisions, it can be easily extended to other types of priors such as local rigidity to further enhance the performance.

Extensive experiments on three public dynamic datasets and a newly created dynamic multipart dataset have shown the extraordinary performance of our method in the challenging task of future frame extrapolation over all baselines. Notably, our approach achieves this while eliminating the computational overhead of neural ODE integration, thanks to the direct derivation of velocity and acceleration from learned physical parameters. Additionally, the explicitly learned physical parameters facilitate practical applications such as motion segmentation according to the similarity of their dynamics, further demonstrating the utility of our physical parameter learning paradigm.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The ability to model dynamic 3D scenes with physical accuracy and predict their future states is a cornerstone of advancing intelligent systems in robotics, augmented reality, and embodied AI. Over the course of this thesis, we have addressed the critical limitations of existing dynamic 3D modeling methods, specifically their inability to produce physically plausible future predictions, inefficiencies in physics integration, and lack of explicit control over motion dynamics, through a series of progressive innovations. By developing three interconnected frameworks — NVFi, FreeGave, and TRACE — we have established a comprehensive pipeline for learning disentangled geometry, appearance, and physical properties from multi-view videos, enabling accurate, efficient, and controllable future frame extrapolation.

In Chapter 3, we introduce NVFi, the first framework to deeply fuse dynamic radiance fields with general physics rules for various kinds of scenes. By encoding mass and momentum conservation laws into loss functions via PINNs and jointly optimizing the keyframe radiance fields and interframe velocity fields, NVFi learns physically consistent velocity fields, laying the foundation for physics-aware future prediction

without relying on object masks or material annotations. This work validates that integrating fundamental physical priors into dynamic scene modeling is key to enable the model to understand the physics world.

In Chapter 4, we propose FreeGave. Building on NVFi’s progress, we next tackle the inefficiencies of PINN-based methods, which suffer from high computational overhead due to global spatiotemporal sampling and training instability from multi-objective optimization. FreeGave addresses these limitations through two critical innovations: a divergence-free module that ensures mass conservation at every spatial point without global PDE validation, and learnable physics encoding that decomposes complex motions into efficient patterns. By treating 3D Gaussian kernels as rigid particles with latent physics codes, FreeGave maintains physical plausibility while drastically reducing training costs, making physics-aware modeling feasible for large and complex scenes. This work demonstrates that physical constraints can be enforced intrinsically and efficiently, overcoming the scalability barriers of early physics-integrated methods.

In Chapter 5, we develop TRACE, which shifts from implicit neural fitting of motion patterns to explicit learning of underlying physical parameters. By modeling each 3D Gaussian as a rigid particle with independent translation-rotation dynamics systems, TRACE directly learns accelerations for rotation and translation properties, enabling velocity and acceleration to be derived via classical mechanics rather than neural ODE integration. This explicit parameterization eliminates inference overhead, improves extrapolation accuracy, and enables novel applications like motion segmentation based on parameter similarity. TRACE achieves controllable future prediction by grounding dynamics in explicit physical parameters.

6.2 Broader Impact and Future Directions

The research presented in this thesis advances dynamic 3D scene modeling from a descriptive task (interpolating observed motions) to a predictive and controllable one (extrapolating future states with physical rules). By progressively addressing physical plausibility, efficiency, and explicit parameter learning, we have laid the groundwork for practical applications: robots that anticipate environmental changes, augmented reality systems with realistic scene interactions, and content creation tools that generate physically consistent dynamic content. Looking forward, several promising directions are opened up by this thesis.

6.2.1 Physics-Aware Robot Policy Learning

A compelling extension of our research lies in bridging dynamic 3D scene physics modeling with robot policy learning, enabling autonomous systems to interact with the physical world more intelligently. Current robotic systems often rely on predefined motion primitives or short-term trajectory predictions, struggling to adapt to unforeseen dynamic changes, such as a falling object altering its trajectory due to a collision, or a deformable object shifting shape during manipulation. These limitations stem from a lack of explicit understanding of the underlying governing physics, which prevents robots from anticipating long-term consequences of their actions.

By integrating our framework’s ability to perceive and predict physical parameters (*e.g.*, accelerations, rotation dynamics) into robot policy learning, we can enable more robust decision-making. For instance, a household service robot equipped with our physics-aware perception could first reconstruct the 3D dynamics of a cluttered table, predict its future position over the next few seconds, and then plan a collision-free trajectory to grasp a target object without disturbing others. Similarly, industrial collaborative robots could use learned physical parameters of assembly line components

to adjust their gripper force and timing, reducing errors in high-speed operations.

Critical to this direction is the integration of our explicit physical parameter learning (from TRACE) with reinforcement learning or model-predictive control (MPC) frameworks. The learned translation-rotation dynamics and acceleration parameters would serve as a “physics prior” in the robot’s decision loop, constraining policy search to physically plausible actions and reducing the sample complexity of training. This synergy could unlock robots that not only react to the present but proactively plan for the future, marking a significant step toward truly adaptive physical interaction.

6.2.2 Physics-Guided Video Generation

Another impactful application of our research is enhancing video generation models with physical plausibility, addressing a key limitation of current state-of-the-art methods. Modern video generators such as diffusion models excel at producing visually realistic content with coherent textures and short-term motion, but they often fail to respect fundamental physical laws—resulting in artifacts like floating objects, unnatural accelerations, or inconsistent object interactions. These flaws arise because existing models focus on statistical patterns in pixel sequences rather than the underlying physics that govern motion.

Notably, this limitation of physical implausibility cannot be fundamentally addressed by the prevalent scaling law paradigm in generative AI. Approaches driven by the scaling law improve visual quality and coherence primarily by scaling model size, training data volume, or computational resources, essentially enhancing the model’s ability to capture and replicate statistical patterns in existing datasets. In contrast, our work represents an orthogonal direction: instead of refining how well the model leverages existing data, we introduce a novel modal representation for motion that is grounded in physical principles, enabling algorithmic innovation at the source. This shift allows us to move beyond mere statistical mimicry of pixels to explicit modeling

of motion dynamics, directly addressing the root cause of physical artifacts rather than superficially masking them.

Integrating our physics modeling framework into video generation pipelines offers a critical solution to this discrepancy by establishing a supervised pipeline for reconstructing physical information. This pipeline enables the training of generalizable physical features that capture fundamental motion dynamics across diverse dynamic scenes, such as divergence-free velocity fields (from FreeGave) and consistent acceleration patterns (from TRACE). These learned physical features, grounded in explicit physical principles rather than mere statistical pixel patterns, serve as a robust foundation for guiding video generation.

Specifically, this pipeline empowers two key mechanisms for enhancing generative models: first, by injecting or aligning these generalizable physical features into video generators during training, we constrain the model to learn representations that respect physical plausibility alongside visual fidelity. Second, during inference, the pipeline can directly generate structured physical information, including initial velocities, acceleration parameters, that explicitly controls the motion logic of generated content. By anchoring video generation in these learned physical features, whether through feature alignment or direct physical parameter guidance, the model transcends mere visual coherence, ensuring that generated motions adhere to natural physical laws. This approach fundamentally elevates video generation from “looking visually consistent” to “behaving physically realistic”, significantly improving both the authenticity and physical plausibility of synthetic content.

References

- [1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: Particle based encoding for online neural radiance fields. *WACV*, 2024.
- [2] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O’Toole. ToRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis. *NeurIPS*, 2021.
- [3] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. *ECCV*, 2024.
- [4] Daniele Baieri, Stefano Esposito, Filippo Maggioli, and Emanuele Rodolà. Fluid Dynamics Network: Topology-Agnostic 4D Reconstruction via Fluid Dynamics Priors. *arXiv:2303.09871*, 2023.
- [5] Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3D Gaussian Splatting: Survey, Technologies, Challenges, and Opportunities. *arXiv:2407.17418*, 2024.
- [6] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, and Ricardo Martin-brualla Pratul. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *ICCV*, 2021.
- [7] Jonathan T Barron, Keunhong Park, Steven M Seitz, and Ricardo Martin-brualla. Nerfies: Deformable Neural Radiance Fields. *ICCV*, 2021.

-
- [8] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural Surface Reconstruction of Dynamic Scenes with Monocular RGB-D Camera. *NeurIPS*, 2022.
 - [9] Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gaussian-Informed Continuum for Physical Property Identification and Simulation. *NeurIPS*, 2024.
 - [10] Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR*, 2023.
 - [11] Nithin Chalapathi, Yiheng Du, and Aditi S Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *ICLR*, 2024.
 - [12] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensorRF: Tensorial Radiance Fields. *ECCV*, 2022.
 - [13] Guikun Chen and Wenguan Wang. A Survey on 3D Gaussian Splatting. *arXiv:2401.03890*, 2024.
 - [14] Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit Neural Spatial Representations for Time-dependent PDEs. *arXiv:2210.00124*, 2022.
 - [15] Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *NeurIPS*, 2018.
 - [16] Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. SEINE: Short-to-Long Video Diffusion Model for Generative Transition and Prediction. *ICLR*, 2024.
 - [17] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *CVPR*, 2019.

- [18] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. *CVPR*, 2022.
- [19] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. *NeurIPS*, 2020.
- [20] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *ECCV*, 2016.
- [21] Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *TOG*, 2022.
- [22] Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. Learning Vortex Dynamics for Fluid Inference and Prediction. *ICLR*, 2023.
- [23] Carl Doersch, Pauline Luc, Yi Yang, Dilara Gokay, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ignacio Rocco, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped training for tracking-any-point. *Asian Conference on Computer Vision*, 2024.
- [24] Yilun Du, Yanan Zhang, and Joshua B Tenenbaum. Neural Radiance Flow for 4D View Synthesis and Video Processing. *ICCV*, 2021.
- [25] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *CVPR*, 2017.
- [26] Jiemin Fang, Xinggang Wang, and Matthias Nießner. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *SIGGRAPH Asia*, 2022.
- [27] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3D Gaussian Splatting as New Era: A Survey. *TVCG*, 2024.

-
- [28] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *ICLR*, 2016.
- [29] Erik Franz, Barbara Solenthaler, and Nils Thuerey. Learning to Estimate Single-view Volumetric Flow Motions without 3D Supervision. *ICLR*, 2023.
- [30] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. *CVPR*, 2023.
- [31] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4D facial avatar reconstruction. *CVPR*, 2021.
- [32] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic View Synthesis from Dynamic Monocular Video. *ICCV*, 2021.
- [33] Frederic Gibou, David Hyde, and Ron Fedkiw. Sharp interface approaches and deep learning techniques for multiphase flows. *Journal of Computational Physics*, 2019.
- [34] Thibault Groueix, Matthew Fisher, V G Kim, B C Russell, and M Aubry. A Papier-Mache Approach to Learning 3D Surface Generation. *CVPR*, 2018.
- [35] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *ICML*, 2019.
- [36] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications. *arXiv:2211.08064*, 2022.

- [37] Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, Jian Song, and Ze Cheng. Bi-level Physics-Informed Neural Networks for PDE Constrained Optimization Using Broyden’s Hypergradients. *ICLR*, 2023.
- [38] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *arXiv:2204.03458*, 2022.
- [39] Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion Models for Video Prediction and Infilling. *TMLR*, 2022.
- [40] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024.
- [41] Woobin Im, Geonho Cha, Sebin Lee, Jumin Lee, Juhyeong Seon, Dongyoon Wee, and Sung-Eui Yoon. Regularizing Dynamic Radiance Fields with Kinematic Fields. *ECCV*, 2024.
- [42] Alvaro Sanchez-gonzalez Jonathan, Godwin Tobias, Pfaff Rex, Ying Jure, and Peter W Battaglia. Learning to Simulate Complex Physics with Graph Networks. *ICML*, 2020.
- [43] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015.
- [44] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *TPAMI*, 2017.
- [45] Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How Far is Video Generation from World Model: A Physical Law Perspective. *arXiv:2411.02385*, 2024.

-
- [46] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021.
 - [47] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *CVPR*, 2018.
 - [48] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A Compact Dynamic 3D Gaussian Representation for Real-Time Dynamic View Synthesis. *ECCV*, 2024.
 - [49] Bernhard Kerbl, Université Côte, Georgios Kopanas, Université Côte, Thomas Leimkühler, Max-planck-institut Informatik, and G R Aug. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *SIGGRAPH*, 2023.
 - [50] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. DynMF: Neural Motion Factorization for Real-time Dynamic View Synthesis with 3D Gaussian Splatting. *ECCV*, 2024.
 - [51] Isaac Elias Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 1998.
 - [52] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic Gaussian Fusion from Casual Videos via 4D Motion Scaffolds. *arXiv:2405.17421*, 2024.
 - [53] Jinxi Li, Ziyang Song, and Bo Yang. NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. *NeurIPS*, 2023.
 - [54] Jinxi Li, Ziyang Song, and Bo Yang. TRACE: Learning 3D Gaussian Physical Dynamics from Multi-view Videos. *ICCV*, 2025.
 - [55] Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang. FreeGave: 3D Physics Learning from Dynamic Videos by Gaussian Velocity. *CVPR*, 2025.

- [56] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3D Video Synthesis from Multi-view Video. *CVPR*, 2022.
- [57] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. *ICLR*, 2023.
- [58] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *CVPR*, 2024.
- [59] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *CVPR*, 2021.
- [60] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural Dynamic Image-Based Rendering. *CVPR*, 2023.
- [61] Marten Lienen, David Lüdke, Jan Hansen-Palmus, and Stephan Günnemann. From Zero to Turbulence: Generative Modeling for 3D Flow Simulation. *ICLR*, 2024.
- [62] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *ECCV*, 2014.
- [63] Youtian Lin, Zuo Zhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. *CVPR*, 2024.
- [64] Fangfu Liu, Hanyang Wang, Shunyu Yao, and Shengjun Zhang. Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion. *arXiv:2406.04338*, 2024.

-
- [65] Isabella Liu, Hao Su, and Xiaolong Wang. Dynamic Gaussians Mesh: Consistent Mesh Reconstruction from Monocular Videos. *arXiv:2404.12379*, 2024.
- [66] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. PhysGen: Rigid-Body Physics-Grounded Image-to-Video Generation. *ECCV*, 2024.
- [67] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection – a new baseline. In *CVPR*, 2018.
- [68] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. *arXiv:2402.17177*, 2024.
- [69] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust Dynamic Radiance Fields. *CVPR*, 2023.
- [70] Zhicheng Lu, Xiang Guo, Le Hui, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D Geometry-aware Deformable Gaussian Splatting for Dynamic View Synthesis. *CVPR*, 2024.
- [71] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.
- [72] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, Calif., 1967. University of California Press.
- [73] Arjun Mani, Ishaan Preetam Chandratreya, Elliot Creager, Carl Vondrick, and Richard Zemel. SURFSUP: Learning Fluid Simulation for Novel Surfaces. *ICCV*, 2023.

- [74] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *CVPR*, 2019.
- [75] Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. SplatFields: Neural Gaussian Splats for Sparse 3D and 4D Reconstruction. *ECCV*, 2024.
- [76] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *ECCV*, 2020.
- [77] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 2023.
- [78] Haomiao Ni, Changhao Shi, Kai Li, Sharon X. Huang, and Martin Renqiang Min. Conditional Image-to-Video Generation with Latent Flow Diffusion Models. *CVPR*, 2023.
- [79] Atsuhiko Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural Articulated Radiance Field. *ICCV*, 2021.
- [80] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural Scene Graphs for Dynamic Scenes. *CVPR*, 2021.
- [81] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.
- [82] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hyper-NeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *SIGGRAPH Asia*, 2021.

-
- [83] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal Interpolation Is All You Need for Dynamic Neural Radiance Fields. *CVPR*, 2023.
- [84] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. *ICCV*, 2021.
- [85] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. *CVPR*, 2021.
- [86] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *CVPR*, 2021.
- [87] Shenhan Qian, Jiale Xu, Ziwei Liu, Liqian Ma, and Shenghua Gao. Unif: United neural implicit functions for clothed human reconstruction and animation. In *ECCV*, pages 121–137. Springer, 2022.
- [88] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. *CVPR*, 2024.
- [89] Yi-Ling Qiao, Alexander Gao, and Ming C. Lin. NeuPhysics: Editable Neural Geometry and Physics from Monocular Videos. *NeurIPS*, 2022.
- [90] Jiaxiong Qiu, Ruihong Cen, Zhong Li, Han Yan, Ming-ming Cheng, and Bo Ren. NeuSmoke: Efficient Smoke Reconstruction and View Synthesis with Neural Transportation Fields. *SIGGRAPH Asia*, 2024.
- [91] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems

- involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [92] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 2020.
- [93] Amit Raj, Julian Tanke, James Hays, Minh Vo, Carsten Stoll, and Christoph Lassner. ANR: Articulated Neural Rendering for Virtual Avatars. *CVPR*, 2021.
- [94] Siyu Ren, Junhui Hou, Xiaodong Chen, Hongkai Xiong, and Wenping Wang. Ddm: A metric for comparing 3d shapes using directional distance fields. *TPAMI*, 2025.
- [95] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d’un système solide dans l’espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire. *Journal de Mathématiques Pures et Appliquées*, 1840.
- [96] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. *CVPR*, 2016.
- [97] Richard Shaw, Jifei Song, Arthur Moreau, Michal Nazarczuk, Sibi Catley-Chandar, Helisa Dharmo, and Eduardo Perez-Pellitero. SWAGS: Sampling Windows Adaptively for Dynamic 3D Gaussian Splatting. *ECCV*, 2024.
- [98] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv:2004.01806*, 2020.
- [99] Tomas Simon, Hanbyul Joo, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. *CVPR*, 2017.

-
- [100] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.
- [101] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *NeurIPS*, 2020.
- [102] Ziyang Song and Bo Yang. OGC: Unsupervised 3D Object Segmentation from Rigid Dynamics of Point Clouds. *NeurIPS*, 2022.
- [103] Ziyang Song and Bo Yang. Unsupervised 3D Object Segmentation of Point Clouds by Geometry Consistency. *TPAMI*, 2024.
- [104] Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose. *NeurIPS*, 2021.
- [105] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Nithin Raghavan, and Jonathan T Barron. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS*, 2020.
- [106] Yingzhi Tang, Qijian Zhang, Junhui Hou, and Yebin Liu. Human as points: Explicit point-based 3d human reconstruction from single-view rgb images. *TPAMI*, 47:5884–5900, 2023.
- [107] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *ICCV*, 2017.
- [108] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner,

- Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in Neural Rendering. *SIGGRAPH*, 2021.
- [109] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos. *ICCV*, 2023.
- [110] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene from Monocular Video. *ICCV*, 2021.
- [111] Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Representation and Rendering. *ICCV*, 2021.
- [112] Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. *NeurIPS*, 2020.
- [113] Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation. *NeurIPS*, 2022.
- [114] Bing Wang, Lu Chen, and Bo Yang. DM-NeRF: 3D Scene Geometry Decomposition and Manipulation from 2D Images. *ICLR*, 2023.
- [115] Bing Wang, Zhengdi Yu, Bo Yang, Jie Qin, Toby Breckon, Ling Shao, Niki Trigoni, and Andrew Markham. RangeUDF: Semantic Surface Reconstruction from 3D Point Clouds. *arXiv:2204.09138*, 2022.
- [116] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural Trajectory Fields for Dynamic Novel View Synthesis. *arXiv:2105.05994*, 2021.

-
- [117] Jiepeng Wang, Yuan Liu, Peng Wang, Cheng Lin, Junhui Hou, Xin Li, Taku Komura, and Wenping Wang. Gaussurf: Geometry-guided 3d gaussian splatting for surface reconstruction. *arXiv preprint arXiv:2411.19454*, 2024.
 - [118] Peng-shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *TOG*, 36, 2017.
 - [119] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40), 2021.
 - [120] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing. *CVPR*, 2021.
 - [121] Yiming Wang, Siyu Tang, and Mengyu Chu. Physics-Informed Learning of Characteristic Trajectories for Smoke Reconstruction. *SIGGRAPH*, 2024.
 - [122] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhöfer. Learning Compositional Radiance Fields of Dynamic Human Heads. *CVPR*, 2021.
 - [123] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint Rendering of Moving People from Monocular Video. *CVPR*, 2022.
 - [124] William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3D Particle-based Simulators from RGB-D Videos. *ICLR*, 2024.
 - [125] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *CVPR*, 2024.

- [126] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slot-Former: Unsupervised Visual Dynamics Simulation with Object-Centric Models. *ICLR*, 2023.
- [127] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time Neural Irradiance Fields for Free-Viewpoint Video. *CVPR*, 2021.
- [128] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *CVPR*, 2024.
- [129] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum*, 41, 2022.
- [130] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-NeRF: Neural Radiance Fields for Rendering and Temporal Reconstruction of Humans in Motion. *NeurIPS*, 2021.
- [131] Haotian Xue, Antonio Torralba, Daniel LK Yamins, Joshua B. Tenenbaum, Yunzhu Li, and Hsiao-Yu Tung. 3D-IntPhys: Learning 3D Visual Intuitive Physics for Fluids, Rigid Bodies, and Granular Materials. *NeurIPS*, 2023.
- [132] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3D Object Reconstruction from a Single Depth View. *TPAMI*, 2019.
- [133] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *IJCV*, 128:53–73, 2020.
- [134] Chuanxiang Yang, Yuanfeng Zhou, Guangshun Wei, Long Ma, Junhui Hou, Yuan Liu, and Wenping Wang. Monge-ampere regularization for learning arbitrary shapes from point clouds. *TPAMI*, pages 1–15, 2025.

-
- [135] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos, 2023.
 - [136] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. *ICLR*, 2024.
 - [137] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. *CVPR*, 2024.
 - [138] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024.
 - [139] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera, 2020.
 - [140] Meng You and Junhui Hou. Decoupling Dynamic Monocular Videos for Dynamic View Synthesis. *arXiv:2304.01716*, 2023.
 - [141] Meng You and Junhui Hou. Decoupling dynamic monocular videos for dynamic view synthesis. *TVCG*, 2024.
 - [142] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked Generative Video Transformer. *CVPR*, 2023.
 - [143] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *CVPR*, pages 1029–1037, 2019.

- [144] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering. *CVPR*, 2021.
- [145] Qijian Zhang, Junhui Hou, Yu Qian, Juyong Zhang, and Ying He. Paranet: Deep regular representation for 3d point clouds. *ArXiv*, abs/2012.03028, 2020.
- [146] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation. *ECCV*, 2024.
- [147] Yiqun Zhao, Zibo Zhao, Jing Li, Sixun Dong, and Shenghua Gao. Roomdesigner: Encoding anchor-latents for style-consistent and shape-compatible indoor scene generation. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2024.
- [148] Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsFormer: A Transformer-Based Framework For Physics-Informed Neural Networks. *ICLR*, 2024.
- [149] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. *ICCV*, 2021.
- [150] Yihao Zhi, Shenhan Qian, Xinhao Yan, and Shenghua Gao. Dual-space nerf: Learning animatable avatars and scene lighting in separate spaces. In *3DV*, September 2022.
- [151] Licheng Zhong, Hong-xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. *ECCV*, 2024.
- [152] Chuhan Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. *ICCV*, 2017.