Spatiotemporal World Models for Robot Manipulation

Bardienus Pieter Duisterhof

CMU-RI-TR-26-112

October 2026

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Jeffrey Ichnowski, Chair
Deva Ramanan
Shubham Tulsiani
Abhishek Gupta (University of Washington)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2026 Bardienus Pieter Duisterhof

Keywords: Robot Manipulation, Spatiotemporal Representations, 3D Reconstruction, Generative Models, World-Action Models

Abstract

The promise of robot manipulation is to automate tasks that are too dull, dirty, or dangerous for humans. Realizing this promise requires remarkable resource efficiency: robots must adapt to new tasks with limited data and compute, while meeting stringent industrial performance requirements. Current learning-based systems can succeed in dexterous tasks, but require large amounts of resources to reach industrial standards. A possible explanation for the inefficiency is that frontier models learn from images. Content that is irrelevant to robotics dominates the images. Previous work has improved efficiency through task-specific feature engineering. Can we construct similarly focused representations that remain scalable and broadly applicable? This thesis explores spatiotemporal representations as a possible alternative through spatiotemporal reconstruction, learning generative priors, and world-action models.

The first part of this thesis addresses *spatiotemporal reconstruction*. Robots may benefit from observing past experiences in a precise spatial (3D reconstruction) and temporal (3D point tracks) context. A crucial prerequisite for obtaining these contexts is geometric camera calibration. I propose a method for robustly calibrating wide-angle lenses in a new easy-to-use calibration toolbox, TartanCalib, which researchers use around the world. TartanCalib requires dedicated calibration captures, which are not always available. I propose MASt3R-SfM, which uses a pairwise reconstruction foundation model (MASt3R) to achieve unconstrained structure-from-motion. Using a set of calibrated captures, I then explore neural rendering techniques for multi-view reconstruction. Residual-NeRF uses a known background to reduce ambiguity when reconstructing transparent objects. DeformGS extends to dynamic scenes and estimates dense scene motion in highly deformable environments. Cloth-Splatting reconstructs cloth faster and with greater quality, using a pre-trained cloth dynamics model.

The second part investigates *generative spatiotemporal priors* for robot manipulation. Humans understand viable geometry and motion from partial observations, and robots benefit from a similar prior. I propose RaySt3R, a foundation model that completes objects from a single view. It achieves this by casting object completion as a novel-view depth prediction task. However, acquiring this kind of geometric supervision at scale is expensive. To learn from arbitrary images, I propose a post-training recipe (Modality Forcing) to generate monocular depth maps from a text-to-image (T2I) model. I discover that depth quality scales with T2I pre-training, and contribute a state-of-the-art monocular depth estimator. However, this image-based prior does not yet capture temporal structure. In PointZero, I study 3D point track completion as a pre-training objective for 3D world-action models (WAMs) without requiring robot action labels. I show that PointZero improves performance in action-conditioned dynamics prediction and imitation learning.

The third part considers *spatiotemporal world-action models*. In 3PoinTr, I study 3D point tracks as an alternative to video in world-action models (WAMs). To condition a closed-loop policy on compact task plans, 3PoinTr predicts 3D point tracks instead of video. These plans outperform 2D track plans. However, complementary modalities may improve on 3D point tracks alone. With ModAR, I investigate world-action models that autoregressively predict multiple future modalities before generating actions. I contribute the first controlled study across data, modalities and WAM formulations. I find that learned visual features from DINO (a self-supervised visual encoder), point tracks, and depth are beneficial while predicting future RGB images provides no consistent additional benefit at the evaluated scale.

Together, these works connect spatiotemporal reconstruction, generative priors, and action generation to study how spatiotemporal representations can support scalable robot learning.

Acknowledgments

This PhD thesis was made possible through an incredible group of people. I will look back at this time as some of the most formative years of my life, when I started to understand my place in this world. Although challenging, it has been a time of tremendous personal and professional growth. I came to CMU for my career, but I leave with great gratitude for what this place has taught me about myself and the people I met.

It has been a pleasure to be advised by brilliant advisors. Thank you Sebastian Scherer for your expertise in field robotics and your help and understanding during my advisor transition. Thank you Jeff for introducing me to robot manipulation and optimization and for letting me try many (crazy) ideas. Under your guidance I have found my own research taste and confidence and was able to make a big pivot. You took a chance on me by hiring someone with little manipulation experience. I appreciate your confidence in me. Thank you Deva for all your wise advice and for sharing the excitement for training diffusion models from scratch and uncovering their mysteries. Thank you Guido de Croon at TU Delft for your incredible energy, enthusiasm, and mentorship. You helped me fall in love with science. Thank you Vijay Janapa Reddi at Harvard for your guidance and helping me understand academic impact.

Thanks to lab members and collaborators. In the AirLab, I was lucky to get to know Cherie, Brady, Mateo, Nikhil, Rebecca, Yaoyu, Nayana, Sam, Shibo, Bowen, and Andrew. In the Momentum lab, I was incredibly lucky to work with Adam, who has been an excellent collaborator and friend. Thanks to Yuemin, Arthur, Karthik, Yunchao, Jan, and collaborators beyond CMU. Special thanks to Kaifeng, Mandi, Bowen, Stan, Jenny, Alberta, Holly, Wenlong, and Marcel. These collaborations have materially changed my PhD experience.

I have had the privilege of making life-long friends here in Pittsburgh. Although I cannot name everyone here, thank you for your support Tejus, Pranav, Sam, Connor, Nathan, Alicia, Nari, Mateo, Daphne, Uksang, Peter, Nayana, Ananya, Sarvesh, Rayna, Arka, Rishi, Yorai, Ben, Vivian, Mrinal, Homanga, Chris.

Thank you to my friends back in the Netherlands for their unconditional support from far away. Special thanks to Rano, Michaela, Lisa, Menno, José, Benjamin, Luna, Milton, and Sophie.

Thanks to my family, starting with my dad, Gerard Duisterhof. You have been an inspiration to me from the very beginning. From Legos, you have introduced me to this world and I admire your integrity and service to this world. I tried hard not to work on robot manipulation, but here we are. Thank you to my mother, who took care of us from home and gave us the most carefree childhood imaginable. Thank you to my sister for loving me from across the ocean and inspiring me to do good in the world. Thank you to my grandparents for supporting the academic effort and providing a grounding ‘nuchter’ perspective.

During my PhD, I had the privilege of departing on two 6-month internships that changed my perspective and skills. Thank you to Naver Labs Europe for introducing me to deep learning and the French Alps. Thanks to mentors Jerome, Vincent, Thomas and friends Lojze, Alexandre, Andrea, Lucas, and Wonbong. I also had the pleasure of joining World Labs in San Francisco. Thank you to my friends and mentors Justin, Keunhong, KD, Mo, Nikhil, Anushka. A special thanks to bike squat Nielson, Meghan, Chris, Kelly, and Katja.

Thank you to Lucy Page for giving me the courage to pursue my dreams and to stop and smell the flowers.

Contents

1 Introduction **1**
- 1.1 General and efficient representations . . . 2
- 1.2 Thesis contributions . . . 3
  - 1.2.1 Spatiotemporal reconstruction (Part I) . . . 3
  - 1.2.2 Generative spatiotemporal priors (Part II) . . . 5
  - 1.2.3 Spatiotemporal world-action models (Part III) . . . 6

I Spatiotemporal Reconstruction **7**

2 TartanCalib: Iterative Wide-Angle Lens Calibration using Adaptive SubPixel Refinement of AprilTags **9**
- 2.1 Introduction . . . 10
- 2.2 Related Work . . . 11
  - 2.2.1 Camera Models . . . 12
  - 2.2.2 Calibration Toolboxes . . . 12
  - 2.2.3 Pattern Design and Feature Detection . . . 13
- 2.3 Preliminaries . . . 13
  - 2.3.1 Notation . . . 13
  - 2.3.2 Camera Models . . . 14
- 2.4 Method . . . 14
  - 2.4.1 Undistortion . . . 14
  - 2.4.2 Target Reprojection . . . 14
  - 2.4.3 Corner Filtering . . . 15
  - 2.4.4 Subpixel Refinement . . . 15
- 2.5 Results . . . 16
  - 2.5.1 Evaluation Metrics . . . 16
  - 2.5.2 Experimental Setup . . . 17
  - 2.5.3 Feature Detection . . . 17
  - 2.5.4 Subpixel Refinement . . . 19
  - 2.5.5 Comparison on Human-Annotated Features . . . 20
  - 2.5.6 Reprojection Error . . . 21
  - 2.5.7 Stereo Depth . . . 21

2.6 Conclusion . . . 22

3 MASt3R-SfM: a Fully-Integrated Solution for Unconstrained Structure-from-Motion 25
3.1 Introduction . . . 26
3.2 Related Works . . . 28
3.3 Preliminaries . . . 29
3.4 Proposed Method . . . 30
3.4.1 Scene graph . . . 31
3.4.2 Local reconstruction . . . 32
3.4.3 Coarse alignment . . . 33
3.4.4 Refinement . . . 33
3.5 Experimental Results . . . 34
3.5.1 Experimental setup . . . 35
3.5.2 Comparison with the state of the art . . . 36
3.5.3 Ablations . . . 38
3.6 Conclusion . . . 41

4 Residual-NeRF: Learning Residual NeRFs for Transparent Object Manipulation 43
4.1 Introduction . . . 44
4.2 Related Work . . . 46
4.3 Problem Statement . . . 47
4.4 Method . . . 47
4.4.1 Preliminary: Training NeRF . . . 48
4.4.2 Depth from NeRF . . . 49
4.4.3 Learning a Residual NeRF . . . 49
4.5 Experiments . . . 50
4.5.1 Hyperparameters . . . 50
4.5.2 Baselines . . . 50
4.5.3 Synthetic Blender Data . . . 50
4.5.4 Real World Data . . . 51
4.5.5 Implementation Details . . . 51
4.5.6 Blender Depth Results . . . 52
4.5.7 Real World Depth Results . . . 53
4.5.8 Training Speed . . . 54
4.5.9 Grasp Planning using Dex-Net . . . 54
4.5.10 Residual-NeRF-S Mixnet Ablation . . . 55
4.6 Conclusion and Discussion . . . 56
4.7 Acknowledgement . . . 56

5 DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation **57**
5.1 Introduction . . . 58
5.2 Related Work . . . 60
5.2.1 Neural Rendering for Novel View Synthesis . . . 60
5.2.2 Dynamic Novel View Synthesis . . . 60
5.2.3 Point Tracking . . . 61
5.2.4 Tracking for Robotics . . . 61
5.3 Problem Statement . . . 62
5.4 Preliminary . . . 62
5.4.1 Gaussian Splatting . . . 62
5.4.2 Deformation Fields for Dynamic Scenes . . . 63
5.5 Method . . . 63
5.5.1 4D Gaussian Splatting . . . 64
5.5.2 3D Tracking using 4D Gaussians . . . 65
5.5.3 Learning 3D Masks . . . 66
5.6 Experiments . . . 66
5.6.1 Simulation Experiment Setup . . . 67
5.6.2 Simulation Results . . . 68
5.6.3 Real-World Experiment Setup . . . 68
5.6.4 Real-World Experiment Results . . . 69
5.7 Conclusions . . . 71

6 Cloth-Splatting: 3D Cloth State Estimation from RGB Supervision **73**
6.1 Introduction . . . 74
6.2 Related Work . . . 75
6.3 Problem Formulation . . . 76
6.4 Method . . . 77
6.4.1 State Prediction . . . 77
6.4.2 Mesh-constrained Gaussian Splatting . . . 78
6.4.3 State Update . . . 79
6.5 Experimental Evaluation . . . 80
6.5.1 Experimental Set-up . . . 80
6.5.2 Quantitative Results . . . 81
6.5.3 Qualitative Results . . . 83
6.5.4 Robotic Manipulation Use Case . . . 83
6.6 Conclusions . . . 85

II Generative Spatiotemporal Priors **87**

7 RaySt3R: Predicting Novel Depth Maps for Zero-Shot Object Completion **89**
7.1 Introduction . . . 90

7.2 Related work . . . 91
7.3 Problem statement . . . 92
7.4 Methods . . . 93
7.4.1 Network architecture . . . 94
7.4.2 Training objectives . . . 94
7.4.3 View sampling . . . 95
7.4.4 Merging predictions . . . 95
7.5 Results . . . 96
7.5.1 Training dataset . . . 96
7.5.2 Evaluation datasets . . . 96
7.5.3 Data augmentation . . . 97
7.5.4 Implementation details . . . 97
7.5.5 Baselines . . . 97
7.5.6 Quantitative results . . . 98
7.5.7 Qualitative results . . . 98
7.5.8 Ablation studies . . . 99
7.6 Conclusion and future work . . . 101

8 Modality Forcing for Scalable Spatial Generation 103
8.1 Introduction . . . 104
8.2 Related Work . . . 106
8.3 Method . . . 107
8.3.1 Problem Definition . . . 107
8.3.2 Modality Forcing . . . 108
8.3.3 Self-Distillation . . . 110
8.4 Results . . . 110
8.4.1 Scaling experiment. . . . 110
8.4.2 FLUX-Based Modality Forcing . . . 112
8.4.3 Denoising Trajectory Ablations . . . 115
8.5 Conclusion and Limitations . . . 116

9 PointZero: 3D Point Track Completion for Learning Transferable 3D Dynamics 119
9.1 Introduction . . . 120
9.2 Related Work . . . 122
9.3 Problem Statement . . . 123
9.4 Methods . . . 124
9.4.1 Training Objectives . . . 125
9.4.2 Dataset . . . 126
9.4.3 Post-training Recipes . . . 126
9.5 Results . . . 127
9.5.1 Implementation Details . . . 127
9.5.2 Baselines . . . 128

9.5.3 Evaluation Metrics . . . 128
9.5.4 Synthetic Evaluation . . . 129
9.5.5 Real-World Zero-shot Partial Point Track Completion . . . 129
9.5.6 Post-training: Robot-Conditioned 3D Dynamics Prediction . 131
9.5.7 Post-training: Imitation Learning for Robot Manipulation . . 131
9.6 Conclusion . . . 132
9.7 Supplementary Details . . . 133
9.7.1 Training Loss Weighting . . . 133
9.7.2 Dynamics Prediction Evaluation Metrics . . . 133
9.7.3 Dataset Generation Details . . . 134
9.7.4 Additional Manipulation Comparisons . . . 135
9.7.5 Ablations . . . 135
9.7.6 Additional Synthetic Data Point Dynamics Completion Results 136
9.7.7 Real-World Dynamics Prediction Quantitative Results . . . 137
9.7.8 PGND Zero-shot Evaluation . . . 137
9.7.9 Imitation Learning Experimental Setup . . . 139

III Spatiotemporal World-Action Models 141

10 3PoinTr: 3D Point Tracks for Learning Manipulation from Unconstrained Human Videos 143
10.1 Introduction . . . 144
10.2 Related Work . . . 146
10.3 Method . . . 147
10.3.1 Problem Formulation . . . 148
10.3.2 Network architecture and training . . . 148
10.4 Experiments . . . 149
10.4.1 Implementation Details . . . 150
10.4.2 Baselines . . . 151
10.4.3 Results: 3D Point Track Prediction . . . 152
10.4.4 Results: Sample-Efficient Policy Learning . . . 152
10.4.5 Results: Ablations . . . 154
10.5 Conclusion . . . 154
10.6 Limitations . . . 155
10.7 Implementation Details . . . 155
10.8 Visualization of Task Spatial Variation . . . 156
10.9 Distribution Shift Between Human Videos and Robot Demonstrations 157
10.10 Data Collection Details . . . 157
10.11 Additional Point Track Prediction Results . . . 158
10.12 Additional Ablation Results . . . 158
10.13 Future Work . . . 159

11 Modality-Autoregressive World-Action Models **161**
11.1 Introduction . . . 162
11.2 Related work . . . 163
11.2.1 World-action models . . . 163
11.2.2 Multimodal generation . . . 164
11.3 Method . . . 165
11.3.1 Problem definition . . . 165
11.3.2 ModAR: Modality-Autoregressive World Modeling . . . 165
11.4 Experiments . . . 168
11.4.1 Experimental setup . . . 168
11.4.2 Simulation results . . . 169
11.4.3 Ablations . . . 174
11.4.4 Real-world experiments . . . 175
11.4.5 Implementation details . . . 176
11.5 Conclusion . . . 178
11.6 Limitations . . . 178

12 Conclusion **179**
12.1 Summary of contributions . . . 179
12.1.1 Spatiotemporal reconstruction . . . 179
12.1.2 Generative spatiotemporal priors . . . 180
12.1.3 Spatiotemporal world-action models . . . 180
12.2 Revisiting SOLF . . . 181
12.3 Future work . . . 182
12.4 Closing perspective . . . 183

Bibliography **185**

List of Figures

1.1 Generality and efficiency in representation design. This thesis investigates spatiotemporal world models that retain broadly useful physical structure while focusing prediction on information relevant to manipulation. The representation choices and dashed curve are schematic, rather than measured comparisons. 2
1.2 Reconstruction problems studied in Part I: calibrating cameras, recovering camera poses and scene geometry, and reconstructing geometry as it changes over time. 4
1.3 Spatiotemporal generation in Part II: unobserved object shape, depth through joint RGB–depth denoising, and 3D point tracks from sparse conditioning. Blue dashed curves and dotted meshes mark completed geometry and motion. 5
1.4 World–action models (WAMs) in Part III: generating 3D point-track plans to condition a closed-loop policy, and denoising future modalities sequentially to condition robot actions. Point tracks, visual features, and depth in a single WAM. 6

2.1 Comparison of target detection and feature refinement between TartanCalib and Kalibr. (Left) Target detection. Green circles: features newly detected by TartanCalib, orange circles: features previously picked up by Kalibr [245]. (Right) Zoomed-in view of detected features. Green point inside green and orange circles: refined features of TartanCalib, red point inside orange circle: features from Kalibr. TartanCalib detects more features near the image border and the features have better location accuracy. 11
2.2 The pipeline of TartanCalib. The pipeline consists of an iterative calibration procedure with a newly proposed adaptive feature refinement method. . . . 12
2.3 The aerial robot used in the experiment. It has three identical sets of cameras and fisheye lenses. Every camera uses a Lensagon BF5M2023S23C Lens (195° FOV), with a Leopard Imaging LI-IMX264-MIPI-M12 sensor. Red circles: the two fisheye lenses that are used in Section 2.5.7 to form a stereo pair. 18

2.4 AprilTag features detected on the Lensagon BF5M. The results show that especially at the edge of the lens with polar angle approaching 100 degrees, TartanCalib detects significantly more features by reprojecting points from the target frame into the image frame. 19
2.5 This figure shows the measure of symmetry as was proposed in Equation 2.5, on top of an AprilTag corner. The arrows are pointed in the direction of the symmetry gradient to show how an optimizer may behave locally. 20
2.6 This figure shows the results of an experiment for evaluating downstream performance when using pinhole reprojections at the edge of the Lensagon BF5M fisheye lens. We create a stereo pair of virtual pinhole images to show that (1) the images are less distorted using TartanCalib, and (2) the disparity maps generated using the stereo pair are less noisy. This result demonstrates better features detected by TartanCalib help improve downstream tasks. . . 22

3.1 Top: Relative rotation (RRA) and translation (RTA) accuracies on the CO3Dv2 dataset when varying the number of input views with random subsampling (the more views, the greater their overlap). In contrast to our competitors, MASt3R-SfM offers nearly constant performance on the full range, even for very few views. Bottom: MASt3R-SfM also works *without motion*, *i.e.* in purely rotational settings. We show here a reconstruction from 6 views sharing the same optical center. 27
3.2 Overview of the proposed MASt3R-SfM method. Given an unconstrained image collection, possibly small (1 image) or large (> 1000 images), we start by computing a sparse scene graph using efficient image retrieval techniques given a frozen MASt3R's per-image features. We then compute local 3D reconstruction and matches for each edge using again a frozen MASt3R's decoder. Global optimization proceeds with gradient descent of a matching loss in 3D space, followed by refinement in terms of 2D reprojection error. 30
3.3 Factor graph for MASt3R-SfM. Free variables on the top row serve to construct the constrained pointmap χ, which follows the pinhole camera model by design and onto which the loss functions from Equations 3.3 and 3.4 are defined. 33
3.4 Pose accuracy (↑) on T&T-200 w.r.t. the number of iterations of the coarse and refinement stages (resp. ν_1 and ν_2). 41

4.1 Residual-NeRF, a method that leverages mostly static scenes to improve depth perception and speed up training. Residual-NeRF begins by learning a *background NeRF* of the entire scene without transparent objects. Following this, we learn a *residual NeRF* and a *Mixnet* to complement the *background NeRF*. 45

4.2 Residual-NeRF, a method that leverages mostly static scenes to improve depth perception and speed up training. We first learn a *background NeRF* of the scene without transparent objects and leverage it as a scene prior. Following this, we learn a *residual NeRF* and *Mixnet*. The *Mixnet* is an MLP that learns to combine the *background NeRF* and the *residual NeRF*. Equation 4.5 describes how the output of the *Mixnet* MLP is used to combine the two NeRFs. 48
4.3 The scenes used for evaluation. We create nine synthetic Blender scenes with transparent objects and three real-world scenes, divided into scenes A-C with increasing difficulty. 51
4.4 Depth maps for Residual-NeRF and Dex-NeRF evaluated on the synthetic Blender dataset. The results suggest Residual-NeRF improves depth maps with fewer holes and less noise. 53
4.5 Depth maps inferred by Dex-NeRF [127] and Residual-NeRF in the real world. The results suggest Residual-NeRF results in fewer holes and less noise. 53
4.6 RMSE and MAE in depth maps, logged over the number of training epochs and averaged over all synthetic scenes. The results suggest Residual-NeRF greatly improves training speed with respect to depth reconstruction. . . . 54
4.7 Dex-Net evaluated on the output of Dex-NeRF and Residual-NeRF. The results suggest the output from Residual-NeRF leads to more robust grasping. 55
4.8 Naive Merging (Residual-NeRF-S) as introduced in Equation 4.8 compared against Mixnet (Residual-NeRF). The results suggest Mixnet helps view reconstruction and depth mapping when merging NeRFs. 56

5.1 We propose DeformGS, a method that improves state-of-the-art methods for accurate 3D point tracking in highly deformable scenes. This figure shows DeformGS tracking in a real-world scene from the Robo360 [176] dataset. . 59
5.2 DeformGS maps a set of Gaussians with canonical properties to world space using a deformation function F. The deformation function takes in the position of a Gaussian (x, y, z) and a queried timestamp t, to infer shadow s, rotation R' and metric position (x', y', z'). During training, we use the metric positions and rotations to regularize the deformation function, considering the state at $t = \{i-1, i, i+1\}$ with Gaussian metric states P'_{t-1}, P'_t, P'_{t+1}. 64
5.3 DeformGS uses three adjacent timesteps at every iteration to enforce physics-inspired regularization terms. All Gaussians are deformed to world space using the deformation function F, and rasterized to compute the photometric loss and its gradients. We use the positions of the Gaussians in world coordinates to compute the regularization terms based on local isometry and conservation of momentum (Section 5.5.2). 65

5.4 This figure shows the rendering and tracking of DeformGS in the six dynamic Blender scenes used for evaluation. We will refer to the scenes in this figure as Scenes 1, 2, 3, 4, 5 and 6 ordered from left to right, and top to bottom. 70
5.5 Results on Scene 5. 70
5.6 A person manipulating a duvet in the Robo360 [176] dataset, reconstructed using DeformGS. The top row shows the 4D Gaussians as point clouds, where the color represents dense correspondences. The bottom row shows rendered views overlaid with 3D trajectories projected to image space. . . 71
5.7 3D point tracking on duvet (left) and cloth (right) folding scenes. Compared with 4D-GS (top), DeformGS (bottom) infers smoother, less tangled trajectories. 71

6.1 Cloth-Splatting state estimation of a real-world cloth. We introduce Cloth-Splatting, a 3D state estimation method for cloth from sparse RGB observations that combines Graph Neural Network (GNN) with Gaussian Splatting (GS). Given an initial sequence of 3D mesh predictions from the GNN (left), Cloth-Splatting (right) updates it using RGB observations, achieving state-of-the-art 3D tracking performance for deformables. 75
6.2 Overview of Cloth-Splatting. We estimate the 3D state of the cloth through a prediction-update framework. We use a graph neural network (GNN) to approximate the transition function that estimates the next state $\hat{\mathbf{M}}_{t+1}$ given an action $\mathbf{a}_t$ and the state $\mathbf{M}_t$. We then employ our proposed mesh-constrained Gaussian Splatting (GS) as a measurement model h_{GS} to obtain an estimate of the observation $\tilde{\mathbf{Y}}_{t+1}$ given the current refined state $\tilde{\mathbf{M}}_{t+1} = \hat{\mathbf{M}}_{t+1} + \delta\hat{\mathbf{M}}_{t+1}$. The differentiability of GS allows us to iteratively optimize the update $\delta\hat{\mathbf{M}}_{t+1}$ of the state estimate via the photometric consistency loss between the ground-truth observation $\mathbf{Y}_{t+1}$ and the rendered state $\tilde{\mathbf{Y}}_{t+1}$. . . 78
6.3 Qualitative results (sim). Tracking and rendering results on `SHIRT` (top) and `SHORTS` (bottom) scenes. 84
6.4 Time ablation. Convergence times of Cloth-Splatting and MD-Splatting on a `SHORTS` scene. The offset to the y-axis arises from the GNN predictions (1.36 s) and the 3D Gaussian initialization (26.55 s). 84

7.1 RaySt3R is a method for zero-shot 3D shape completion from a single foreground-masked RGB-D image. It predicts depth maps, object masks, and per-pixel confidence scores for novel viewpoints, and fuses them to reconstruct a complete 3D shape. RaySt3R is able to recover the geometry of full objects in cluttered real-world scenes, despite only being trained on synthetic data. 90

7.2 The architecture of RaySt3R. RaySt3R takes a single RGB-D image and foreground mask as input, and predicts depth maps, object masks, and per-pixel confidence scores for novel views. First, we apply the foreground mask to the RGB and point map input. Next, we use self-attention layers for the point map and ray map inputs, and feed the RGB image into the frozen DINOv2 [225] encoder. We feed all features into cross-attention layers followed by two separate DPT heads [244] for depth and mask predictions. Finally, we provide a confidence- and occlusion-aware multi-view merging formulation. 93
7.3 Qualitative results of RaySt3R in real-world multi-object scenes. For each scene, we select a subset of methods to preserve visual clarity, but we share all method predictions in the supplementary material of the original paper [73]. The results suggest RaySt3R produces the most geometrically accurate shapes compared to the baselines. The most competitive baseline, OctMAE [129], tends to predict softer shapes as a result of its coarse 3D MAE grid. We observe TRELLIS [361], Unique3D [356], and LaRI [162] struggle with relative object placing, aspect ratios, and out-of-distribution objects, occasionally leading to incorrect registration to the ground truth points. Finally, SceneComplete [1] proves to be brittle to single points of failure such as missing object masks. 100
7.4 Confidence threshold vs error metrics averaged over all real-world datasets. The results suggest increasing the confidence threshold improves accuracy and degrades completeness. 102
7.5 The impact of the confidence threshold on the predicted 3D points. This experiment suggests increasing the confidence threshold can aid in reducing the edge bleeding issue. 102

8.1 We present Modality Forcing, a post-training recipe to extract spatial priors from text-to-image (T2I) models. A single DiT models the joint distribution over images and depth, enabling joint and conditional generation in arbitrary combinations. We demonstrate depth predictions improve with increasingly capable T2I pre-training, suggesting T2I is a scalable objective for spatial generation. 104
8.2 Modality Forcing generates rich RGB-Depth from text prompts. Unprojecting the points to 3D, Modality Forcing generates faithful and sharp geometry. The same checkpoint enables monocular depth estimation, and depth-to-image generation competitive with the best specialist models. . . 105
8.3 Modality Forcing is a recipe to post-train image-generation models for depth prediction. We encode RGB using a pre-trained VAE and depth using a per-pixel tokenizer. During training (a), each modality is independently noised and trained with separate losses. At inference (b), each modality's noise level can be controlled to yield image→depth, depth→image, or joint generation. 108

8.4 Scaling experiments. Depth accuracy (δ_1, ↑, bottom) and AbsRel (↓, top) by T2I model size. Each line represents a T2I pre-training dataset size (none, 128M, 640M, 1.92B). Training larger T2I models on more image data yields better depth performance. 111
8.5 Qualitative image-to-depth generation results. Modality Forcing generates robust and sharp depth maps yielding plausible 3D point clouds. We find JointDiT [30] sometimes fails catastrophically, missing structure or misestimating scale. PPD [366] produces much closer results, but Modality Forcing produces more robust results. 114
8.6 Qualitative joint image-depth generation results. Modality Forcing samples RGB and geometry jointly, and proves to produce consistent and compelling point clouds. JointDiT [30] inherits compelling image generations from FLUX, but struggles with depth. 115
8.7 Modality Forcing inference-time analysis. We find that denoising RGB first acts as a kind of 'scratch pad' in latent space, leading to higher-quality depth predictions. 116
8.8 The denoising trajectory across depth and RGB dictates the strength of modality conditioning. Denoising more of depth early on means RGB will more rigidly match it. Here we show that this enables a new level of controllability of depth-conditioned image generation. 117

9.1 PointZero proposes *3D point track completion* as a pre-training objective for learning rich 3D dynamics priors. Given one RGB-D image and one or a few point trajectories (shown in orange), PointZero predicts the future 3D tracks of all observed scene points. PointZero is a robot-free pre-training objective for instilling 3D dynamics understanding in world models. Post-trained for dynamics modeling [392] and robot manipulation [125], it outperforms *application-specific* baselines. 120
9.2 PointZero architecture overview. PointZero takes in a single RGB-D image and a small number of complete point tracks, and predicts dense 3D point trajectories. PointZero can also be fine-tuned to condition on robot pose instead of partial point tracks, or to output robot action chunks in addition to point track predictions. PointZero combines a Perceiver-IO [131] architecture with self-attention and cross-attention to encode diverse dynamics. 124
9.3 We contribute a novel dataset to learn diverse dynamics in simulation and the real world. The dataset contains 2.9 million synthetic image frames with deformable, articulated and rigid objects. We also collect a dataset with 14 objects and 124 interactions in the real world for evaluation of zero-shot sim-to-real transfer. 127
9.4 Real-world qualitative 3D point track completion evaluation. All methods are trained with identical inputs and conditioning, and with identical data. We find GBND often predicts zero motion, while PTv3 and PGND struggle to recover rigid (articulated) motions. 130

9.5 Visualizations of the simulation and real-world tasks used for imitation learning evaluation. 138

10.1 3PoinTr learns manipulation from *unconstrained* human videos: videos where the human demonstrator can act freely rather than mimicking target robot kinematics. 3PoinTr first predicts dense 3D point tracks — *how the scene should move to complete the task* — and then conditions a closed-loop multitask policy on these tracks. 3PoinTr outperforms strong behavior cloning and learning-from-video baselines across simulated and real-world evaluations. 145
10.2 Diagram of the 3PoinTr network architecture. Given an initial point cloud, a transformer predicts dense 3D point tracks describing how objects should evolve during the task. A Perceiver-IO point track encoder compresses these tracks into a small set of learned point track tokens. The policy conditions on these point track tokens together with the current point cloud and robot state, enabling closed-loop action prediction. Beyond global conditioning, residual track-token cross-attention in every U-Net block gives the action head a direct path to the predicted object motion, encouraging a simpler mapping from point tracks to robot actions. 147
10.3 Simulation and real-world tasks used for evaluation. Simulation tasks use procedurally generated robot trajectories for both video pre-training and policy learning; real-world tasks use unconstrained human videos for video pre-training and robot teleoperation data for policy learning. Green tracks show 2D projections of the 3D point-track prediction targets (only tracks for moving points are visualized). 149
10.4 Ablation average success rates over simulation tasks with 20 robot demonstrations. 154
10.5 Visualization of the spatial variation used in our experiments. 156
10.6 Human vs. robot trajectories: 3PoinTr learns object motion from unconstrained videos (left) and maps it to robot actions (right); embodiment and object motions need not match. 157

11.1 We present ModAR, a world-action model (WAM) that predicts future observations as a sequence of multiple modalities. ModAR autoregressively denoises one modality at a time before finally denoising robot actions. We train all models from scratch for controlled comparisons across predicted modalities and training data. We find that additional modalities improve performance, and our multimodal formulation outperforms prior WAM formulations. We further validate ModAR on real-world bimanual manipulation using robot demonstrations and human videos. 162

11.2 WAM formulations. We draw block diagrams of the evaluated WAM formulations and the Action-only baseline, which differ only in how they couple future prediction with action prediction. ModAR (a) autoregressively denoises future modalities before denoising actions. Unified (b) represents joint-generation WAMs such as DreamZero and Cosmos Policy [148, 377], which denoise future-observation and action streams together. Disjoint (c) represents Fast-WAM [383], which predicts future observations and actions independently and omits future-observation prediction at inference time. Action-only (d) denoises actions directly without predicting future observations. Independent-noise (not drawn) is identical to Unified (b) except during training: rather than applying one shared noise level to all predicted streams, it samples each stream's flow timestep independently. 164
11.3 ModAR architecture. We embed the current observation from each modality and pass the resulting tokens into a shared diffusion transformer. The DiT combines cross-modal blocks with modality-specific experts and output heads. Robot configuration and a learned task embedding provide global adaLN conditioning to each DiT layer. ModAR sequentially denoises future modalities, conditioning each on earlier predictions, and denoises actions last. 166
11.4 RoboTwin simulation results. Success rates over six tasks with 50 trials per task. We train all models with 50 action-labeled demonstrations per task, then add actionless demonstrations to increase the total demonstration count from 50 to 250 to 1,250. (a) ModAR achieves the highest average success rate among the formulations at each data scale. (b) We find it beneficial to predict future tracks, DINO, and depth, while additionally predicting future RGB provides no consistent gain. RGB (dark blue) matches the typical WAM setting of predicting only future RGB and actions. Independent-noise is identical to Unified except during training: rather than applying one shared noise level to all predicted streams, it samples each stream's flow timestep independently. 170
11.5 Evaluating predicted futures with a separate IDM. Recall from Section 11.3 that ModAR's final action-prediction step acts as an inverse-dynamics model (IDM), mapping its future-observation predictions to actions. In contrast, Unified denoises futures and actions jointly, so its action prediction conditions on partially noisy predicted futures. To remove this asymmetry, we discard both models' native action predictions and pass their predicted futures to the same separately trained IDM. ModAR still outperforms Unified, showing that its predicted futures are themselves more useful for predicting actions. 173

11.6 ModAR ablations in RoboTwin. Average success rate over six tasks, with 250 total demonstrations per task and 50 action-labeled demonstrations. We ablate context noise (No context noise), generation order (Reverse order), and each individual future-observation modality. Results suggest that context noise is critical for robust autoregressive generation. Generating modalities in reverse order also reduces performance, supporting our hypothesis that more structured modalities act as "scratchpads" for more detailed modalities. Removing future RGB prediction does not reduce performance, suggesting that RGB contributes the least of the four future-observation modalities. 175
11.7 Real-world results. Success rates on real-world tasks, with 30 trials per model per task. (a) All methods use 100 robot demonstrations per task; ModAR and Unified additionally use 200 in-domain actionless human demonstrations and 1,000 out-of-domain EgoDex demonstrations. ModAR achieves the highest success rate on all three tasks. (b) We train separate ModAR models with just 100 robot demonstrations per task, then adding 200 in-domain actionless human demonstrations (used to supervise future-observation predictions, but not action prediction), and then also adding 1,000 out-of-domain human demonstrations from EgoDex (used similarly). The average success rate progressively improves from 70.0% to 81.1% to 83.3%. 177

12.1 Two trajectories of AI progress. Circles represent the capabilities required for human jobs; hatched regions represent tasks an AI system can perform. The upper row illustrates broad improvement toward artificial general intelligence (AGI). The lower row illustrates increasingly uneven capabilities, with exceptional performance in a few domains and persistent gaps elsewhere. Adapted from an illustration by Adam Hunt. 182

List of Tables

2.1 Number of features detected in datasets collected on a GoPro Hero 8 camera and the Lensagon BF5M ultra-wide fisheye lens. The percentages signify the fraction of features detected w.r.t. the theoretical maximum (i.e., number of frames $\times$ number of features). 19

2.2 The normalized number of features detected using the BF5M ultra-wide fisheye camera, sorted by polar angle. The results clearly show that TartanCalib detects most features, especially at the edge of the lens. 19

2.3 The average detection error (px) in feature detection as compared to human-annotated data on 1,440 data points. The results show TartanCalib outperforms Kalibr significantly, and it shows symmetry-based refinement performs slightly better than `cornerSubPix()` on the GoPro dataset, but it performs worse on the high-distortion Lensagon BF5M fisheye lens dataset. This shows that symmetry is an unstable metric for feature refinement. 21

2.4 Reprojection error evaluated on the Lensagon BF5M ultra-wide fisheye camera and GoPro Hero 8. DS-None: Double-Sphere model [311], Omni-Radtan: Omnidirectional camera model with Radial-Tangential distortion [303], and the Central Generic Model [18, 267]. 21

3.1 Results on Tanks&Temples in terms of ATE and overall registration rate (Reg.). For easier readability, we color-code ATE results as a linear gradient between worst and best ATE for a given dataset or split; and Reg results with linear gradient between 0% and 100%. Top: impact of the number of input views, regularly sampled from the full set. 'N/A' indicates that at least one scene did not converge. Bottom: ATE$\downarrow$ on different datasets with the arbitrary splits defined in FlowMap [275]. 36

3.2 Multi-view pose regression on CO3Dv2 [247] and RealEstate10K [401] with 10 random frames. Parentheses () denote methods that do not report results on the 10-view set; we report their best for comparison (8 views). We distinguish between (a) multi-view and (b) pairwise methods. 37

3.3 Detailed per-scene translation and rotation accuracies ($\uparrow$) on ETH-3D. For clarity, we color-code results with a linear gradient between the worst and best result for a given scene. 38

3.4 Ablation of scene graph construction on Tanks&Temples (200 view subset). See text for details. 39
3.5 Ablations on Tanks&Temples (200 view subset). See text for details. 39
3.6 Comparison of retrieval based on MASt3R features using ASMK with the state-of-the-art FIRe method when localizing with MASt3R (bottom rows), as well as with other state-of-the-art visual localization methods (top rows). 40

4.1 Root Mean Square Error (RMSE) in synthetic Blender Scenes. 52
4.2 Mean Absolute Error (MAE) in synthetic Blender Scenes. 52

5.1 3D tracking results on the deformable cloth dataset (Figure 5.4). For each metric, the methods above the solid line had access to privileged information, see ^{ab} and Section 5.6.1 for more details. The results suggest that DeformGS outperforms the baselines in all averaged metrics, and is competitive with the oracle models. The results also suggest our novel deformation function architecture, learning per-Gaussian masks, and physics-inspired regularization losses improve the tracking performance compared to 4D-GS [354]. We do not consider the oracle methods to be fair baselines and therefore do not bold their results. 69

6.1 Quantitative evaluation. Comparison of Cloth-Splatting and the baselines in tracking quality. We report the mean and the standard deviation per metric ($\mu \pm \sigma$) and mark the best result bold. 82
6.2 Model ablation. 83
6.3 Manipulation results. We report the mean and standard deviation of the MSE computed between the final cloth state and the goal state. Results are presented in units of 10^{-3}. 84

7.1 Quantitative evaluation of multi-object scene completion on synthetic and real-world datasets. We evaluate on the test split of OctMAE [129], and the BoP benchmarks YCB-Video [362], HOPE [309], and HomebrewedDB [142]. We report Chamfer distance (CD) [mm] and F1-Score@10mm (F1). The first section contains numbers copied from OctMAE [129], the second section contains recent works we evaluated. For alignment of LaRI [162], Unique3D [356], and TRELLIS [361] with the ground truth mesh, we apply brute force search followed by ICP [277]. We evaluate TRELLIS [361] with masked and unmasked RGB inputs. SceneComplete [1] runs out of VRAM on HOPE [309]. The results suggest RaySt3R outperforms all baselines. 99
7.2 Ablation study on RaySt3R training on the YCB-Video dataset [362]. The results suggest data scale, data diversity, DINOv2 [225] features, data augmentation, and model size affect RaySt3R's performance. 101

7.3 Ablation study on RaySt3R view merging on the OctMAE test dataset [129]. We ablate querying the input view (Section 7.4.3), occlusion masking with the input mask, and finally RaySt3R's predicted masks. The results suggest all steps contribute to performance, especially the predicted masks. 101

8.1 Training data. We train on twelve real-world and simulated datasets totaling 17M frames across 58k scenes, spanning indoor scanning, outdoor driving, and synthetic rendering. 111
8.2 Base T2I model sizes used for scaling experiment, ranging from 370M to 3.3B parameters. 111
8.3 Affine-invariant depth estimation. Comparison of monocular depth estimation methods on five benchmarks. The results suggest Modality Forcing outperforms all models built for joint RGB-Depth generation, and generative depth estimators. Modality Forcing approaches the performance of the very best depth models such as MoGe-2 [334], scoring better on NYUv2 [273] and ETH3D [268]. The overall best are underlined, within-type best are bolded. 113
8.4 Depth-conditioned image generation on 6,000 OpenImages samples. . . . 115

9.1 Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in 10^{-2} m^2. Bold marks the best result in each column, and underlining indicates the second-best result. 129
9.2 Dynamics prediction errors on the PGND benchmark. 131
9.3 Imitation learning success (%). PointZero uses 20 labeled demos and 100 actionless videos per task. 132
9.4 Pre-training with downstream track supervision. Success (%); 20 demos, no extra videos. 132
9.5 Pre-training without downstream track supervision. Success (%); 20 demos, no extra videos. 132
9.6 Data-scaling ablation on the articulated split of the real-world dataset: performance degrades gracefully as the pre-training dataset is reduced. Error values are in centimeters; MSE is in cm^2. Training epochs are scaled inversely with the data fraction (250 epochs at 10%, 500 at 5%). Bold marks the best result for each metric. 136
9.7 Model-size ablation on the articulated split of the real-world dataset. Small uses 384 token dimensions, 6 heads, and 6 layers (45M parameters) versus the Base configuration (768 dimensions, 12 heads, 12 layers). Error values are in centimeters; MSE is in cm^2. Bold marks the best result for each metric. 136

9.8 Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in 10^{-2} m^2. First-sample, mean, pointwise medoid, per-metric oracle, and pooled scalar median statistics use ten seeded predictions and are defined in Section 9.7.6. Pooled medians summarize a different statistic from mean errors; no cross-statistic ranking is applied. . . 137
9.9 Dynamics prediction errors on the real-world dataset (centimeters; MSE in cm^2). All baselines were trained on the proposed simulation dataset. "-first" rows use a single sample; PointZero-FM and PointZero-JiT report all metrics of the sample with the lowest MDE out of 10 (best-of-10). Bold marks the best result in each column, and underlining indicates the second-best result. 138
9.10 Evaluation of different 3D dynamics architectures on zero-shot predictions on the PGND benchmark [392]. All methods were trained on synthetic data from the PointZero dataset, and evaluated on unseen PGND scenes. Errors are in centimeters. Bold marks the best result in each column. 139

10.1 Average Displacement Error (ADE) and ADE of the 5 % of points that move the most (5% ADE) for 3PoinTr and General Flow [382] on real-world tasks. Units are millimeters. 152
10.2 Simulation success rates (%) evaluated over 200 rollouts per task. Results are reported for policies trained with 20, 50, and 100 action-labeled demonstrations and 100 actionless videos per task. 153
10.3 Real-world success rates for policies trained with 20 robot demonstrations and 50 human videos per task. 3PoinTr achieves the highest success rate on all four tasks, with a 25.0 percentage point higher average success rate than the best baseline, DP3. 153
10.4 Average Displacement Error (ADE) and ADE of the 5 % of points that move the most (5% ADE) for 3PoinTr and General Flow [382] on simulation and real-world tasks. Units are millimeters. 158
10.5 Ablation simulation success rates (%) evaluated over 200 rollouts per task. Results are reported for policies trained with 20 demonstrations. 158

11.1 Per-task success rates by WAM formulation at total-demonstration scale D: action-only, independent-noise, disjoint, unified, and ModAR. 171
11.2 Per-task success rates by predicted-modality subset at total-demonstration scale D. Modalities comprise RGB, depth, point tracks, and DINO features. 172

Chapter 1

Introduction

Robot manipulation promises to automate repetitive, physically demanding and dangerous work. Carefully engineered systems can already deliver high throughput, precision, and reliable execution of many tasks. This good old-fashioned engineering (GOFE) is effective when tasks are highly repeatable, but extending to new tasks requires vast resources.

The goal of this dissertation is to lay the groundwork to develop systems that can adapt to new and non-repeatable tasks with minimal human intervention. Towards this goal, learning-based approaches aim to acquire manipulation skills from data. In imitation learning, we train a policy on observations paired with demonstrated robot actions. In world modeling, we learn from a much broader source of data: unstructured (visual) data without robot action labels. By learning to predict observations and how scenes change, a model can acquire a broad understanding of the world without robot action labels.

Recently, researchers have scaled imitation learning and world modeling towards broadly capable foundation models. These models often break the training process into two parts: pre-training and post-training. During *pre-training*, a model acquires broadly useful knowledge from diverse data. During *post-training*, researchers adapt that knowledge to a particular set of robot embodiments and tasks using demonstrations or interaction. Pre-training helps achieve robust task performance, but adaptation still requires astonishing amounts of data, computation, and human effort. One potential explanation for the inefficiency is that frontier models perceive and generate in RGB pixel space directly, which is a noisy modality with low signal for robotics tasks.

This dissertation proposes methods for acquiring *spatiotemporal intelligence*. Like humans, robots benefit from understanding spatial relationships over time. From infancy, humans develop object permanence: the understanding that objects continue to exist even when they are out of sight. Reasoning about the physical world beyond immediate observations also matters for robot manipulation. I develop methods that learn geometry and dynamics to infer unobserved structure and predict how scenes change.

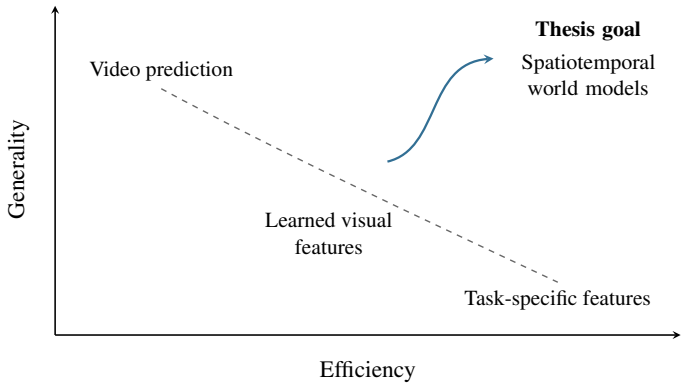


Figure 1.1: **Generality and efficiency in representation design.** This thesis investigates spatiotemporal world models that retain broadly useful physical structure while focusing prediction on information relevant to manipulation. The representation choices and dashed curve are schematic, rather than measured comparisons.

1.1 General and efficient representations

To frame this dissertation's investigation of general and efficient spatiotemporal representations, I define four design goals: sufficient, obtainable, learnable, and focused (*SOLF*) representations. These goals provide an application-independent framework for comparing representations, although their relative importance depends on the application.

Sufficient. Retain the information needed to choose high-quality actions for the intended tasks.

Obtainable. Be recoverable from broadly available images, video, or simulation with an acceptable yield rate.

} General

Learnable. Make task-relevant information easy to extract and predict.

Focused. Retain only the information necessary to execute the task.

} Efficient

When a representation is sufficient and obtainable, we may call it *general*. We may call a representation that is both focused and learnable *efficient*. Notably, efficiency is not only the number of bytes a representation uses to describe a state (focus), it also measures the effort it takes to retrieve task-relevant features (learnability). Figure 1.1 shows existing trade-offs.

RGB images are cheap to *obtain*, being close to a raw sensor reading. Images also contain rich world information and are *sufficient* for many tasks, making them a *general* representation. However, images are hard to learn from and generate and lack focus on task-relevant features, making them *inefficient*.

Task-specific features can focus on the quantities needed for a particular behavior, but they are not general or scalable. Learned visual features from DINO, a self-supervised visual encoder, offer a more general alternative, with higher learnability than images. This thesis investigates explicit geometry and motion as another route to combining broad applicability with focused prediction, as Figure 1.1 illustrates.

1.2 Thesis contributions

The work in this thesis comprises three complementary parts. First, I develop methods to *reconstruct* past experiences with 3D geometry and correspondences over time, which can serve as a policy input or an auxiliary learning objective. Second, I propose generative formulations that learn focused geometric and dynamic priors from visual data. Third, I incorporate these priors into spatiotemporal world-action models (WAMs) and demonstrate their benefits on simulated and real-world robot manipulation tasks.

1.2.1 Spatiotemporal reconstruction (Part I)

Part I develops methods for recovering 3D geometry and dynamics from visual observations (Figure 1.2). The objective is to reconstruct the complete geometry and correspondences of the scene over time, which can provide supervision for downstream tasks.

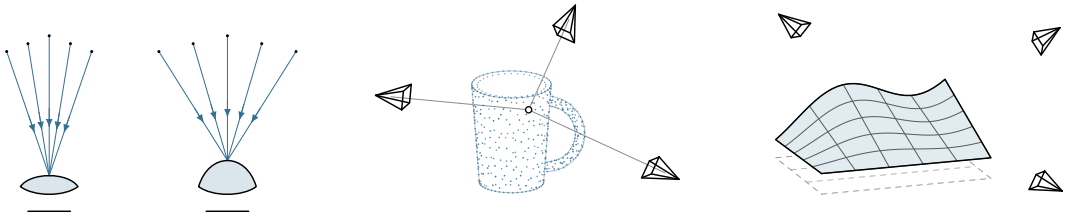
A rich body of work exists on reconstruction, recovering geometry from arbitrary image sets through structure-from-motion (SfM), 2D and 3D tracking, and geometric camera calibration. The constraints of these methods limit their adoption: existing methods only perform well in constrained cases, such as dense captures for SfM. This part aims to relieve some of these constraints in order to make spatiotemporal representations more *obtainable*, and as a result more *general*.

1.2.1.1 Geometric camera calibration

Geometric camera calibration is a crucial prerequisite for many downstream spatiotemporal reconstruction tasks such as stereo depth or visual-inertial odometry (VIO). It aims to recover the intrinsics and extrinsics of a set of cameras. Intrinsics capture the ray direction of each pixel, whereas extrinsics contain the relative pose between cameras.

TartanCalib (Chapter 2) focuses on wide-angle camera calibration. Wide-angle cameras offer a compelling form factor for mobile robots but are notoriously challenging to calibrate. TartanCalib improves wide-angle camera calibration through iterative target reprojection and adaptive subpixel refinement. TartanCalib uses intermediate camera estimates to improve feature detection, which increases calibration accuracy and supports more accurate downstream depth mapping.

TartanCalib requires dedicated calibration captures, which are not always available. In MASt3R-SfM (Chapter 3), I contribute a method to calibrate multiple cameras using a single image from each camera (details in Section 1.2.1.2).



(a) **Geometric camera calibration.** TartanCalib (Chapter 2) and MASt3R-SfM (Chapter 3).

(b) **Static 3D reconstruction.** MASt3R-SfM (Chapter 3) and Residual-NeRF (Chapter 4).

(c) **Dynamic 3D reconstruction.** DeformGS (Chapter 5) and Cloth-Splatting (Chapter 6).

Figure 1.2: Reconstruction problems studied in Part I: calibrating cameras, recovering camera poses and scene geometry, and reconstructing geometry as it changes over time.

1.2.1.2 Static 3D reconstruction

Reconstructing previous experiences in 3D may help provide richer supervision, and equip robots with spatial understanding. *MASt3R-SfM* (Chapter 3) uses learned pairwise geometry and correspondences to jointly estimate cameras and 3D structure from image collections. Global alignment and efficient image retrieval make these predictions usable for reconstruction in challenging capture conditions, including limited covisibility or limited motion.

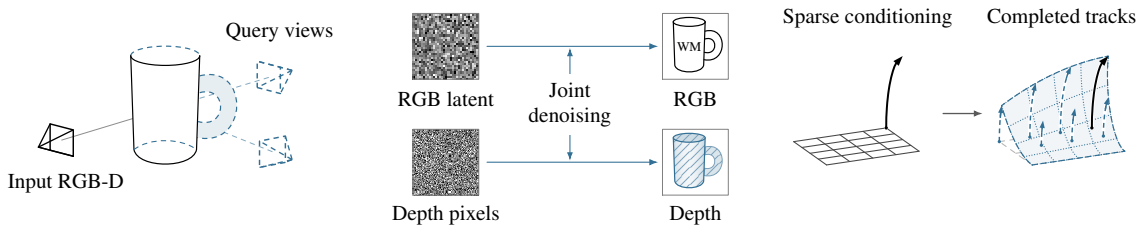
Next, I use the calibrated captures for reconstruction with neural rendering techniques. In many robotics use cases, such as a factory working cell, we know the environment ahead of time. Residual-NeRF (Chapter 4) uses a background reconstruction to improve the speed and accuracy of geometric reconstruction. Residual-NeRF deploys a neural radiance field (NeRF) to estimate the geometry of transparent objects. It learns a residual NeRF and improves reconstruction accuracy, training speed, and grasp planning.

1.2.1.3 Dynamic 3D reconstruction

Static 3D reconstruction captures scene geometry but does not model how scenes evolve under interaction. I extend my previous work to recover 3D correspondences.

DeformGS (Chapter 5) recovers dense 3D motion from synchronized, calibrated multi-view videos. A canonical Gaussian representation, a learned deformation function, and physics-inspired regularization improve tracking in scenes with large deformations, shadows, and occlusions. DeformGS unlocks millimeter-precise tracking in challenging scenes.

Cloth-Splatting (Chapter 6) extends DeformGS by learning a generic cloth dynamics model as a prior. It couples a cloth mesh to 3D Gaussians and allows RGB observations to update predicted states. Compared to DeformGS, it improves estimation quality and convergence time.



(a) **3D shape completion.** RaySt3R (Chapter 7).

(b) **RGB-D generation.** Modality Forcing (Chapter 8).

(c) **3D point-track completion.** PointZero (Chapter 9).

Figure 1.3: Spatiotemporal generation in Part II: unobserved object shape, depth through joint RGB–depth denoising, and 3D point tracks from sparse conditioning. Blue dashed curves and dotted meshes mark completed geometry and motion.

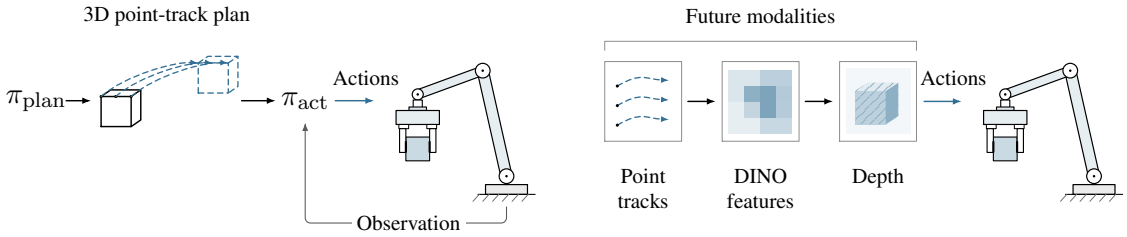
1.2.2 Generative spatiotemporal priors (Part II)

Reconstruction provides rich evidence in observed scenes, but humans can reason about unobserved states, whether current or future. Part II studies how models can learn such priors and apply them to new tasks, with particular attention to the supervision they require. That is, I contribute methods that absorb easily *obtainable* data. Figure 1.3 summarizes the shape, depth, and motion completion problems studied in this part.

RaySt3R (Chapter 7) formulates object completion as novel-view depth prediction. From a single RGB-D observation and a foreground mask, it predicts depth, masks, and confidence at queried viewpoints and then fuses the predictions into a completed shape. Training on synthetic views produces a prior that transfers to real objects and cluttered scenes.

Modality Forcing (Chapter 8) adapts text-to-image models to jointly generate RGB and depth, with a training formulation that accommodates sparse depth supervision. Controlled experiments show that additional image-only pre-training improves depth prediction with fixed depth training data. This establishes a way to acquire geometric knowledge from data without geometric labels.

PointZero (Chapter 9) learns dynamics through 3D point-track completion: given an RGB-D observation and sparse future trajectories, it predicts the motion of the remaining observed points. Pre-training uses simulated interactions without robot action labels. The pre-training transfers to real-object motion completion and improves downstream manipulation.



(a) **Track-conditioned control.** 3PoinTr (Chapter 10).

(b) **Modality-autoregressive control.** ModAR (Chapter 11).

Figure 1.4: World–action models (WAMs) in Part III: generating 3D point-track plans to condition a closed-loop policy, and denoising future modalities sequentially to condition robot actions. Point tracks, visual features, and depth in a single WAM.

1.2.3 Spatiotemporal world-action models (Part III)

A world-action model connects predictions of the environment to the generation of robot actions. Part III studies which future representations to predict and how a policy should use them. Figure 1.4 illustrates the two approaches studied.

3PoinTr (Chapter 10) learns 3D scene-motion plans from human videos and uses robot demonstrations to train a closed-loop policy conditioned on those plans. Its design separates desired scene motion from robot-specific actions, allowing the policy to benefit from human demonstrations that need not reproduce the robot's movements. The experiments suggest that 3D track plans outperform 2D track plans.

ModAR (Chapter 11) predicts future modalities sequentially before generating actions. In controlled simulation comparisons, this formulation achieves the highest average success rate at each tested data scale. Point tracks, depth, and DINO features provide complementary gains, while adding future RGB prediction does not produce a consistent improvement.

Chapter 12 synthesizes these findings through SOLF and develops future directions in selective prediction, online adaptation, and the role of world modeling in control.

Part I

Spatiotemporal Reconstruction

Chapter 2

TartanCalib: Iterative Wide-Angle Lens Calibration using Adaptive SubPixel Refinement of AprilTags

ORIGINALLY APPEARED AS

Bardienus P. Duisterhof, Yaoyu Hu, Si Heng Teng, Michael Kaess, Sebastian Scherer.
TartanCalib: Iterative Wide-Angle Lens Calibration using Adaptive SubPixel Refinement of AprilTags [69].
arXiv, 2022.

Chapter in Context

Accurate geometric camera calibration is a critical prerequisite for many reconstruction applications, such as visual-inertial odometry (VIO) or multi-view stereo (MVS). Wide-angle lenses are a desirable form factor as they reduce ambiguity and increase covisibility between views. But accurate calibration at the highly distorted edge is notoriously difficult. Here we introduce a novel method and easy-to-use calibration toolbox that is used around the world. In SOLF, this improves *obtainability* by making accurate geometric calibration more accessible.

Wide-angle cameras are uniquely positioned for mobile robots, by virtue of the rich information they provide in a small, light, and cost-effective form factor. An accurate calibration of the intrinsics and extrinsics is a critical prerequisite for using the edge of a wide-angle lens for depth perception and odometry. Calibrating wide-angle lenses with current state-of-the-art techniques yields poor results due to extreme distortion at the edge, as most algorithms assume a lens with low to medium distortion closer to a pinhole projection. In this chapter we present our methodology for accurate wide-angle calibration. Our pipeline generates an intermediate model, and leverages it to iteratively improve feature detection and eventually the camera parameters. We test three key methods to utilize intermediate camera models: (1) undistorting the image into virtual pinhole cameras, (2) reprojecting the target into the image frame, and (3) adaptive subpixel refinement. Combining adaptive

subpixel refinement and feature reprojection significantly improves reprojection errors by up to $26.59\,\%$, helps us detect up to $42.01\,\%$ more features, and improves performance in the downstream task of dense depth mapping. Finally, TartanCalib is open-source and implemented into an easy-to-use calibration toolbox. We also provide a translation layer with other state-of-the-art works, which allows for regressing generic models with thousands of parameters or using a more robust solver. To this end, TartanCalib is the tool of choice for wide-angle calibration. Project website and code: `http://tartancalib.com`.

2.1 Introduction

Cameras with wide-angle lenses enlarge the field-of-view (FOV) of a mobile robot in a compact form factor. This feature provides many benefits for crucial tasks such as visual odometry and depth mapping since a larger FOV brings more visual features. However, careful calibration is mandatory to fully leverage the image regions with high levels of distortion. Typically, calibration is obtained by showing the camera a known calibration target, which is then used to estimate the intrinsics of the camera model and extrinsics for multi-camera systems. Unlike ordinary lenses, wide-angle lenses, due to the high level of distortion in the image, impose certain special challenges: (1) it is challenging to robustly and accurately detect the visual features of the calibration target, due to extreme lens distortion. (2) The camera models available may not fit the lens very well. Previous work predominantly focused on obtaining more suitable camera models [59, 138, 267, 311], or finding better procedures to fit the camera models [190, 262, 263, 310]. Consequently, most publicly available tools only provide acceptable calibration results in the middle region of a wide-angle lens, where distortion is moderate, leaving the majority of the highly distorted border areas unusable. To make more image regions usable and effectively explore the visual information embedded near the border of a wide-angle lens, we need better calibration that delivers improved intrinsics and extrinsics parameter estimation.

Here, we focus on accurate and robust target detection in the high distortion regions. State-of-the-art target calibration pipelines [153, 224, 267] fail in the presence of high distortion, as shown in Figure 2.1. The two key procedures of these calibration pipelines, target detection and feature refinement, rely on the assumption of low to medium distortion or a camera projection that is close to a pinhole camera. These assumptions are violated in the case of wide-angle lenses, especially near the image border. The result is that many features are either not detected or are not detected in an accurate way. For those sparsely detected border features, their pixel locations tend to be inaccurate because of poor performance of the feature refinement methods in the high distortion regions. The above issues deteriorate the quality of the estimated camera model and limit the usability of the border region of a wide-angle lens.

In this chapter, we propose an iterative calibration pipeline (Figure 2.2), which consists of three core elements: (1) undistortion of the original image, (2) reprojecting the target into the image frame using the intermediate camera model, and (3) adaptive subpixel refinement based on reprojected target size. For our method, we develop two new subpixel feature

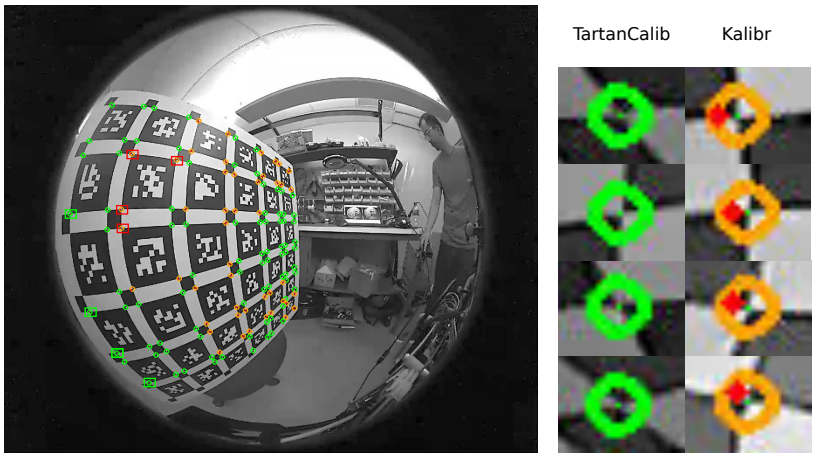


Figure 2.1: Comparison of target detection and feature refinement between TartanCalib and Kalibr. (Left) Target detection. Green circles: features newly detected by TartanCalib, orange circles: features previously picked up by Kalibr [245]. (Right) Zoomed-in view of detected features. Green point inside green and orange circles: refined features of TartanCalib, red point inside orange circle: features from Kalibr. TartanCalib detects more features near the image border and the features have better location accuracy.

refinement methods to facilitate accurate target detection in highly distorted regions, towards a better overall calibration in the border area of a wide-angle lens. Our contributions are:

- A novel methodology for wide-angle lens calibration, using iterative target reprojection and adaptive subpixel refinement.
- We show the benefit of our pipeline using traditional quality metrics such as reprojection error, as well as some downstream tasks relevant to mobile robots.
- We present our pipeline as an open-source easy-to-use package, which we term 'TartanCalib'.

Using our method, we find up to $42.01\,\%$ more features, and up to $26.59\,\%$ lower overall reprojection error. The entire pipeline is made open-source and integrated into Kalibr.

2.2 Related Work

This Section lays down the related work in the areas of camera models (Section 2.2.1), calibration toolboxes (Section 2.2.2), and pattern design and feature detection (Section 2.2.3).

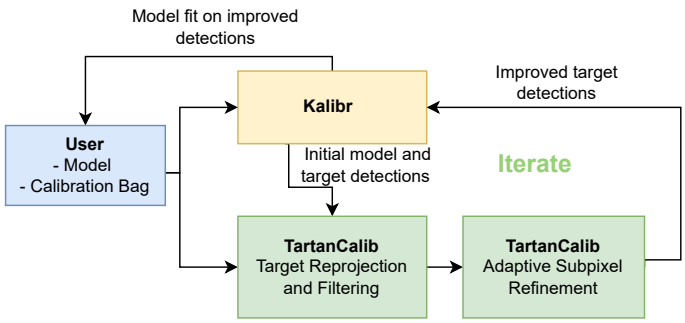


Figure 2.2: The pipeline of TartanCalib. The pipeline consists of an iterative calibration procedure with a newly proposed adaptive feature refinement method.

2.2.1 Camera Models

A typical calibration procedure needs to select a parametric or generic camera model for a lens and estimates the model parameters during the calibration process. There are a number of models designed specifically for wide-angle lenses, as their projection is significantly different from low-distortion cameras. Some of the more common models are the Double Sphere model [311], the Kannala-Brandt model [138], and the Field-of-View model [59]. Parametric models typically have only a few degrees of freedom, making their parameters easier to be estimated as compared with generic models, but providing a trade-off in accuracy. Generic models have far more parameters, aiming to more accurately represent lens geometry. It has been shown that these models have a significantly lower reprojection error [267]. A distinction can be made between non-central generic models and generic models. Central generic models [18, 19] assume all observation lines intersect in the center of projection, whereas non-central generic models do not make that assumption [101, 243]. Typically non-central generic models perform better but may be more complicated to deploy (e.g., undistortion to a pinhole image is not possible without knowing pixel depth). Our toolbox, TartanCalib, supports both parametric and generic camera models to achieve the best possible calibration.

2.2.2 Calibration Toolboxes

As geometric camera calibration is an important prerequisite for many machine vision applications, numerous calibration toolboxes have been developed [281, 310, 395]. The famous computer vision package OpenCV [28] has its own wide-angle lens calibration support, and supports checkerboard targets. OcamCalib [310] is another well-known toolbox, using exclusively (less accurate [104, 224]) checkerboard targets for calibration.

Recently BabelCalib [190] was proposed, with its robust optimization strategy being the key advantage. However, the most commonly used calibration toolbox is Kalibr [245], which is easy to use and allows for retrieving the intrinsics and extrinsics of multiple cameras with a wide variety of camera models and targets. In this chapter, TartanCalib is integrated

into Kalibr [245] as an easy-to-use toolbox. In addition to Kalibr, TartanCalib also supports the use of BabelCalib as a solver, and the generally more accurate generic models [267].

2.2.3 Pattern Design and Feature Detection

Target detection is one of the key functions of a calibration pipeline. By far the most commonly used calibration targets are checkerboard [28], dot patterns [112], and AprilTags [224]. Dot patterns are susceptible to perspective and lens distortion, whereas a checkerboard tends to fail when it is only partially observed, which makes calibrating wide-angle lenses extremely challenging. Some researchers proposed to use novel patterns, such as triangle features [104, 267] to increase the gradient information, but these typically are not robust enough for the high distortion as present at the edge of a wide-angle (fisheye) lens.

In [267], the authors use a single AprilTag to determine the pose of a custom target and assume a homography as a camera model to reproject the target onto the image frame. This approach has three fundamental issues for high-distortion wide-angle lenses: (1) using a single AprilTag is not robust enough, (2) using a homography as a camera model for target reprojection will impose a reprojection error that makes it impossible to recover the true target position, and (3) the refinement method used is shown to be unstable for AprilTags (and checkerboards).

Inspired by [267], we propose a pipeline designed for a grid of AprilTags. TartanCalib adopts an iterative process, that makes it possible to use a relatively accurate intermediate camera model instead of a homography, to reproject target points into the image frame. Additionally, we propose two novel adaptive subpixel refinement methods, arriving at more features detected with superior subpixel accuracy.

2.3 Preliminaries

This section addresses the preliminaries related to notation (Section 2.3.1) and camera models (Section 2.3.2), necessary for this chapter.

2.3.1 Notation

Our notations are inspired by [311]. In the equations presented later, lowercase characters are scalars (e.g., α), whereas lowercase bold characters (e.g., $\boldsymbol{x}$) are vectors. Matrices are denoted using bold uppercase letters (e.g., $\boldsymbol{T}$). Pixel coordinates are represented as $\boldsymbol{u} = [u, v]^\intercal \in \times \subset \mathbb{R}^2$, here $\times$ is the image domain. 3D points are presented as $\boldsymbol{x} = [x, y, z]^\intercal \in \otimes \subset \mathbb{R}^3$, here $\otimes$ is the subset of valid 3D points that can be projected into the image frame. We denote the transformation from the camera frame to the calibration target frame as $\boldsymbol{T}_{\text{tar}} \in SE(3)$. The image matrix is denoted as I.

2.3.2 Camera Models

A camera model typically consists of a projection and unprojection function. The projection function $\boldsymbol{\pi} : \otimes \rightarrow \times$ projects a 3D point to image coordinates. Its inverse, the unprojection function: $\boldsymbol{\pi}^{-1} : \times \rightarrow \mathbb{S}^2$ unprojects image coordinates onto a unit sphere. The projection of a 3D point can be described as $\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{i})$ where $\boldsymbol{x}$ is a point in 3D space, and $\boldsymbol{i}$ is the set of parameters for the camera model. Similarly, the unprojection function is denoted as $\boldsymbol{\pi}^{-1}(\boldsymbol{u}, \boldsymbol{i})$, where $\boldsymbol{u}$ is the coordinate in image space.

2.4 Method

The high-level idea behind TartanCalib (Figure 2.2) is to iteratively optimize a camera model, by leveraging intermediate camera models to improve target detection. The iteration includes several key components that will be detailed in the following sections. The components are Undistortion (Section 2.4.1), Target Reprojection (Section 2.4.2), Corner Filtering (Section 2.4.3), and Subpixel Refinement (Section 2.4.4).

The visual features of the calibration target manifest themselves as corner features. In later sections, we interchangeably refer to corners as target features.

2.4.1 Undistortion

An intuitive way to improve target detection in wide-angle camera calibration is to undistort the image into multiple pinhole reprojections. This approach should get rid of some of the difficulties caused by highly distorted targets. To undistort the image we model a virtual pinhole camera, which has four parameters $\boldsymbol{i} = [f_x, f_y, c_x, c_y]^\intercal$. The projection function is defined as:

$$\boldsymbol{\pi}(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2.1}$$

where f_x and f_y are focal lengths and c_x and c_y are the pixel coordinates of the principal point. Creating a virtual pinhole camera is possible by first unprojecting the pinhole pixel coordinates to $\mathbb{S}^2$ space, to then reproject those points back into the distorted image frame. We then query the pixel location at that location in the distorted frame and substitute it back into the pinhole image to arrive at an undistorted image.

2.4.2 Target Reprojection

While undistortion reduces lens distortion, we may still be unable to detect the target due to perspective distortion and other visual artifacts such as motion blur. As proposed in [267], it is possible to reproject known target coordinates back into the image frame without actually detecting the target. The authors show that a homography can be used for this

purpose. Equation 2.2 shows how a camera model can be used to reproject a point from target coordinates ($\boldsymbol{x}_\text{t}$) to image coordinates ($\boldsymbol{u}_\text{t}$).

$$\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{i}) = \boldsymbol{\pi}(\boldsymbol{T}_\text{tar} \cdot \boldsymbol{x}_\text{t}, \boldsymbol{i}) \tag{2.2}$$

Here $\boldsymbol{T}_\text{tar}$ is the transformation from the target frame to the camera frame, $\boldsymbol{x}_\text{t}$ is a coordinate in the target frame, and $\boldsymbol{x}$ is a vector in the camera frame.

2.4.3 Corner Filtering

While reprojecting the target into the image frame using an intermediate camera model may yield somewhat accurate estimates, it is uncertain if all of the target is visible in the frame. Therefore, a filtering policy is required, only keeping the features (corners) that appear in the frame. We achieve robust filtering by following these steps: 1) loop over all detected quads (detected squares), 2) check if all 4 corners of each quad are close to a reprojected target corner, and 3) perform subpixel refinement on all corners.

2.4.4 Subpixel Refinement

Subpixel refinement is required to translate the features reprojected using the intermediate model into features that actually match the corners as seen in the image. We propose two algorithms: 1) a simple modification to OpenCV's `cornerSubPix()` function [86], and 2) a symmetry-based refinement method specifically designed for high-distortion lenses.

2.4.4.1 Adaptive cornerSubPix()

`cornerSubPix()` [86] computes the image gradient within a search window in order to iteratively converge towards the corner. In doing this, the size of the search window is a critical hyperparameter: if the search window is too small, the algorithm may never find the corner, whereas a large window will yield inaccurate results or even converge to another corner. The window size is typically a fixed parameter, not changed for different image resolutions or distortion levels. Figure 2.1 demonstrates the resulting problem when the window size is chosen inappropriately.

In this chapter we present an adaptive version of `cornerSubPix()`, that changes window size based on the tag appearance in the image frame. Equation 2.3 shows how the size of the resize window is determined. The algorithm reprojects all target features into the image frame, and for each feature finds its nearest neighbor in the image frame. We then use that information to scale the search window, according to Equation 2.3.

$$w_{\mathbf{x}_\text{t}} = \min_{\boldsymbol{x}_\text{t}^* \in \mathcal{Q}} s \cdot (\boldsymbol{\pi}(\boldsymbol{T}_\text{tar} \cdot \boldsymbol{x}_\text{t}, \boldsymbol{i}) - \boldsymbol{\pi}(\boldsymbol{T}_\text{tar} \cdot \boldsymbol{x}_\text{t}^*, \boldsymbol{i})) \tag{2.3}$$

Here s is a user-defined scalar that can be used to make the window either bigger or smaller in the target frame. $\boldsymbol{x}^*$ is a target coordinate within $\mathcal{Q}$, the subspace of $\mathbb{R}^3$ that consists of all possible feature coordinates in the target frame.

2.4.4.2 Symmetry-Based Refinement

The authors of [267] first proposed refining calibration target detections by optimizing a symmetry-based cost function. The original cost function is shown in Equation 2.4.

$$C_{\text{sym}}(\mathbf{H}) = \sum_{k=1}^{n} (I(\mathbf{H}(\boldsymbol{s}_k)) - I(\mathbf{H}(-\boldsymbol{s}_k)))^2 \tag{2.4}$$

Here $\mathbf{H}$ is a homography from the target frame to the image frame, with the center in the target frame being the feature that is being refined. $\boldsymbol{s}_k$ and $-\boldsymbol{s}_k$ are two samples in the target frame, that are defined such that the origin corresponds to the feature location. The authors then optimize $\mathbf{H}$ with the Levenberg-Marquardt method in order to minimize C_{sym}. The authors show that using deltille grids, checkerboard, and their custom target, this method is the most precise refinement method. Their custom target consists of a single AprilTag to determine a homography between the target plane and image plane, which is surrounded by a number of star-shaped feature points. The homography determined using the single AprilTag is used to make an initial guess for the homography for each feature; it is therefore important that the camera being calibrated behaves somewhat close to a pinhole camera.

As a result, this approach is effective for low-distortion to medium-distortion as the projection is close to a homography. For true ultra-wide lenses, however, the proposed approach fails for two reasons: (1) the single AprilTag is not detected, and (2) the homography assumption causes the refinement procedure to fail.

We propose an adapted version that optimizes a modified symmetry-based objective function, shown in Equation 2.5.

$$\begin{aligned} C_{\text{sym}}(\boldsymbol{x}_{\text{t}}) = \sum_{k=1}^{n} (&I(\boldsymbol{\pi}(\boldsymbol{T}_{\text{tar}} \cdot (\boldsymbol{x}_{\text{t}} + \boldsymbol{s}_k), \boldsymbol{i})) \\ &- I(\boldsymbol{\pi}(\boldsymbol{T}_{\text{tar}} \cdot (\boldsymbol{x}_{\text{t}} - \boldsymbol{s}_k), \boldsymbol{i})))^2 \end{aligned} \tag{2.5}$$

Here $\boldsymbol{x}_{\text{t}}$ is the location of the feature in target space, that is transformed to the camera frame using $\boldsymbol{T}_{\text{tar}}$, and projected to the image frame using the projection function $\boldsymbol{\pi}$. All steps in this equation are differentiable, which we use to optimize the feature location in the target frame directly. With this we leverage the intermediate model to retrieve higher quality symmetric samples, and a better initial estimate.

2.5 Results

2.5.1 Evaluation Metrics

In this section, we evaluate a number of metrics in order to compare TartanCalib against other state-of-the-art approaches for feature detection and geometric camera calibration. Evaluating geometric camera calibration is challenging as no ground truth exists for feature

detections and camera geometry. A metric that is traditionally used for geometric camera calibration is reprojection error, especially when different models or optimization routines are compared. This is only an appropriate metric when the features are the same, as a different feature distribution yields an entirely different optimization problem. We will see that TartanCalib brings more detected features in the entire image, especially in the highly distorted regions, fulfilling its design purpose. Reprojection error could be higher if the camera model is not complex enough to fit all the detected features in the distorted regions. This observation indicates that using reprojection error as the metric is not appropriate. Nevertheless, the reprojection error is presented for reference, while we use other quantitative metrics to show that the accuracy and coverage of the detected features from TartanCalib are of higher quality.

2.5.2 Experimental Setup

To evaluate TartanCalib, its performance is evaluated in two camera configurations: (1) an aerial robot equipped with multiple cameras with ultra-wide Lensagon BF5M fisheye lenses (Figure 2.3, 195° FOV), and (2) a GoPro Hero 8 with a $122.6^\circ \times 149.2^\circ$ FOV. For the GoPro, data is collected using its proprietary software, whereas the ROS gscam drivers are used to extract images from the cameras on the aerial robot. The Lensagon BF5M fisheye lens is installed on a Leopard Imaging LI-IMX264 camera. This combination will be referred to as the Lensagon BF5M from now on. We use an AprilTag target for calibration experiments. The board presents a 6×6 grid with an $8.8\,\mathrm{cm}$ tag size. For each experiment, 500 images are retrieved per camera by subsampling each dataset.

2.5.3 Feature Detection

The first metric considered is the number of features detected, and their coverage in spherical coordinates. In this section we consider undistortion and target reprojection as methods for retrieving more features, and compare them against state-of-the-art approaches for AprilTag detection.

2.5.3.1 Undistortion

An intuitive way to improve target detection in wide-angle camera calibration is to undistort the image into multiple pinhole projections. This idea is evaluated by iteratively undistorting the image into five pinhole projections. The first plane is pointed in the positive z direction (the principal axis) of the camera frame, while the other projections are pointed at a polar angle of 90°. The azimuth angles start from zero and are incremented by 90° for each plane. The five planes together form a cube. In correspondence with Figure 2.2, the undistorted pinhole projections are used to iteratively improve the camera model.

To quantify the effect of undistortion, the previously collected images for Lensagon BF5M are further sampled into a subset that contains 100 frames. TartanCalib is then used

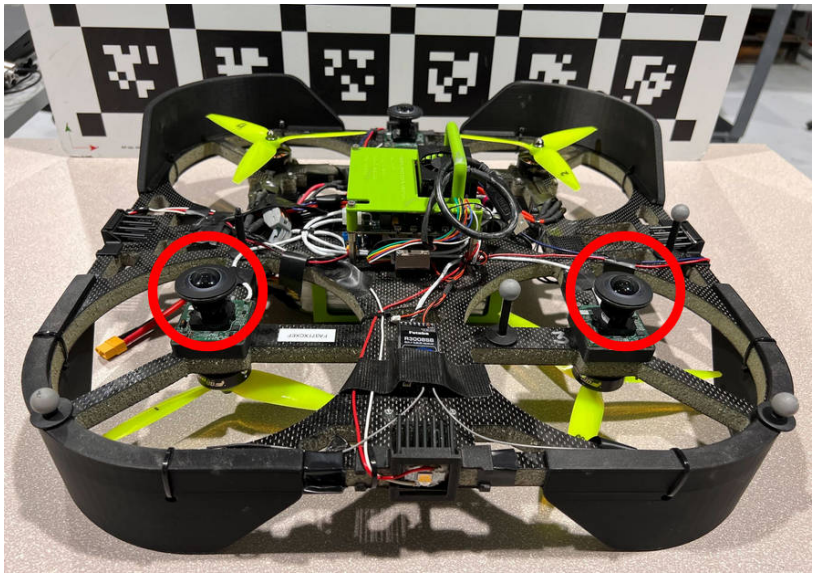


Figure 2.3: The aerial robot used in the experiment. It has three identical sets of cameras and fisheye lenses. Every camera uses a Lensagon BF5M2023S23C Lens (195° FOV), with a Leopard Imaging LI-IMX264-MIPI-M12 sensor. Red circles: the two fisheye lenses that are used in Section 2.5.7 to form a stereo pair.

to iteratively undistort the wide-angle images into pinhole projections. In this experiment undistortion yielded an additional 5 %-10 % of features.

2.5.3.2 Reprojection

While 5 %-10 % more features is a significant improvement, it does not resolve the issue of too few features at the edge of the lens. Reprojecting the features from the target frame into the image frame, as described in Section 2.4.2, is therefore investigated as an alternative solution. For this experiment the dataset described in Section 2.5.2 is used, with 500 images for each evaluation. Figure 2.4 shows the distribution of detected features by the various methods considered. TartanCalib is compared against other state-of-the-art AprilTag detectors, being: (1) Kalibr, (2) Deltille [104], (3) AprilTag3 (AT3) [224, 327], a C++ implementation of AprilTags, originally written by Dr. Michael Kaess, and (4) the ArUco tag detector [98, 137, 253]. Table 2.1 shows the number of features detected using each method, whereas Table 2.2 shows the number of features detected, sorted by polar angle.

The results show that TartanCalib is the most robust against lens distortion, recording far more features at the edge of the Lensagon BF5M fisheye lens. The results for GoPro (not shown in the figure) show that the numbers of detected features are roughly the same across all models. We attribute this to the fact that it is a far less challenging lens to calibrate with its smaller FOV. The experiments that follow in later sections use reprojection and no undistortion.

Table 2.1: Number of features detected in datasets collected on a GoPro Hero 8 camera and the Lensagon BF5M ultra-wide fisheye lens. The percentages signify the fraction of features detected w.r.t. the theoretical maximum (i.e., number of frames $\times$ number of features).

	GoPro Hero 8	Lensagon BF5M
Deltille	**70,464** (97.87 %)	34,149 (47.43 %)
AprilTag3	45,788 (63.59 %)	29,332 (40.74 %)
AprilTag Kaess	68,708 (95.43 %)	38,456 (53.41 %)
ArUco (OpenCV)	66,092 (91.79 %)	22,364 (31.06 %)
Kalibr	62,383 (86.64 %)	33,433 (46.43 %)
TartanCalib	67,693 (94.01 %)	**54,685** (75.95 %)

Table 2.2: The normalized number of features detected using the BF5M ultra-wide fisheye camera, sorted by polar angle. The results clearly show that TartanCalib detects most features, especially at the edge of the lens.

Normalized Number of Features Detected, Sorted by Polar Angle										
	0°-10°	10°-20°	20°-30°	30°-40°	40°-50°	50°-60°	60°-70°	70°-80°	80°-90°	90°-100°
Kalibr	0.75	0.80	0.87	0.84	0.72	0.56	0.48	0.38	0.19	0.04
Deltille	0.62	0.61	0.63	0.63	0.64	0.64	0.63	0.60	0.52	0.60
AT3	0.75	0.75	0.71	0.63	0.57	0.53	0.46	0.32	0.19	0.09
Kaess	0.78	0.83	0.92	0.91	0.81	0.68	0.63	0.50	0.24	0.05
ArUco	0.89	0.85	0.82	0.63	0.43	0.24	0.10	0.02	0.00	0.00
TartanCalib	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

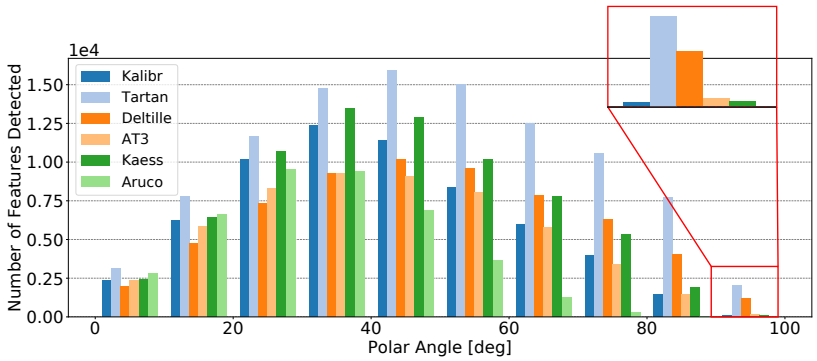


Figure 2.4: AprilTag features detected on the Lensagon BF5M. The results show that especially at the edge of the lens with polar angle approaching 100 degrees, TartanCalib detects significantly more features by reprojecting points from the target frame into the image frame.

2.5.4 Subpixel Refinement

In this section we compare two novel subpixel refinement strategies, and show that symmetry-based refinement is too unstable for grids of AprilTags. The authors of [267] deployed low to medium distortion lenses and tested symmetry-based refinement on deltille grids and checkerboards. Especially in areas of high distortion the refinements appear to be misaligned with the feature.

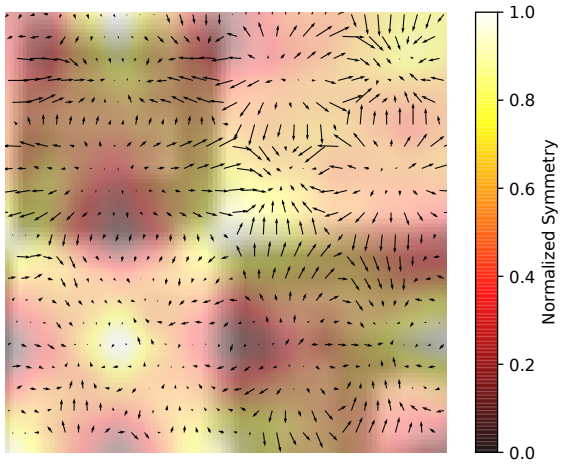


Figure 2.5: This figure shows the measure of symmetry as was proposed in Equation 2.5, on top of an AprilTag corner. The arrows are pointed in the direction of the symmetry gradient to show how an optimizer may behave locally.

Figure 2.5 shows why symmetry is an unsuitable metric for feature refinement with AprilTags. The figure shows that numerous local optima exist, making a high-quality initial estimate a requirement. The initial estimate of the target position in the image frame is possibly several pixels off (as shown in Figure 2.1), and may therefore converge to a local optimum.

The resulting features, therefore, match the model better but do not appear to match the features in the real-world in high-distortion areas. We therefore hypothesize that adaptive `cornerSubPix()` yields more stable features. This hypothesis is studied in more detail in Section 2.5.5.

2.5.5 Comparison on Human-Annotated Features

The closest thing to ground-truth data would be human-annotated feature locations. In this experiment, a subject with no access to the detections annotated perceived feature locations through software. The observations were then compared against TartanCalib with adaptive `cornerSubPix()` and symmetry-based refinement. While a human is unable to detect features at a subpixel level, the experiment remains useful to quantify our observations. All annotated data are available through the project website. Table 2.3 shows that TartanCalib outperforms Kalibr. Symmetry-based refinement outperforms adaptive `cornerSubPix()` in the GoPro dataset, but performs worse on the high-distortion Lensagon BF5M dataset. This shows that symmetry is an unstable metric for feature refinement, as was previously demonstrated in Figure 2.5. For later experiments, we will exclusively use adaptive `cornerSubPix()` unless explicitly stated otherwise.

Table 2.3: The average detection error (px) in feature detection as compared to human-annotated data on 1,440 data points. The results show TartanCalib outperforms Kalibr significantly, and it shows symmetry-based refinement performs slightly better than `cornerSubPix()` on the GoPro dataset, but it performs worse on the high-distortion Lensagon BF5M fisheye lens dataset. This shows that symmetry is an unstable metric for feature refinement.

Camera	Kalibr	TartanCalib	
		Symmetry-Based	Adp. `cornerSubPix()`
Lensagon BF5M	1.038	0.861	**0.815**
GoPro Hero 8	0.926	**0.904**	0.915

Table 2.4: Reprojection error evaluated on the Lensagon BF5M ultra-wide fisheye camera and GoPro Hero 8. DS-None: Double-Sphere model [311], Omni-Radtan: Omnidirectional camera model with Radial-Tangential distortion [303], and the Central Generic Model [18, 267].

Camera	Model	Kalibr Reproj. Err. [px]	Tartan Reproj. Err. [px]
Lensagon BF5M	Central Generic Model	0.3297	**0.2630**
GoPro Hero 8	DS-None	1.0893	**0.7981**
GoPro Hero 8	Omni-Radtan	1.0869	**0.7979**

2.5.6 Reprojection Error

The most widely-used error metric for geometric camera calibration is reprojection error. Given an estimated target pose and camera model, it is possible to reproject the features back into the target frame and compare them against the detected corners (hence reprojection error). As aforementioned, this is not a particularly informative metric as the different methods output substantially different feature distributions (e.g., TartanCalib has far more features at the edge of the lens). Reprojection error is not only a function of the features, but also the ability of the camera model to fit the lens geometry. We perform a simple experiment to show that TartanCalib can improve reprojection error. Table 2.4 shows a substantial improvement when using TartanCalib.

2.5.7 Stereo Depth

Figure 2.6 shows the results of an experiment that was performed to evaluate downstream performance when using pinhole reprojections at the edge of the Lensagon BF5M fisheye lens (see Figure 2.3). We create a stereo pair of virtual pinhole images to show that (1) the images are less distorted using TartanCalib, and (2) the disparity maps generated using the stereo pair are less noisy. The results suggest TartanCalib helps improve downstream tasks.

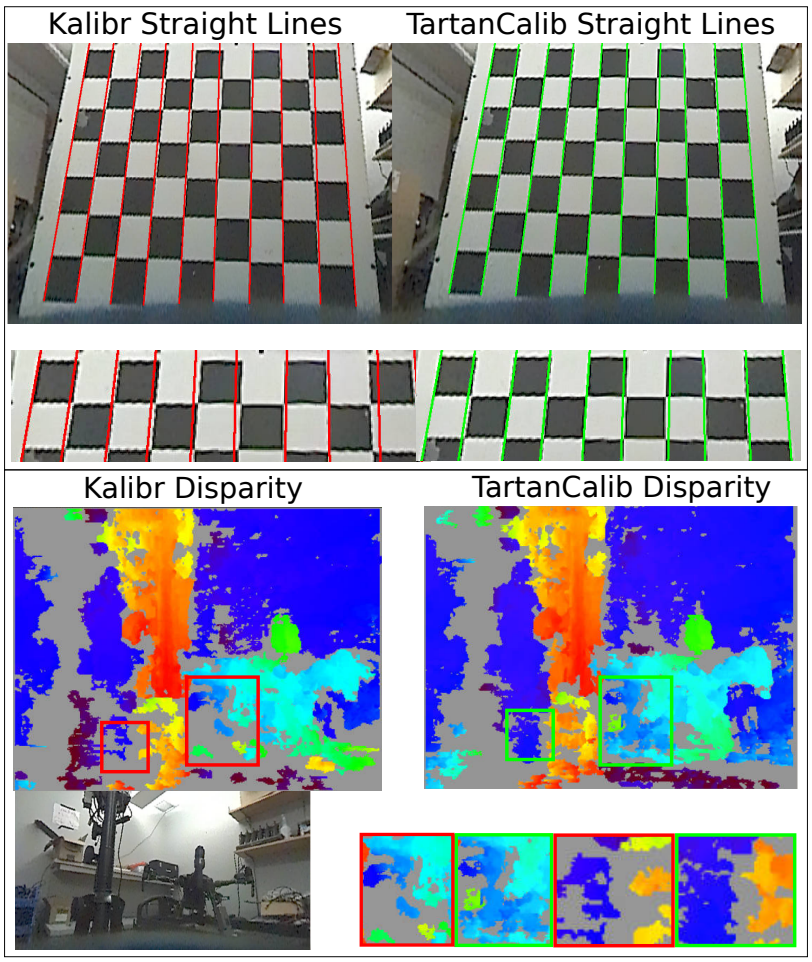


Figure 2.6: This figure shows the results of an experiment for evaluating downstream performance when using pinhole reprojections at the edge of the Lensagon BF5M fisheye lens. We create a stereo pair of virtual pinhole images to show that (1) the images are less distorted using TartanCalib, and (2) the disparity maps generated using the stereo pair are less noisy. This result demonstrates better features detected by TartanCalib help improve downstream tasks.

2.6 Conclusion

In this chapter, we have addressed the problem of geometric wide-angle camera calibration. Previous methods lack features at the edge of the image region, where distortion is strongest. Our approach focuses on retrieving more features at the edge and improving their accuracy. We propose a novel calibration pipeline, TartanCalib, which iteratively improves camera models and leverages intermediate camera models to improve feature detections. Two novel subpixel refinement strategies are proposed that leverage the intermediate model to achieve better accuracy. The results show that symmetry-based refinement is not a stable metric, and

that a simple modification to `cornerSubPix()` yields the best results for high-distortion lenses.

Finally, the entire pipeline is implemented in an open-source easy-to-use toolbox. TartanCalib can be used as an augmentation to a state-of-the-art camera calibration toolbox, e.g. Kalibr, and improves the calibration effectiveness on wide-angle lenses. With its iterative nature, TartanCalib does not require tedious hyperparameter tuning and typically only takes 2-3 times longer to run compared to the non-iterative baseline method. For wide-angle lenses TartanCalib delivers the highest feature coverage publicly available to date.

Chapter 3

MASt3R-SfM: a Fully-Integrated Solution for Unconstrained Structure-from-Motion

ORIGINALLY APPEARED AS

Bardienus P. Duisterhof*, Lojze Zust*, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, Jérome Revaud.
* Equal contribution.
MASt3R-SfM: a Fully-Integrated Solution for Unconstrained Structure-from-Motion [68]. International Conference on 3D Vision (3DV), 2025.

Chapter in Context

TartanCalib assumed a pre-recorded calibration sequence for careful geometric camera calibration. In this chapter, we consider the case of 3D reconstruction and calibration from an arbitrary set of images: structure from motion (SfM). This is especially challenging when only sparse captures with limited overlap are available. In this chapter, we enable SfM using the foundation model MASt3R, which predicts point maps for image pairs. This advances SOLF's *obtainability* by recovering geometry and calibration from less constrained image collections.

Structure-from-Motion (SfM), a task aiming at jointly recovering camera poses and 3D geometry of a scene given a set of images, remains a hard problem with still many open challenges despite decades of significant progress. The traditional solution for SfM consists of a complex pipeline of minimal solvers which tends to propagate errors and fails when images do not sufficiently overlap, have too little motion, etc. Recent methods have attempted to revisit this paradigm, but we empirically show that they fall short of fixing these core issues. In this chapter, we propose instead to build upon a recently released foundation model for 3D vision that can robustly produce local 3D reconstructions and accurate matches. We introduce a low-memory approach to accurately align these local reconstructions in a global coordinate system. We further show that such foundation models can serve as efficient image retrievers without any overhead, reducing the overall complexity from quadratic to linear. Overall, our novel SfM pipeline is simple, scalable, fast and truly unconstrained, i.e.

it can handle any collection of images, ordered or not. Extensive experiments on multiple benchmarks show that our method provides steady performance across diverse settings, especially outperforming existing methods in small- and medium-scale settings.

3.1 Introduction

Structure-from-Motion (SfM) is a long-standing problem of computer vision that aims to estimate the 3D geometry of a scene as well as the parameters of the cameras observing it, given the images from each camera [108]. Since it conveniently provides jointly for cameras and map, it constitutes an essential component for many practical computer vision applications, such as navigation (mapping and visual localization [31, 217, 265]), dense multi-view stereo reconstruction (MVS) [231, 266, 322, 372], novel view synthesis [14, 145, 209], auto-calibration [105] or even archaeology [233, 301].

In reality, SfM is a "needle in a haystack" type of problem, typically involving a highly non-convex objective function with many local minima [307]. Since finding the global minimum in such a landscape is too challenging to be done directly, traditional SfM approaches such as COLMAP [265] have been decomposing the problem as a series (or *pipeline*) of minimal problems, *e.g*. keypoint extraction and matching, relative pose estimation, and incremental reconstruction with triangulation and bundle adjustment. The presence of outliers, such as wrong pixel matches, poses additional challenges and compels existing methods to repeatedly resort to hypothesis formulation and verification at multiple occasions in the pipeline, typically with RAndom SAmple Consensus (RANSAC) or its many flavors [11, 12, 85, 156, 306, 342]. This approach has been the standard for several decades, yet it remains brittle and fails when the input images do not sufficiently overlap, or when motion (*i.e*. translation) between viewpoints is insufficient [31, 268].

Recently, a set of innovative methods propose to revisit SfM in order to alleviate the heavy complexity of the traditional pipeline and solve its shortcomings. VGGSfM [324], for instance, introduces an end-to-end differentiable version of the pipeline, simplifying some of its components. Likewise, detector-free SfM [111] replaces the keypoint extraction and matching step of the classical pipeline with learned components. These changes must, however, be put into perspective, as they do not fundamentally challenge the overall structure of the traditional pipeline. In comparison, FlowMap [275] and Ace-Zero [27] independently propose a radically novel type of approach to solve SfM, which is based on simple first-order gradient descent of a global loss function. Their trick is to train a geometry regressor network during scene optimization as a way to reparameterize and regularize the scene geometry. Unfortunately, this type of approach only works in certain configurations, namely for input images exhibiting high overlap and low illumination variations. Lastly, DUSt3R [157, 335] demonstrates that a single forward pass of a transformer architecture can provide for a good estimate of the geometry and camera parameters of a small two-image scene. These particularly robust estimates can then be stitched together again using simple gradient descent, allowing to relax many of the constraints mentioned earlier. However it yields rather imprecise global SfM reconstructions and does not scale well.

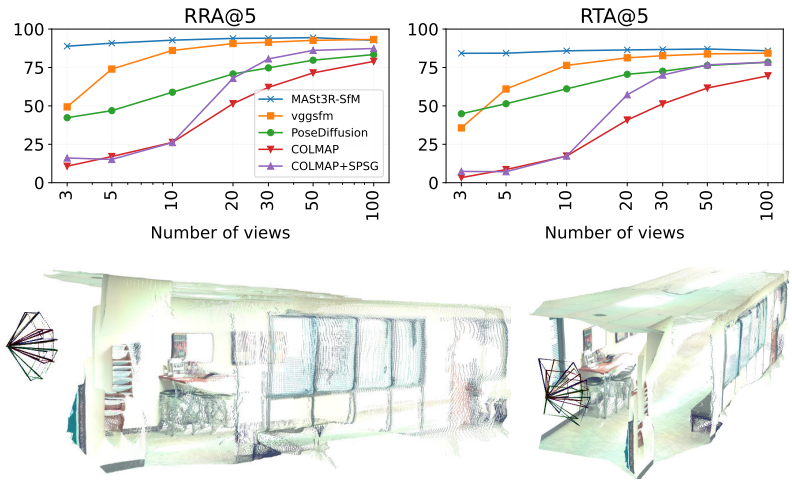


Figure 3.1: **Top**: Relative rotation (RRA) and translation (RTA) accuracies on the CO3Dv2 dataset when varying the number of input views with random subsampling (the more views, the greater their overlap). In contrast to our competitors, MASt3R-SfM offers nearly constant performance on the full range, even for very few views. **Bottom**: MASt3R-SfM also works *without motion*, *i.e.* in purely rotational settings. We show here a reconstruction from 6 views sharing the same optical center.

In this chapter, we propose MASt3R-SfM, a fully-integrated SfM pipeline that can handle completely unconstrained input image collections, *i.e.* ranging from a single view to large-scale scenes, possibly without any camera motion as illustrated in Figure 3.1. We build upon the recently released DUSt3R [335], a foundation model for 3D vision, and more particularly on its recent extension MASt3R that is able to perform local 3D reconstruction and matching in a single forward pass [157]. Since MASt3R is fundamentally limited to processing image pairs, it scales poorly to large image collections. To remedy this, we hijack its frozen encoder to perform fast image retrieval with negligible computational overhead, resulting in a scalable SfM method with quasi-linear complexity in the number of images. Thanks to the robustness of MASt3R to outliers, the proposed method is able to completely get rid of RANSAC. The SfM optimization is carried out in two successive gradient descents based on frozen local reconstructions output by MASt3R: first, using a matching loss in 3D space; then with a 2D reprojection loss to refine the previous estimate. Interestingly, our method goes beyond structure-from-motion, as it works even when there is *no motion* (*i.e.* purely rotational case), as illustrated in Figure 3.1.

In summary, we make three main contributions. First, we propose MASt3R-SfM, a full-fledged SfM pipeline able to process unconstrained image collections. To achieve linear complexity in the number of images, we show as a second contribution how the encoder from MASt3R can be exploited for large-scale image retrieval. Note that our entire SfM pipeline is training-free, provided an off-the-shelf MASt3R checkpoint. Lastly, we conduct extensive benchmarking on a diverse set of datasets, showing that existing approaches are still prone to failure in small-scale settings, despite significant progress. In comparison,

MASt3R-SfM demonstrates state-of-the-art performance in a wide range of conditions, as illustrated in Figure 3.1.

3.2 Related Works

Traditional SfM. At the core of Structure-from-Motion (SfM) lies matching and Bundle Adjustment (BA). Matching, *i.e*. the task of finding pixel correspondences across different images observing the same 3D points, has been extensively studied in the past decades, beginning from handcrafted keypoints [16, 197, 254] and more recently being surpassed by data-driven strategies [38, 58, 76, 77, 135, 251, 260, 287, 332]. Matching is critical for SfM, since it builds the basis to formulate a loss function to minimize during BA. BA itself aims at minimizing reprojection errors for the correspondences extracted during the matching phase by jointly optimizing the positions of 3D points and camera parameters. It is usually expressed as a non-linear least squares problem [3], known to be brittle in the presence of outliers and prone to fall into suboptimal local minima if not provided with a good initialization [2, 276]. For all these reasons, traditional SfM pipelines like COLMAP are heavily handcrafted in practice [111, 182, 265]. By triangulating 3D points to provide an initial estimate for BA, they incrementally build a scene, adding images one by one by formulating hypotheses and discarding the ones that are not verified by the current scene state. Due to the large number of outliers, and the fact that the structure of the pipeline tends to propagate errors rather than fix them, robust estimators like RANSAC are extensively used for relative pose estimation, keypoint track construction and multi-view triangulation [265].

SfM revisited. There has been a recent surge of methods aiming to simplify or even completely revisit the traditional SfM pipeline [27, 111, 275, 324, 335]. The recently proposed FlowMap and Ace-Zero, for instance, both rely on the idea of training a regressor network at test time. In the case of FlowMap [275], this network predicts depthmaps, while for Ace-Zero [27] it regresses dense 3D scene coordinates. While this type of approach is appealing, it raises several problems such as scaling poorly and depending on many off-the-shelf components for FlowMap. Most importantly, both methods only apply to constrained settings where the input image collections offer enough uniformity and continuity in terms of viewpoints and illuminations. This is because the regressor network is only able to propagate information incrementally from one image to other tightly similar images. As a result, they cannot process unordered image collections with large viewpoint and illumination disparities. On the other hand, VGGSfM, Detector-Free SfM (DF-SfM) and DUSt3R cast the SfM problem in a more traditional manner by relying on trained neural components that are kept frozen at optimization time. VGGSfM [324], for its part, essentially manages to train end-to-end all components of the traditional SfM pipeline but still piggybacks itself onto handcrafted solvers for initializing keypoints, cameras and to triangulate 3D points. As a result, it suffers from the same fundamental issues as traditional SfM, *e.g*. it struggles when there are few views or little camera motion. Likewise, DF-SfM [111] improves for

texture-less scenes thanks to relying on trainable dense pairwise matchers, but sticks to the overall COLMAP pipeline. Finally, DUSt3R [335] is a foundation model for 3D vision that essentially decomposes SfM into two steps: local reconstruction for every image pair in the form of pointmaps, and global alignment of all pointmaps in world coordinates. While the optimization appears considerably simpler than for previous approaches (*i.e*. not relying on external modules, and carried out by minimizing a global loss with first-order gradient descent), it unfortunately yields rather imprecise estimates and does not scale well. Its recent extension MASt3R [157] adds pixel matching capabilities and improved pointmap regression, but does not address the SfM problem. In this chapter, we fill this gap and present a fully-integrated SfM pipeline based on MASt3R that is both precise and scalable.

Image Retrieval for SfM. Since matching is essentially considering pairs in traditional SfM, it has a quadratic complexity which becomes prohibitive for large image collections. Several SfM approaches have proposed to leverage faster, although less precise, image comparison techniques relying on comparing global image descriptors, *e.g*. AP-GeM [250] for Kapture [124] or by distilling NetVLAD [4] for HLoc [259]. The idea is to cascade image matching in two steps: first, a coarse but fast comparison is carried out between all pairs (usually by computing the similarity between global image descriptors), and for image pairs that are similar enough, a second stage of costly keypoint matching is then carried out. This is arguably much faster and more scalable. In this chapter, we adopt the same strategy, but instead of relying on an external off-the-shelf module, we show that we can simply exploit the frozen MASt3R's encoder for this purpose, considering the token features as local features and directly performing efficient retrieval with Aggregated Selective Match Kernels (ASMK) [304].

3.3 Preliminaries

The proposed method builds on the recently introduced MASt3R model which, given two input images $I^n, I^m \in \mathbb{R}^{H \times W \times 3}$, performs joint *local 3D reconstruction* and *pixel-wise matching* [157]. We assume here for simplicity that all images have the same pixel resolution $W \times H$, but of course they can differ in practice. In the next section, we show how to leverage this powerful *local* predictor for achieving large-scale *global* 3D reconstruction.

At a high level, MASt3R can be viewed as a function

$$f(I^n, I^m) \equiv \text{Dec}(\text{Enc}(I^n), \text{Enc}(I^m)),$$

where $\text{Enc}(I) \rightarrow F$ denotes the Siamese ViT encoder that represents image I as a feature map of dimension d, width w and height h, $F \in \mathbb{R}^{h \times w \times d}$, and $\text{Dec}(F^n, F^m)$ denotes twin ViT decoders that regress pixel-wise pointmaps X and local features D for each image, as well as their respective corresponding confidence maps. These outputs intrinsically contain rich geometric information from the scene, to the extent that camera intrinsics and (metric) depthmaps can straightforwardly be recovered from the pointmap, see [335] for

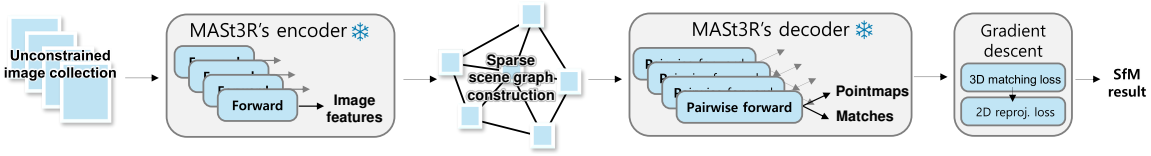


Figure 3.2: **Overview of the proposed MASt3R-SfM method**. Given an unconstrained image collection, possibly small (1 image) or large (> 1000 images), we start by computing a sparse scene graph using efficient image retrieval techniques given a frozen MASt3R's per-image features. We then compute local 3D reconstruction and matches for each edge using again a frozen MASt3R's decoder. Global optimization proceeds with gradient descent of a matching loss in 3D space, followed by refinement in terms of 2D reprojection error.

details. Likewise, we can recover sparse correspondences (or *matches*) by application of the fastNN algorithm described in [157] with the regressed local feature maps D^n, D^m. More specifically, the fast NN searches for a subset of reciprocal correspondences from two feature maps D^n and D^m by initializing seeds on a regular pixel grid and iteratively converging to mutual correspondences. We denote these correspondences between I^n and I^m as $\mathcal{M}^{n,m} = \{y_c^n \leftrightarrow y_c^m\}_{c=1..|\mathcal{M}^{n,m}|}$, where $y_c^n, y_c^m \in \mathbb{N}^2$ denotes a pair of matching pixels.

3.4 Proposed Method

Given an unordered collection of N images $\mathcal{V} = \{I^n\}_{1 \leq n \leq N}$ of a static 3D scene, captured with respective cameras $\mathcal{K}_n = (K_n, P_n)$, where $K_n \in R^{3\times3}$ denotes the intrinsic parameters (*i.e.* calibration in terms of focal length and principal point) and $P_n \in R^{4\times4}$ its world-to-camera pose, our goal is to recover all camera parameters $\{\mathcal{K}_n\}$ as well as the underlying 3D scene geometry $\{X^n\}$, with $X^n \in \mathbb{R}^{W\times H\times3}$ a pointmap relating each pixel $y = (i, j) \in \mathbb{N}^2$ from I^n to its corresponding 3D point $X^n_{i,j}$ in the scene expressed in a world coordinate system.

Overview. We present a novel large-scale 3D reconstruction approach consisting of four steps outlined in Figure 3.2. First, we construct a co-visibility graph using efficient and scalable image retrieval techniques. Edges of this graph connect pairs of likely-overlapping images. Second, we perform pairwise local 3D reconstruction and matching using MASt3R for each edge of this graph. Third, we coarsely align every local pointmap in the same world coordinate system using gradient descent with a matching loss in 3D space. This serves as initialization for the fourth step, wherein we perform a second stage of global optimization, this time minimizing 2D pixel reprojection errors. We detail each step below.

3.4.1 Scene graph

We first aim at spatially relating scene objects seen under different viewpoints. Traditional SfM methods use efficient and scalable keypoint matching for that purpose, thereby building point tracks spanning multiple images. However, MASt3R is originally a pairwise image matcher, which has quadratic complexity in the number N of images and therefore becomes infeasible for large collections if done naively.

Sparse scene graph. Instead, we wish to only feed a small but sufficient subset of all possible pairs to MASt3R, whose structure forms a scene graph $\mathcal{G}$. Formally, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph where each vertex $I \in \mathcal{V}$ is an image, and each edge $e = (n, m) \in \mathcal{E}$ is an undirected connection between two likely-overlapping images I^n and I^m. Importantly, $\mathcal{G}$ must have a single connected component, *i.e.* all images must (perhaps indirectly) be linked together.

Image retrieval. To select the right subset of pairs, we rely on a scalable pairwise image matcher $h(I^n, I^m) \mapsto s$, able to predict the approximate co-visibility score $s \in [0, 1]$ between two images I^n and I^m. While any off-the-shelf image retriever can in theory do, we propose to leverage MASt3R's encoder $\text{Enc}(\cdot)$. Indeed, our findings are that the encoder, due to its role of laying foundations for the decoder, is implicitly trained for image matching (see Section 3.5.3). To that aim, we adopt the ASMK (Aggregated Selective Match Kernels) image retrieval method [304] considering the token features output by the encoder as local features. ASMK has shown excellent performance for retrieval, especially without requiring any spatial verification. In a nutshell, we consider the output F of the encoder as a bag of local features, apply feature whitening, quantize them according to a codebook previously obtained by k-means clustering, then aggregate and binarize the residuals for each codebook element, thus yielding high-dimensional sparse binary representations. The ASMK similarity between two image representations can be efficiently computed by summing a small kernel function on binary representations over the common codebook elements. Note that this method is training-free, only requiring to compute the whitening matrix and the codebook once from a representative set of features. We also try learning a small projector on top of the encoder features following the HOW approach [305], but this leads to similar performances. We refer to the supplementary material of the original paper [68] for more details. The output from the retrieval step is a similarity matrix $S \in [0, 1]^{N \times N}$.

Graph construction. To get a small number of pairs while still ensuring a single connected component, we build the graph $\mathcal{G}$ as follows. We first select a fixed number N_a of *key images* (or keyframes) using farthest point sampling (FPS) [80] based on S. These keyframes constitute the core set of nodes and are densely connected together. All remaining images are then connected to their closest keyframe as well as their k nearest neighbors according to S. Such a graph comprises $O(N_a^2 + (k+1)N) = O(N) \ll O(N^2)$ edges, which is linear in the number of images N. We typically use $N_a = 20$ and $k = 10$. Note that, while the

retrieval step has quadratic complexity in theory, it is extremely fast and scalable in practice, so we ignore it and report quasi-linear complexity overall.

3.4.2 Local reconstruction

As indicated in Section 3.3, we run the inference of MASt3R for every pair $e = (n, m) \in \mathcal{E}$, yielding raw pointmaps and sparse pixel matches $\mathcal{M}^{n,m}$. Since MASt3R is order-dependent in terms of its input, we define $\mathcal{M}^{n,m}$ as the union of correspondences obtained by running both $f(I^n, I^m)$ and $f(I^m, I^n)$. Doing so, we also obtain pointmaps $X^{n,n}, X^{n,m}, X^{m,n}$ and $X^{m,m}$, where $X^{n,m} \in \mathbb{R}^{H\times W\times 3}$ denotes a 2D-to-3D mapping from pixels of image I^n to 3D points in the coordinate system of image I^m. Since the encoder features $\{F^n\}_{n=1..N}$ have already been extracted and cached during scene graph construction (Section 3.4.1), we only need to run the ViT decoder Dec(), which substantially saves time and compute.

Canonical pointmaps. We wish to estimate an initial depthmap Z^n and camera intrinsics K_n for each image I^n. These can be easily recovered from a raw pointmap $X^{n,n}$ as demonstrated in [335], but note that each pair $(n, \cdot)$ or $(\cdot, n) \in \mathcal{E}$ would yield its own estimate of $X^{n,n}$. To average out regression imprecision, we hence aggregate these copycat pointmaps into a canonical pointmap $\tilde{X}^n$. Let $\mathcal{E}^n = \{e | e \in \mathcal{E} \wedge n \in e\}$ be the set of all edges connected to image I^n. For each edge $e \in \mathcal{E}^n$, we have a different estimate of $X^{n,n}$ and its respective confidence maps $C^{n,n}$, which we will denote as $X^{n,e}$ and $C^{n,e}$ in the following. We compute the canonical pointmap as a simple per-pixel weighted average of all estimates:

$$\tilde{X}^n_{i,j} = \frac{\sum_{e\in\mathcal{E}^n} C^{n,e}_{i,j} X^{n,e}_{i,j}}{\sum_{e\in\mathcal{E}^n} C^{n,e}_{i,j}}. \tag{3.1}$$

From it, we then recover the canonical depthmap $\tilde{Z}^n = \tilde{X}^n_{:,:,3}$ and the focal length using the Weiszfeld algorithm [335]:

$$f^* = \arg\min_f \sum_{i,j} \left\| \left(i - \frac{W}{2}, j - \frac{H}{2}\right) - f\left(\frac{\tilde{X}^n_{i,j,1}}{\tilde{X}^n_{i,j,3}}, \frac{\tilde{X}^n_{i,j,2}}{\tilde{X}^n_{i,j,3}}\right) \right\|, \tag{3.2}$$

which, assuming a centered principal point and square pixels, yields the canonical intrinsics $\tilde{K}^n$. In this chapter, we assume a pinhole camera model without lens distortion, but our approach could be extended to different camera types.

Constrained pointmaps. Camera intrinsics K, extrinsics P and depthmaps Z will serve as basic ingredients (or rather, optimization variables; Figure 3.3) for the global reconstruction phase. Let $\pi_n : \mathbb{R}^3 \mapsto \mathbb{R}^2$ denote the reprojection function onto the camera screen of I^n, *i.e.* $\pi_n(x) = K_n P_n \sigma_n x$ for a 3D point $x \in \mathbb{R}^3$ ($\sigma_n > 0$ is a per-camera scale factor, *i.e.* we use scaled rigid transformations). To ensure that pointmaps perfectly satisfy the pinhole projective model (they are normally over-parameterized), we define a *constrained pointmap* $\chi^n \in \mathbb{R}^{H\times W\times 3}$ explicitly as a function of K_n, P_n, σ_n and Z^n. Formally, the

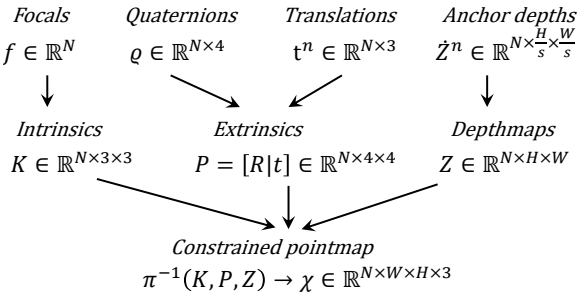


Figure 3.3: **Factor graph for MASt3R-SfM**. Free variables on the top row serve to construct the constrained pointmap χ, which follows the pinhole camera model by design and onto which the loss functions from Equations 3.3 and 3.4 are defined.

3D point $\chi^n_{i,j}$ seen at pixel (i, j) of image I^n is defined using inverse reprojection as $\chi^n_{i,j} = \pi_n^{-1}(\sigma_n, K_n, P_n, Z^n_{i,j}) = {}^1\!/\!_{\sigma_n} P_n^{-1} K_n^{-1} Z^n_{i,j} \left[i, j, 1\right]^\top$.

3.4.3 Coarse alignment

Recently, DUSt3R [335] introduced a global alignment procedure aiming to rigidly move dense pointmaps in a world coordinate system based on pairwise relationships between them. In this chapter, we simplify and improve this procedure by taking advantage of pixel correspondences, thereby reducing the overall number of parameters and its memory and computational footprint.

Specifically, we look for the scaled rigid transformations σ^*, P^* of every canonical pointmap $\chi = \pi^{-1}(\sigma, \tilde{K}, P, \tilde{Z})$ (*i.e.* fixing intrinsics $K = \tilde{K}$ and depth $Z = \tilde{Z}$ to their canonical values) such that any pair of matching 3D points gets as close as possible:

$$\sigma^*, P^* = \underset{\sigma, P}{\arg\min} \sum_{\substack{c \in \mathcal{M}^{n,m} \\ (n,m)\in\mathcal{E}}} q_c \left\| \chi^n_c - \chi^m_c \right\|^{\lambda_1}, \tag{3.3}$$

where c denotes the matching pixels in each respective image by a slight abuse of notation. In contrast to the global alignment procedure in DUSt3R, this minimization only applies to sparse pixel correspondences $y^n_c \leftrightarrow y^m_c$ weighted by their respective confidence q_c (also output by MASt3R). To avoid degenerate solutions, we enforce $\min_n \sigma_n = 1$ by reparameterizing $\sigma_n = \sigma'_n / (min_n \sigma'_n)$. We minimize this objective using Adam [149] for a fixed number ν_1 of iterations.

3.4.4 Refinement

Coarse alignment converges well and fast in practice, but restricts itself to rigid motion of canonical pointmaps. Unfortunately, pointmaps are bound to be noisy due to depth ambiguities during local reconstruction. To further refine cameras and scene geometry, we

thus perform a second round of global optimization akin to bundle adjustment [307] with gradient descent for ν_2 iterations and starting from the coarse solution σ^*, P^* obtained from Equation 3.3. In other words, we minimize the 2D reprojection error of 3D points in all cameras:

$$Z^*, K^*, P^*, \sigma^* = \underset{Z,K,P,\sigma}{\arg\min}\, \mathcal{L}_2, \text{ with} \tag{3.4}$$
$$\mathcal{L}_2 = \sum_{\substack{c \in \mathcal{M}^{n,m} \\ (n,m) \in \mathcal{E}}} q_c \left[\rho \left(y_c^n - \pi_n \left(\chi_c^m \right) \right) + \rho \left(y_c^m - \pi_m \left(\chi_c^n \right) \right) \right],$$

with $\rho : \mathbb{R}^2 \mapsto \mathbb{R}^+$ a robust error function able to deal with potential outliers among all extracted correspondences. We typically set $\rho(x) = \|x\|^{\lambda_2}$ with $0 < \lambda_2 \leq 1$ (*e.g.* λ_2=0.5).

Forming pseudo-tracks. Optimizing Equation (3.4) has little effect, because sparse pixel correspondences $\mathcal{M}^{m,n}$ are rarely *exactly* overlapping across several pairs. As an illustration, two correspondences $y^m_{:,:} \leftrightarrow y^n_{i,j}$ and $y^n_{i+1,j} \leftrightarrow y^l_{:,:}$ from image pairs (m, n) and (n, l) would independently optimize the two 3D points $\chi^n_{i,j}$ and $\chi^n_{i+1,j}$, possibly moving them very far apart despite this being very unlikely as $(i, j) \simeq (i + 1, j)$. Traditional SfM methods resort to forming point tracks, which is relatively straightforward with keypoint-based matching [58, 182, 197, 260, 265]. We propose instead to form pseudo-tracks by creating *anchor points* and rigidly tying together every pixel with its closest anchor point. This way, correspondences that do not overlap exactly are still both tied to the same anchor point with a high probability. Formally, we define anchor points with a regular pixel grid $\dot{y} \in \mathbb{R}^{W/s \times H/s \times 2}$ spaced by δ pixels:

$$\dot{y}_{u,v} = \left(u\delta + \frac{\delta}{2}, v\delta + \frac{\delta}{2} \right). \tag{3.5}$$

We then tie each pixel (i, j) in I^n with its closest anchor $\dot{y}_{u,v}$ at coordinate $(u, v) = (\lfloor i/\delta \rfloor, \lfloor j/\delta \rfloor)$. Concretely, we simply index the depth value at pixel (i, j) to the depth value $\dot{Z}_{u,v}$ of its anchor point, *i.e.* we define $Z_{i,j} = o_{i,j} \dot{Z}_{u,v}$ where $o_{i,j} = \tilde{Z}_{i,j} / \tilde{Z}_{u,v}$ is a constant relative depth offset calculated at initialization from the canonical depthmap $\tilde{Z}$. Here, we make the assumption that canonical depthmaps are locally accurate. All in all, optimizing a depthmap $Z^n \in \mathbb{R}^{W \times H}$ thus only comes down to optimizing a reduced set of anchor depth values $\dot{Z}^n \in \mathbb{R}^{W/\delta \times H/\delta}$ (*e.g.* reduced by a factor of 64 if $\delta = 8$).

3.5 Experimental Results

After presenting the datasets and metrics, we extensively compare our approach with state-of-the-art SfM methods in diverse conditions. We finally present several ablations.

3.5.1 Experimental setup

We use the publicly available MASt3R checkpoint for our experiments, which we do *not* fine-tune unless otherwise mentioned. When building the sparse scene graph in Section 3.4.1, we use $N_a = 20$ anchor images and $k = 10$ non-anchor nearest neighbors. We use the same grid spacing of $\delta = 8$ pixels for extracting sparse correspondences with FastNN (Section 3.4.2) and defining anchor points (Section 3.4.4). For the two gradient descents, we use the Adam optimizer [149] with a learning rate of 0.07 (resp. 0.014) for $\nu_1 = 300$ iterations and $\lambda_1 = 1.5$ (resp. $\nu_2 = 300$ and $\lambda_2 = 0.5$) for the coarse (resp. refinement) optimization, each time with a cosine learning rate schedule and without weight decay. Unless otherwise mentioned we assume shared intrinsics and optimize a shared per-scene focal parameter for all cameras.

Datasets. To showcase the robustness of our approach, we experiment in different conditions representative of diverse experimental setups (video or unordered image collections, simple or complex scenes, outdoor, indoor or object-centric, *etc*.). Namely, we employ Tanks&Temples [151] (T&T), a 3D reconstruction dataset comprising 21 scenes ranging from 151 to 1106 images; ETH3D [268], a multi-view stereo dataset with 13 scenes for which ground-truth is available; CO3Dv2 [247], an object-centric dataset for multi-view pose estimation; and RealEstate10K [401], MIP-360 [14] and LLFF [208], three datasets for novel view synthesis.

Evaluation metrics. For simplicity, we evaluate all methods w.r.t. ground-truth camera poses. For Tanks&Temples where it is not provided, we make a pseudo ground-truth with COLMAP [265] using all frames. Even though this is not perfect, COLMAP is known to be reliable in conditions where there is a large number of frames with high overlap. We evaluate the average translation error (ATE) as in FlowMap [275], *i.e*. we align estimated camera positions to ground-truth ones with Procrustes [200] and report an average normalized error. We ignore unregistered cameras when doing Procrustes, which favors methods that can reject hard images (such as COLMAP [265] or VGGSfM [324]). Note that our method always outputs a pose estimate for all cameras by design, thus negatively impacting our results with this metric. We also report the relative rotation and translation accuracies (resp. RRA@τ and RTA@τ, where τ indicates the threshold in degrees), computed at the pairwise level and averaged over all image pairs [323]. Similarly, the mean Average Accuracy (mAA)@τ is defined as the area under the curve of the angular differences at min(RRA@τ, RTA@τ). Finally, we report the successful registration rate as a percentage, denoted as Reg. When reported at the dataset level, metrics are averaged over all scenes.

Table 3.1: **Results on Tanks&Temples** in terms of ATE and overall registration rate (Reg.). For easier readability, we color-code ATE results as a linear gradient between worst and best ATE for a given dataset or split; and Reg results with linear gradient between 0% and 100%. **Top**: impact of the number of input views, regularly sampled from the full set. 'N/A' indicates that at least one scene did not converge. **Bottom**: ATE↓ on different datasets with the arbitrary splits defined in FlowMap [275].

Method	25 views		50 views		100 views		200 views		full	
	ATE↓	Reg.↑	ATE↓	Reg.↑	ATE↓	Reg.↑	ATE↓	Reg.↑	ATE↓	Reg.↑
COLMAP [265]	0.03840	44.4	0.02920	60.5	0.02640	85.7	0.01880	97.0	-	-
ACE-Zero [27]	0.11160	100.0	0.07130	100.0	0.03980	100.0	0.01870	100.0	0.01520	100.0
FlowMap [275]	0.10700	100.0	0.07310	100.0	0.04460	100.0	0.02420	100.0	N/A	66.7
VGGSfM [324]	0.05800	96.2	0.03460	98.7	0.02900	98.5	N/A	47.6	N/A	0.0
DF-SfM [111]	0.08110	99.4	0.04120	100.0	0.02710	99.9	N/A	33.3	N/A	76.2
MASt3R-SfM	0.03360	100.0	0.02610	100.0	0.01680	100.0	0.01300	100.0	0.01060	100.0

Method	MIP-360	LLFF	T&T	CO3Dv2
NoPE-NeRF [22]	0.04429	0.03920	0.03709	0.03648
DROID-SLAM [299]	0.00017	0.00074	0.00122	0.01728
FlowMap [275]	0.00055	0.00209	0.00124	0.01589
ACE-Zero [27]	0.00173	0.00396	0.00973	0.00520
MASt3R-SfM	0.00079	0.00098	0.00215	0.00538

3.5.2 Comparison with the state of the art

We first evaluate the impact of the amount of overlap between images on the quality of the SfM output for different state-of-the-art methods. To that aim, we choose Tanks&Temples, a standard reconstruction dataset captured with high overlap (originally video frames). We form new splits by regularly subsampling the original images for 25, 50, 100 and 200 frames. Following [275], we report results in terms of Average Translation Error (ATE) against the COLMAP pseudo ground-truth in Table 3.1 (top), computed from the full set of frames and likewise further subsampled. MASt3R-SfM provides nearly constant performance for all ranges, significantly outperforming COLMAP, Ace-Zero, FlowMap and VGGSfM in all settings. Unsurprisingly, the performance of these methods strongly degrades in small-scale settings (or does not even converge on some scenes for COLMAP). On the other hand, we note that FlowMap and VGGSfM crash when dealing with large collections due to insufficient memory despite using 80GB GPUs.

Table 3.2: **Multi-view pose regression on CO3Dv2 [247] and RealEstate10K [401] with 10 random frames**. Parentheses () denote methods that do not report results on the 10-view set; we report their best for comparison (8 views). We distinguish between (a) multi-view and (b) pairwise methods.

	Method	Co3Dv2↑			RealEstate10K↑
		RRA@15	RTA@15	mAA(30)	mAA(30)
(a)	COLMAP+SG [58, 260]	36.1	27.3	25.3	45.2
	PixSfM [182]	33.7	32.9	30.1	49.4
	RelPose [388]	57.1	-	-	-
	PosReg [323]	53.2	49.1	45.0	-
	PoseDiff [323]	80.5	79.8	66.5	48.0
	RelPose++ [177]	(85.5)	-	-	-
	RayDiff [389]	(93.3)	-	-	-
	DUSt3R-GA [335]	**96.2**	86.8	76.7	67.7
	MASt3R-SfM	96.0	**93.1**	**88.0**	**86.8**
(b)	DUSt3R [335]	94.3	88.4	77.2	61.2
	MASt3R [157]	94.6	91.9	81.8	76.4

FlowMap splits. We also report results on the custom splits from the FlowMap paper [275], which concerns 3 additional datasets beyond T&T (LLFF, Mip-360 and CO3Dv2). We point out that, not only do these splits select a *subset* of scenes for each dataset (in detail: 3 scenes from Mip-360, 7 from LLFF, 14 from T&T and 2 from CO3Dv2), but they also select an *arbitrary subset* of consecutive frames in the corresponding scenes. Results in Table 3.1 (bottom) show that our method is achieving better results than NopeNeRF and ACE-Zero, on par with FlowMap overall and slightly worse than DROID-SLAM [299], a method that can only work in video settings. Since we largely outperform FlowMap when using regularly sampled splits, we hypothesize that FlowMap is very sensitive to the input setting.

Multi-view pose estimation. In Figure 3.1 (top), we also compare to various baselines on CO3Dv2 and RealEstate10K, varying the number of input images by random sampling. We follow the PoseDiffusion [323] splits and protocol for comparison purposes. We provide detailed comparisons in Table 3.2 with state-of-the-art multi-view pose estimation methods, whose goal is only to recover camera poses but not the scene geometry. Again, our approach compares favorably to existing methods, particularly when the number of input images is low. Overall, this highlights that MASt3R-SfM is extremely robust to sparse view setups, with its performance not degrading when decreasing the number of views, even for as little as three views.

Table 3.3: **Detailed per-scene translation and rotation accuracies (↑) on ETH-3D.** For clarity, we color-code results with a linear gradient between the worst and best result for a given scene.

Scenes	COLMAP [265]		ACE-Zero [27]		FlowMap [275]		VGGSfM [324]		DF-SfM [111]		**MASt3R-SfM**	
	RRA@5	RTA@5	RRA@5	RTA@5	RRA@5	RTA@5	RRA@5	RTA@5	RRA@5	RTA@5	RRA@5	RTA@5
courtyard	56.3	60.0	4.0	1.9	7.5	3.6	50.5	51.2	80.7	74.8	89.8	64.4
delivery area	34.0	28.1	27.4	1.9	29.4	23.8	22.0	19.6	82.5	82.0	83.1	81.8
electro	53.3	48.5	16.9	7.9	2.5	1.2	79.9	58.6	82.8	81.2	100.0	95.5
facade	92.2	90.0	74.5	64.1	15.7	16.8	57.5	48.7	80.9	82.6	74.3	75.3
kicker	87.3	86.2	26.2	16.8	1.5	1.5	100.0	97.8	93.5	91.0	100.0	100.0
meadow	0.9	0.9	3.8	0.9	3.8	2.9	100.0	96.2	56.2	58.1	58.1	58.1
office	36.9	32.3	0.9	0.0	0.9	1.5	64.9	42.1	71.1	54.5	100.0	98.5
pipes	30.8	28.6	9.9	1.1	6.6	12.1	100.0	97.8	72.5	61.5	100.0	100.0
playground	17.2	18.1	3.8	2.6	2.6	2.8	37.3	40.8	70.5	70.1	100.0	93.6
relief	16.8	16.8	16.8	17.0	6.9	7.7	59.6	57.9	32.9	32.9	34.2	40.2
relief 2	11.8	11.8	7.3	5.6	8.4	2.8	69.9	70.3	40.9	39.1	57.4	76.1
terrace	100.0	100.0	5.5	2.0	33.2	24.1	38.7	29.6	100.0	99.6	100.0	100.0
terrains	100.0	99.5	15.8	4.5	12.3	13.8	70.4	54.9	100.0	91.9	58.2	52.5
Average	49.0	47.8	16.4	9.7	10.1	8.8	65.4	58.9	74.2	70.7	81.2	79.7

Unordered collections. We note that benchmarks in previous experiments were originally acquired using video cameras, and then subsampled into frames. This might introduce biases that may not well represent the general case of unconstrained SfM. We thus experiment on the ETH3D dataset, a photograph dataset, composed of 13 scenes with up to 76 images per scene. Results reported in Table 3.3 show that MASt3R-SfM outperforms all competing approaches by a large margin on average. This is not surprising, as neither ACE-Zero nor FlowMap can handle non-video setups. The fact that COLMAP and VGGSfM also perform relatively poorly indicates a high sensitivity to not having highly overlapping images, meaning that in the end these methods cannot really handle truly unconstrained collections, in spite of some opposite claims [324].

3.5.3 Ablations

We now study the impact of various design choices. All experiments are conducted on the Tanks&Temples dataset regularly subsampled for 200 views per scene.

Scene graph. We evaluate different construction strategies for the scene graph in Table 3.4: 'complete' means that we extract all pairs, 'local window' is a heuristic for video-based collections that connects every frame with its neighboring frames, and 'random' means that we sample random pairs. Except for the 'complete' case, we try to match the number of pairs used in the baseline retrieval strategy.

Slightly better results are achieved with the complete graph, but it is about 10x slower than the retrieval-based graph and not scalable in general. Assuming we use retrieval, we further ablate the scene graph building strategy from the similarity matrix in Table 3.5. As a reminder, it consists of building a small but complete graph of keyframes, and then

Table 3.4: **Ablation of scene graph construction** on Tanks&Temples (200 view subset). See text for details.

Scene Graph	ATE↓	RTA@5↑	RRA@5↑	#Pairs	GPU MEM	Avg. T
Complete	0.01256	75.9	74.8	39,800	29.9 GB	2.2 h
Local window	0.02509	33.1	28.8	2,744	7.6 GB	14.1 min
Random	0.01558	55.2	48.8	2,754	6.9 GB	14.7 min
Retrieval	0.01243	70.9	67.6	2,758	8.4 GB	14.3 min

Table 3.5: **Ablations on Tanks&Temples** (200 view subset). See text for details.

Ablation		ATE↓	RTA@5↑	RRA@5↑	#Pairs
Retrieval	kNN	0.01440	64.1	61.9	3,042
	Keyframes	0.01722	58.1	57.1	740
	Keyframes + kNN	**0.01243**	**70.9**	**67.6**	2,758
Optimization level	Coarse	0.01504	47.4	57.7	2,758
	Fine (w/o depth)	0.01315	67.3	66.9	2758
	Fine	**0.01243**	**70.9**	**67.6**	2,758
Intrinsics	Separate	0.01329	66.9	64.2	2,758
	Shared	**0.01243**	**70.9**	**67.6**	2,758

connecting each image with the closest keyframe and with k nearest non-keyframes. We experiment with using only k-NN with an increased $k = 13$ to compensate for the missing edges, denoted as 'k-NN', or to only use the keyframe graph (*i.e.* $k = 0$), denoted as 'Keyframe'. Overall, we find that combining short-range (k-NN) and long-range (keyframes) connections is important for reaching top performance.

Retrieval with MASt3R. To better assert the effectiveness of our image retrieval strategy alone, we conduct experiments for the task of retrieval-assisted visual localization. We follow the protocol from [157] and retrieve the top-k posed images in the database for each query, extract 2D–3D correspondences and run RANSAC to obtain predicted camera poses. We compare ASMK on MASt3R features to the off-the-shelf retrieval method FIRe [344], also based on ASMK, on the Aachen-Day-Night [261] and InLoc [291] datasets. We report standard visual localization accuracy metrics, *i.e.* the percentages of images successfully localized within error thresholds of (0.25m, 2°) / (0.5m, 5°) / (5m, 10°) and (0.25m, 2°) / (0.5m, 10°) / (1m, 10°) respectively,[1] in Table 3.6. Interestingly, using frozen MASt3R features for retrieval performs on par with FIRe, a state-of-the-art method specifically trained for image retrieval and operating on multi-scale features (bottom row). Our method also reaches competitive performance compared to dedicated visual localization pipelines (top

[1] `https://www.visuallocalization.net/`

Table 3.6: **Comparison of retrieval based on MASt3R features** using ASMK with the state-of-the-art FIRe method when localizing with MASt3R (bottom rows), as well as with other state-of-the-art visual localization methods (top rows).

Method	Aachen-Day-Night↑		InLoc↑	
	Day	Night	DUC1	DUC2
Kapture [124]+R2D2 [251]	**91.3/97.0**/99.5	**78.5**/91.6/**100**	41.4/60.1/73.7	47.3/67.2/73.3
SuperPoint [58]+LightGlue [183]	90.2/96.0/99.4	77.0/91.1/**100**	49.0/68.2/79.3	55.0/74.8/79.4
LoFTR [287]	88.7/95.6/99.0	**78.5**/90.6/99.0	47.5/72.2/84.8	54.2/74.8/85.5
DKM [76]	-	-	51.5/75.3/86.9	63.4/82.4/87.8
MASt3R (FIRe top20)	89.8/96.8/**99.6**	75.9/**92.7**/**100**	**60.6/83.3**/93.4	65.6/86.3/88.5
MASt3R (MASt3R-ASMK top20)	88.7/94.9/98.2	77.5/90.6/97.9	58.1/82.8/**94.4**	**69.5/90.8/92.4**

rows), even setting a new state of the art for InLoc. We refer to the supplementary material of the original paper [68] for further comparisons.

Optimization level. We also study the impact of the coarse optimization and refinement (Table 3.5). As expected, coarse optimization alone, which is somewhat comparable to the global alignment proposed in DUSt3R (except we are using sparse matches and fewer optimization variables), yields significantly less precise pose estimates. In Figure 3.4, we plot the pose accuracy as a function of the number of iterations during coarse optimization and refinement. As expected, refinement, a strongly non-convex bundle-adjustment problem, cannot recover from a random initialization ($\nu_1 = 0$). Good enough poses are typically obtained after $\nu_1 \simeq 250$ iterations of coarse optimization, from which point refinement consistently improves. We also try to perform the optimization without optimizing depth (*i.e*. using frozen canonical depthmaps, which proves useful for purely rotational cases, denoted as 'Fine without depth' in Table 3.5), in which case we observe a smaller impact on the performance, indicating the high-quality of canonical depthmaps output by MASt3R (Section 3.4.2).

Shared intrinsics. We finally evaluate the impact of only optimizing one set of intrinsics for all views ('shared'), which is small, indicating that our method is not sensitive to varying intrinsics.

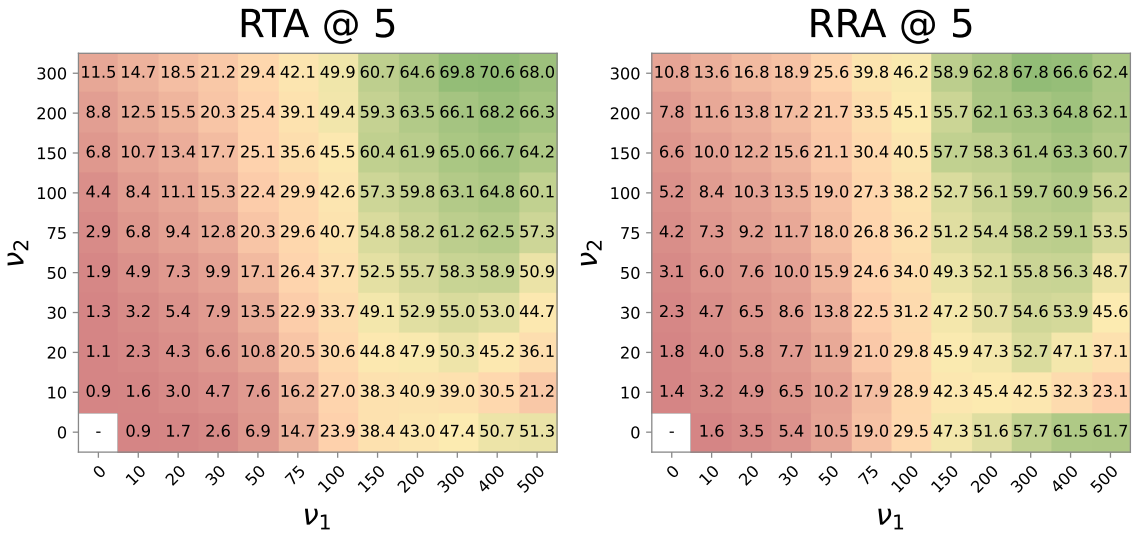


Figure 3.4: **Pose accuracy** (↑) on T&T-200 w.r.t. the number of iterations of the coarse and refinement stages (resp. ν_1 and ν_2).

3.6 Conclusion

This chapter introduced MASt3R-SfM, a comparatively simpler fully-integrated solution for unconstrained SfM. In contrast with prior SfM pipelines, it can handle very small image collections without apparent issues. Thanks to the strong priors encoded in the underlying MASt3R foundation model upon which our approach is based, it can even deal with cases without motion, and does not rely at all on RANSAC, both features that are normally not possible with standard triangulation-based SfM.

Chapter 4

Residual-NeRF: Learning Residual NeRFs for Transparent Object Manipulation

ORIGINALLY APPEARED AS

Bardienus P. Duisterhof, Yuemin Mao, Si Heng Teng, Jeffrey Ichnowski.
Residual-NeRF: Learning Residual NeRFs for Transparent Object Manipulation [72].
IEEE International Conference on Robotics and Automation (ICRA), 2024.

Chapter in Context

Accurate geometric camera calibration from TartanCalib and MASt3R-SfM may be used to obtain a neural radiance field (NeRF) for 3D reconstruction. NeRFs take considerable time to converge and struggle with highly ambiguous transparent objects. This chapter introduces *Residual-NeRF*, which speeds up and improves neural radiance fields for transparent objects. The core insight is to first reconstruct a known background scene, which reduces ambiguity and training time. Within SOLF, reducing reconstruction ambiguity and training time makes transparent-object geometry more *obtainable*.

Transparent objects are ubiquitous in industry, pharmaceuticals, and households. Grasping and manipulating these objects is a significant challenge for robots. Existing methods have difficulty reconstructing complete depth maps for challenging transparent objects, leaving holes in the depth reconstruction. Recent work has shown neural radiance fields (NeRFs) work well for depth perception in scenes with transparent objects, and these depth maps can be used to grasp transparent objects with high accuracy. NeRF-based depth reconstruction can still struggle with especially challenging transparent objects and lighting conditions. In this chapter, we propose Residual-NeRF, a method to improve depth perception and training speed for transparent objects. Robots often operate in the same area, such as a kitchen. By first learning a *background NeRF* of the scene without transparent objects to be

manipulated, we reduce the ambiguity faced by learning the changes with the new object. We propose training two additional networks: a *residual NeRF* learns to infer residual RGB values and densities, and a *Mixnet* learns how to combine background and residual NeRFs. We contribute synthetic and real experiments that suggest Residual-NeRF improves depth perception of transparent objects. The results on synthetic data suggest Residual-NeRF outperforms the baselines with a 46.1 % lower RMSE and a 29.5 % lower MAE. Real-world qualitative experiments suggest Residual-NeRF leads to more robust depth maps with less noise and fewer holes. Website: `https://residual-nerf.github.io`

4.1 Introduction

Dexterous manipulation of transparent objects can enable robots to perform new and impactful tasks in industry, pharmaceuticals, and households. Robots often use depth images of objects to decide what action (e.g., pull, lift, or drop) to perform. However, common depth sensors struggle to capture depth images for arbitrary transparent objects [41, 127, 236, 365]. Learning-based approaches for transparent object depth estimation work well in-distribution, but can struggle to generalize outside their training data [127]. The lack of surface features on transparent objects also makes it challenging to retrieve depth maps using approaches such as COLMAP [265].

Neural radiance fields (NeRFs) [209] are implicit neural network scene representations trained on multiple views of the same scene and capable of state-of-the-art novel view synthesis. NeRF combines a multilayer perceptron (MLP) with classic volumetric rendering techniques to achieve photorealistic novel view synthesis results. The only assumptions in NeRF are those imposed by traditional volumetric rendering techniques [209].

Recently, Dex-NeRF [127] and Evo-NeRF [146] showed that NeRFs are effective at depth perception of transparent objects. However, these methods also showed that NeRFs tend to struggle with particularly challenging transparent objects, such as wine glasses or kitchen foil with challenging light conditions. The challenge with transparent objects originates from a lack of features, and large view-dependent changes in appearance.

Dex-NeRF, while achieving high grasp success rates, was slow to compute. Evo-NeRF [146] significantly sped up NeRF-based grasping by *evolving* NeRFs. In this chapter, we extend on the Evo-NeRF motivations: (1) speed up training to a successful grasp, and (2) learn NeRFs for changing scenes, where small gradients in empty regions can be challenging [146].

We propose Residual-NeRF (Figure 4.1), a method to leverage a strong scene prior to improve depth perception of transparent objects. In many scenarios, the geometry of the robot's work area is mostly static and opaque, e.g., shelves, desks, and tables. Residual-NeRF leverages the static and opaque parts of the scene as a prior, to reduce ambiguity and improve depth perception. Residual-NeRF first learns a *background NeRF* of the entire scene by training on images without the transparent objects present. Residual-NeRF then uses images of the full scene with the transparent objects to learn a *residual NeRF* and a *Mixnet*. The algorithm adaptively mixes both NeRFs for each point along the ray, using a

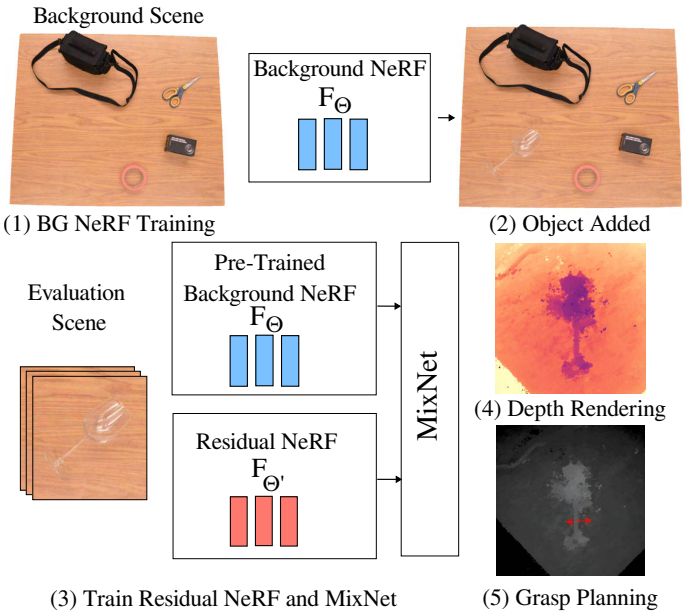


Figure 4.1: Residual-NeRF, a method that leverages mostly static scenes to improve depth perception and speed up training. Residual-NeRF begins by learning a *background NeRF* of the entire scene without transparent objects. Following this, we learn a *residual NeRF* and a *Mixnet* to complement the *background NeRF*.

novel *Mixnet* multi-layer perceptron (MLP). The Mixnet infers a mixing weight between the background and residual NeRF using a location-dependent multi-resolution hash encoding.

We evaluate Residual-NeRF on nine photo-realistic synthetic and three real scenes and compare its performance to other relevant NeRF algorithms. We compare 1) learning speed, 2) depth reconstruction, and 3) robot grasp quality. The results suggest that Residual-NeRF improves on the state-of-the-art in depth mapping accuracy with a 46.1 % lower RMSE and a 29.5 % lower MAE. The results also suggest Residual-NeRF speeds up training and leads to more robust grasp planning.

In summary, we contribute:

- Residual-NeRF, a novel algorithm that learns a *residual NeRF* and *background NeRF*, mixed in 3D space by a *Mixnet* MLP.
- Experiments that suggest Residual-NeRF infers improved depth maps of transparent objects, speeds up training, and yields more robust grasp planning.
- Synthetic and real datasets.

4.2 Related Work

Neural Radiance Fields (NeRF)

NeRF [209] is a scene representation capable of photo-realistic novel view reconstruction. It is inspired by classic volumetric rendering techniques, tracing rays through the scene to infer radiance and density using a neural network. For every point in 3D space, the NeRF MLP infers RGB radiance and density to render an RGB image. NeRF forms the basis for sensing in this chapter.

After the introduction of NeRF, several works proposed improvements. A key challenge for the original NeRF algorithm [209] was the long training and inference time—often requiring several hours to train for a single scene. Recent progress has shown that novel representations and system optimizations [94, 187, 191, 215, 216, 246, 286, 380], and depth supervision [5, 56, 219, 284, 343] can significantly speed up training. We leverage existing optimized code to speed up training [293].

Other works have focused on improving novel view synthesis performance in challenging conditions and with sparse camera views without extrinsic calibration. It is now possible to capture a NeRF of transparent objects [127, 146], reflective surfaces [316] and low-light environments [210]. Many NeRF and other works have aimed to reduce the number of views [46, 221, 308, 387, 389] and extrinsic camera calibration [42, 132, 178, 378, 389] required. Another line of work extends novel view synthesis to dynamic scenes [71, 84, 199, 354].

Merging NeRFs

Previous work has introduced methods to enable evaluating and training multiple NeRFs in a single scene. Reiser et al. [246] introduced KiloNeRF, with the contribution of using thousands of smaller MLPs ordered in a 3D grid instead of using only a single MLP. KiloNeRF maps a position $\mathbf{x}$ to a tiny NeRF using spatial binning to its 3D grid, and achieves faster inference.

Tancik et al. [292] introduced Block-NeRF, a variant of NeRF that can represent large-scale environments. Block-NeRF also contains an appearance embedding to accommodate changes in, e.g., lighting. Zhang et al. [394] introduced NeRFusion, a method that leverages TSDF-based fusion techniques for efficient large-scale reconstruction. The authors propose to infer a per-frame local radiance field via direct network inference. They are then fused using a recurrent neural network to reconstruct a global scene representation.

In summary, related work has demonstrated novel approaches to improve view reconstruction and inference speed in large scenes. Residual-NeRF aims to improve depth mapping of transparent objects for manipulation, by learning a Residual NeRF and Mixnet on top of a pre-trained background NeRF. Mixnet does not require a pre-defined area for each NeRF, but learns to mix both NeRFs at every point in space.

Depth Perception of Transparent Objects

Several works have proposed methods for accurate depth perception, shape estimation, and/or pose estimation. Xie et al. [363] developed a pipeline based on transformer neural networks capable of transparent object segmentation. Phillips et al. [236] leveraged a random forest algorithm to extract the pose and shape of transparent objects. Xu et al. [365] contributed an algorithm for estimating 6-degree-of-freedom (DOF) pose of a transparent object using only a single RGBD image. Wang et al. [337] contributed MVTrans for depth mapping, segmentation, and pose estimation of transparent objects. Chen et al. [41] contributed a benchmark dataset for segmentation, object-centric pose estimation, and depth completion.

Ichnowski et al. [127] showed how NeRFs can be leveraged to infer state-of-the-art depth perception of transparent objects, and unlike data-centric approaches, did not require prior training on a set of objects.

4.3 Problem Statement

Given is a sequence of images of a scene without the transparent objects present, I_{bg_n} with $n \in [1, \ldots, N_{\mathrm{bg}}]$ each with camera matrix P_n (i.e., the intrinsics and extrinsics). In addition, we also have access to images of the same scene with one or more transparent objects present, I_{res_n}, with $n \in [1, \ldots, N_{\mathrm{res}}]$, P'_n. P'_n and P_n are not necessarily the same, N_{bg} and N_{res} are also not necessarily equal.

The objective is to recover novel view depth maps from any given (depth) camera pose P in the scene, and then use the novel views for grasp planning. As with Dex-NeRF and Evo-NeRF, the goal is to facilitate using a grasp planner such as Dex-Net [203]. A high-quality depth map of both opaque and transparent objects in the scene will aid motion planning and the selection of grasp location. Perceived holes in objects can lead to collisions, whereas hallucinating nonexistent surfaces may lead to no available grasp location.

4.4 Method

Residual-NeRF, shown in Figure 4.2, recovers depth using multiple camera views by first learning a background NeRF, then training a Mixnet and a residual NeRF (Section 4.4.3). As it builds on NeRF, we first review preliminaries of novel view synthesis with NeRF (Section 4.4.1). Once trained, we then use the NeRFs to render depth maps (Section 4.4.2) for the grasp planner.

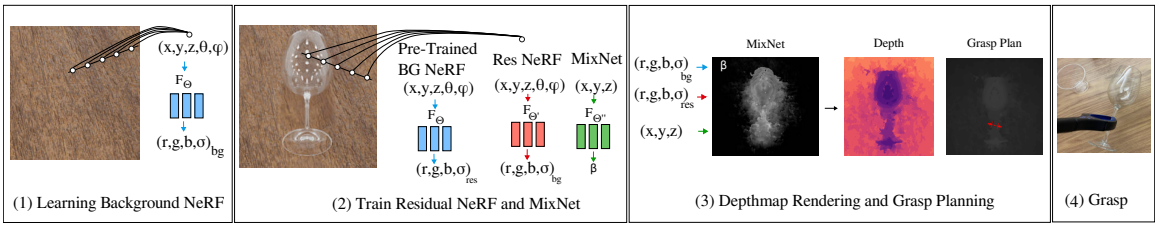


Figure 4.2: Residual-NeRF, a method that leverages mostly static scenes to improve depth perception and speed up training. We first learn a *background NeRF* of the scene without transparent objects and leverage it as a scene prior. Following this, we learn a *residual NeRF* and *Mixnet*. The *Mixnet* is an MLP that learns to combine the *background NeRF* and the *residual NeRF*. Equation 4.5 describes how the output of the *Mixnet* MLP is used to combine the two NeRFs.

4.4.1 Preliminary: Training NeRF

NeRFs [209] trace rays through images of the scene to train an MLP, Φ_r, to infer radiance and density $(RGB\sigma)$ for an input world coordinate and viewing direction (x, y, z, θ, ϕ). To accommodate high-frequency changes in 3D space, the input to NeRF is concatenated with a positional encoding in the form of sinusoids with varying frequencies. Some recent NeRF implementations use alternative representations such as a multi-resolution hash grid [216] encoding to speed up training. Thus, $\Phi_r(h(x, y, z, \theta, \phi)) = (RGB\sigma)$, where $h(\cdot)$ is the encoding function.

The expected color $C(\mathbf{r})$ by volumetric rendering is in Equation 4.1.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{4.1}$$

with

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right),$$

$T(t)$ is the accumulated transmittance along the ray from t_n to t. NeRFs approximate the function $C(\mathbf{r})$ by a learned function $\hat{C}(\mathbf{r})$, computed using Equation 4.2.

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \tag{4.2}$$

with

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right),$$

Here $\delta_i = t_{i+1} - t_i$, i.e., the distance between samples along the ray. The MLP learns $\mathbf{c}_i$, the color emitted at each point on the ray, and σ_i, the density at each point on the ray.

A NeRF can be trained by setting the loss to the RMSE between rendered and GT images. Equation 4.3 defines this photometric loss.

$$\mathcal{L}_{\text{pho}} = \sum_{(i,\mathbf{r})\in\Omega_r} ||\hat{C}_i(\mathbf{r}) - C_i(\mathbf{r})||_2^2, \tag{4.3}$$

Here $i \in [1, ..., N]$ is the frame number, $\mathbf{r}$ is the pixel location, and Ω_r is the set of all pixel locations across all frames.

4.4.2 Depth from NeRF

We adopt the approach for depth perception of transparent objects proposed by Ichnowski et al. [127]. Prior works propose computing depth using Equation 4.4,

$$\hat{D}(\mathbf{r}) = \sum_{i}^{N} T_i(1 - \exp(-\sigma_i\delta_i))t_i, \tag{4.4}$$

When propagating rays through transparent objects, this equation proves to be inaccurate as a result of multiple surfaces radiating light along the ray. Instead, as proposed by Ichnowski et al. [127], we find the point closest to the camera for which $\sigma_i \geq m$, where m is a tunable parameter. Prior work suggested that m was not overly sensitive and that a single tuned value could be reused across multiple scenes and lighting conditions.

4.4.3 Learning a Residual NeRF

In this chapter, we propose a method to leverage a NeRF of the background scene (i.e., the scene without the transparent object) to improve 3D reconstruction. Equation 4.5 shows the modification.

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-((1-\beta)\sigma_{\text{bg}} + \beta\sigma_{\text{res}})_i\delta_i))(\mathbf{c}_{\text{bg+res}})_i, \tag{4.5}$$

With:

$$\begin{aligned}
\mathbf{c}_{\text{bg+res}} &= S((1.0-\beta)\mathbf{c'}_{\text{bg}} + \beta\mathbf{c'}_{\text{res}}),\\
(\mathbf{c'}_{\text{res}}, \sigma_{res}) &= \Phi_{\text{res}}(h(x,y,z,\theta,\phi)),\\
(\mathbf{c'}_{\text{bg}}, \sigma_{bg}) &= \Phi_{\text{bg}}(h(x,y,z,\theta,\phi)),\\
\beta &= S(\Phi_{\text{mix}}(h(x,y,z))),\\
S(x) &= \frac{1}{1+e^{-x}},
\end{aligned}$$

For every point along the ray, MLP Φ_{mix} takes in the encoded position and infers $\beta \in [0, 1]$. The background NeRF infers σ_{bg} and $\mathbf{c'}_{\text{bg}}$ by training on a set of images of the background without the transparent object, while the residual NeRF infers σ_{res} and $\mathbf{c'}_{\text{res}}$ by training with the transparent object in the scene. The vector $\mathbf{c'}$ is the color $\mathbf{c}$ before it is

passed through its sigmoid activation. We add the background and residual NeRF values before going through their final activation layers, such as the sigmoid. The background NeRF is not updated while training the residual NeRF.

4.5 Experiments

We evaluate Residual-NeRF against the baselines (Section 4.5.2) on synthetic Blender scenes (Section 4.5.3) with ground truth depth. We also present a qualitative evaluation against the baselines on real-world scenes (Section 4.5.7). The results suggest Residual-NeRF outperforms the baselines by generating higher-quality depth maps. Finally we show Residual-NeRF speeds up training (Section 4.5.8), can improve grasp planning using Dex-Net (Section 4.5.9), and we ablate the impact of the Mixnet (Section 4.5.10).

4.5.1 Hyperparameters

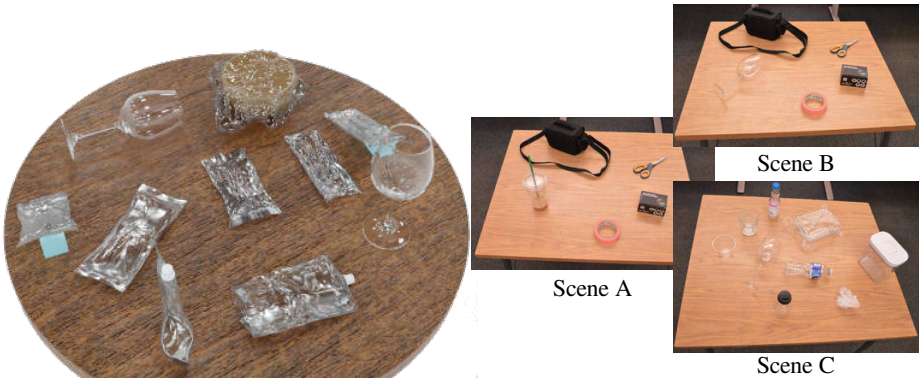
NeRFs require the setting of numerous hyperparameters potentially with great impact on performance. We have kept all vanilla NeRF parameters constant between runs, e.g., scene area, scale, and adaptive ray marching turned off. We set m (Section 4.4.2) to 3.0 for all Blender scene evaluations for all methods, and tune it to the best of our ability for each real environment and method. We tune m by starting from 0.0, to then increase m until no more holes in the depth maps disappear. From this point, increasing m will lead to more noise and no additional gains.

4.5.2 Baselines

As baselines, we evaluate Residual-NeRF against NeRF [209] and Dex-NeRF [127]. While other multi-view stereo (MVS) methods for transparent objects exist, to the best of our knowledge, they do not accept arbitrary poses.

4.5.3 Synthetic Blender Data

Residual-NeRF cannot be evaluated on existing datasets such as ClearPose [41], which captures 63 transparent objects, due to the lack of background images for training the background NeRF. Therefore, we use Blender to render nine photo-realistic objects as shown in Figure 4.3. We use the poses in a hemisphere from the original NeRF datasets [209] to render train, test, and validation images. Background NeRF and residual NeRF receive images taken from the same 100 train poses.



(1) Synthetic Objects in Blender (2) Real-World Scenes

Figure 4.3: The scenes used for evaluation. We create nine synthetic Blender scenes with transparent objects and three real-world scenes, divided into scenes A-C with increasing difficulty.

4.5.4 Real World Data

We take images of transparent objects on a wooden table to evaluate Residual-NeRF in the real world. We take the images using a Nikon D3200 DSLR camera; the poses are taken in a random hemisphere. Lighting conditions, focal length, aperture, ISO, and white balance were constant while collecting datasets. We use COLMAP [265] to retrieve the camera poses.

The background NeRF receives 100 images to create a strong scene prior. The residual NeRF, Mixnet, and the baselines are trained with 50 images, excluding those omitted by COLMAP. This experiment design simulates scenarios where a predominantly static scene can be extensively observed from many poses, but the scene under evaluation is limited to a smaller number of viewpoints.

The three evaluation scenes A-C in increasing difficulty are visible in Figure 4.3. Scene A has four opaque objects in the background scene, with an added transparent coffee container with coffee in the evaluation scene. Scene B has the same four opaque objects in the background scene but with a wine glass added in the evaluation scene. Scene C has 6 transparent objects present in the background scene; 3 more transparent objects are added in the evaluation scene: a wine glass, kitchen wrap, and a glass bottle with a blue cap.

4.5.5 Implementation Details

We implement Residual-NeRF and the baselines by building on Torch-NGP [293]. Torch-NGP [293] is a PyTorch version of Instant-NGP [216], and uses a multi-resolution hash grid to speed up training. It runs at a similar but slightly slower speed as compared to Instant-NGP while offering a more efficient development cycle with only raytracing implemented in CUDA. We train Residual-NeRF and all baselines on an RTX 4090 and an AMD Ryzen Threadripper PRO 5965WX with 256 GB of RAM.

Table 4.1: Root Mean Square Error (RMSE) in synthetic Blender Scenes.

	Bowl	Drink Flat	Drink Up	Needles Flat	Needles Up	Scissors Flat	Scissors Up	Wine Flat	Wine Up
Method					RMSE ↓				
NeRF [209]	0.1900	0.6398	0.4361	0.3529	0.1599	0.5523	0.1622	0.2166	0.5973
Dex NeRF [127]	0.0365	0.1065	0.0699	0.0293	**0.0431**	0.2624	0.0320	0.0425	0.3159
Residual-NeRF	**0.0213**	**0.0234**	**0.0316**	**0.0208**	0.0437	**0.0506**	**0.0289**	**0.0388**	**0.2462**

Table 4.2: Mean Absolute Error (MAE) in synthetic Blender Scenes.

	Bowl	Drink Flat	Drink Up	Needles Flat	Needles Up	Scissors Flat	Scissors Up	Wine Flat	Wine Up
Method					MAE ↓				
NeRF [209]	0.1453	0.3424	0.3062	0.1698	0.1067	0.2505	0.1103	0.1483	0.2157
Dex NeRF [127]	0.0195	0.0189	0.0255	0.0127	**0.0193**	0.0302	0.0160	0.0255	0.0338
Residual-NeRF	**0.0140**	**0.0138**	**0.0170**	**0.0115**	0.0207	**0.0129**	**0.0140**	**0.0142**	**0.0238**

4.5.6 Blender Depth Results

We evaluate the inferred depth maps by Residual-NeRF and the baselines by comparing them against the ground truth provided by Blender.

4.5.6.1 Quantitative Comparison

We compare Residual-NeRF against NeRF and Dex-NeRF by computing the MAE (Equation 4.6) and RMSE (Equation 4.7).

$$\text{MAE} = \frac{\sum_{(i,\mathbf{r})\in\Omega_r} \|\hat{D}_i(\mathbf{r}) - D_i(\mathbf{r})\|_1}{n}, \tag{4.6}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,\mathbf{r})\in\Omega_r} \|\hat{D}_i(\mathbf{r}) - D_i(\mathbf{r})\|^2}{n}}, \tag{4.7}$$

Here $i \in [0, ..., N]$ is the frame number, $\mathbf{r}$ is the pixel location, and Ω_r is the set of all pixel locations across frames. $\hat{D}(\mathbf{r})$ is the inferred depth in meters, $D(\mathbf{r})$ is the GT depth in meters. We crop each image before evaluation to focus evaluation on the transparent object. In each crop, the entire transparent object is visible, while the background is partially cropped out.

Table 4.1 shows the RMSE and Table 4.2 the MAE for Residual-NeRF compared against the baselines. It shows that Residual-NeRF outperforms the baselines except for the 'Needles Up' scene.

4.5.6.2 Qualitative Comparison

Figure 4.4 shows the depth maps inferred by Dex-NeRF [127] and Residual-NeRF. For most scenes, it shows that Residual-NeRF reduces holes and noise in depth maps. Intuitively, the background NeRF makes it less likely for holes in the table to appear, as a result of the scene geometry prior.

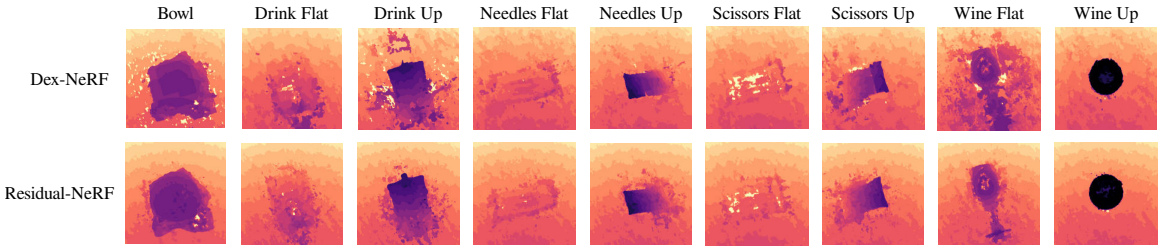


Figure 4.4: Depth maps for Residual-NeRF and Dex-NeRF evaluated on the synthetic Blender dataset. The results suggest Residual-NeRF improves depth maps with fewer holes and less noise.

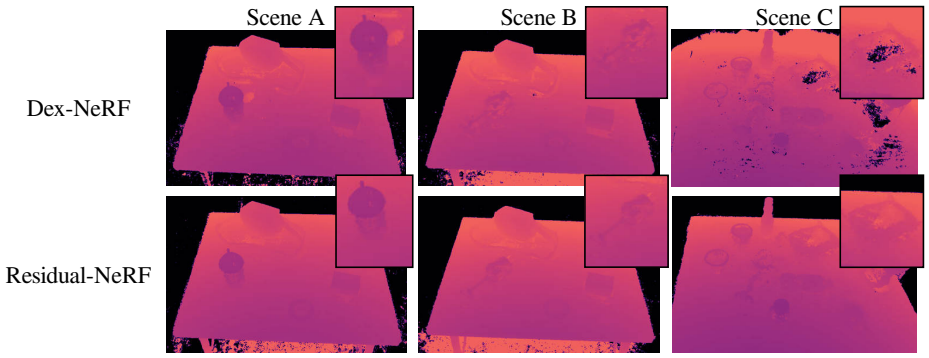


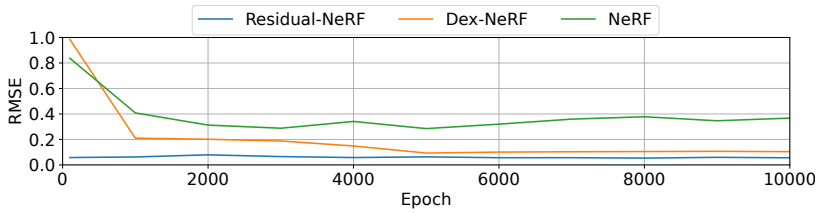
Figure 4.5: Depth maps inferred by Dex-NeRF [127] and Residual-NeRF in the real world. The results suggest Residual-NeRF results in fewer holes and less noise.

The bowl, drink up, and wine flat scenes show that transparent objects can make reconstruction of opaque objects more difficult, as a result of the introduced ambiguity. Holes in opaque objects may cause the end effector to collide with unseen objects.

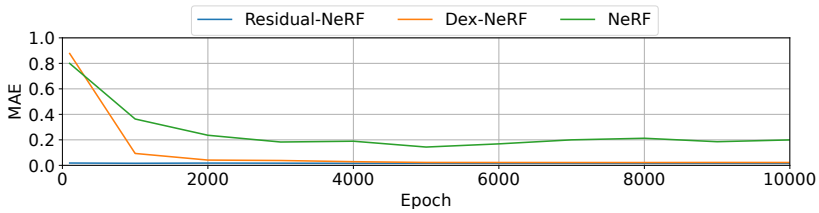
The results could be further improved by tuning m (Section 4.4.2) for each scene; we have opted for setting $m = 3$ for all scenes. In our testing, the optimal m appears to be scene-dependent as well as NeRF implementation-dependent. Ichnowski et al. [127] found $m = 15$ to work best, evaluating different scenes and using a NeRF implementation without multi-resolution hash encoding.

4.5.7 Real World Depth Results

Figure 4.5 shows the depth maps inferred by Dex-NeRF [127] and Residual-NeRF in the real world. The results show Residual-NeRF improves the depth maps using the background NeRF as a scene prior. Similar to the results in simulation, we observe fewer holes in the table and in the objects themselves.



(a) Root Mean Square Error (RMSE).



(b) Mean Absolute Error (MAE).

Figure 4.6: RMSE and MAE in depth maps, logged over the number of training epochs and averaged over all synthetic scenes. The results suggest Residual-NeRF greatly improves training speed with respect to depth reconstruction.

4.5.8 Training Speed

Speeding up NeRF training is critical to making NeRF a viable option in industrial and home applications. Any additional speed-ups provided by Residual-NeRF can further improve sped-up implementations such as Evo-NeRF [146].

To evaluate the quality of depth reconstruction over time, we log depth maps along with the elapsed training epochs. The resulting RMSE and MAE averaged over all synthetic scenes during training are visible in Figure 4.6, assuming a pre-trained background NeRF for Residual-NeRF.

The results suggest Residual-NeRF successfully leverages the background NeRF to speed up training. We implement all methods in the same Torch-NGP [293] codebase, making implementation-related differences in training speed unlikely.

4.5.9 Grasp Planning using Dex-Net

Figure 4.7 shows the output of Dex-Net on Residual-NeRF and Dex-NeRF. The results suggest the output from Residual-NeRF leads to more robust grasping. We use the grasp location inferred by Dex-Net to grasp transparent objects using a Franka robot manipulator and its parallel-jaw gripper.

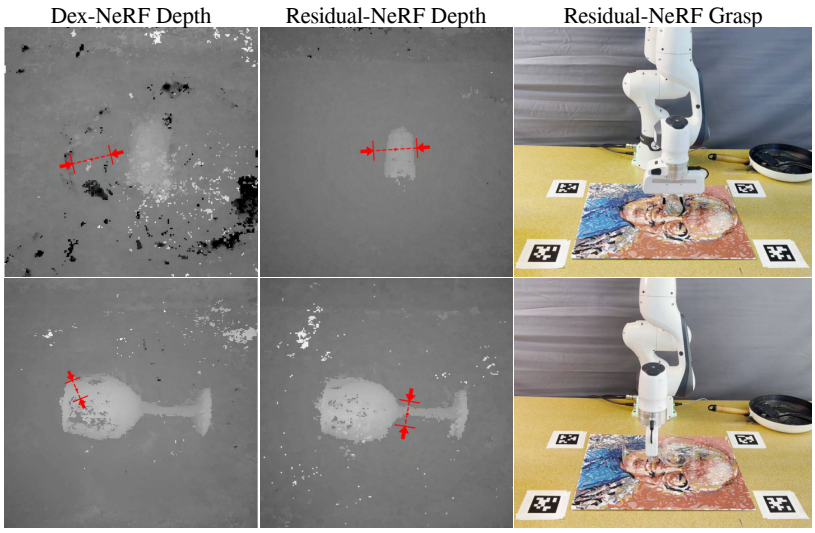


Figure 4.7: Dex-Net evaluated on the output of Dex-NeRF and Residual-NeRF. The results suggest the output from Residual-NeRF leads to more robust grasping.

4.5.10 Residual-NeRF-S Mixnet Ablation

To better understand the use of the Mixnet, we evaluate an alternative method where NeRFs are merged by adding. We refer to this method as Residual-NeRF-S. The equations for Residual-NeRF-S are:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-(\sigma_{\text{bg}} + \sigma_{\text{res}})_i \delta_i))(\mathbf{c}_{\text{bg+res}})_i, \tag{4.8}$$

with

$$\begin{aligned} \mathbf{c}_{\text{bg+res}} &= S(\mathbf{c'}_{\text{bg}} + \mathbf{c'}_{\text{res}}), \\ (\mathbf{c'}_{\text{bg}}, \sigma_{bg}) &= \Phi_{\text{bg}}(h(x, y, z, \theta, \phi)), \\ (\mathbf{c'}_{\text{res}}, \sigma_{res}) &= \Phi_{\text{res}}(h(x, y, z, \theta, \phi)), \\ S(x) &= \frac{1}{1 + e^{-x}}, \end{aligned}$$

Instead of training a Mixnet to merge background and residual NeRFs, the outputs of both MLPs are simply added before their activation layer. Figure 4.8 shows a comparison of depth and view reconstruction; both are worse compared to Residual-NeRF.

Especially with partially opaque objects, significant noise is introduced in the view reconstruction and depth maps. We believe this is due to the unpredictable behavior of the background NeRF in areas that were previously unoccupied. Without Mixnet, the residual NeRF not only has to understand the scene but also deduce the background NeRF's color and density predictions in previously empty regions.

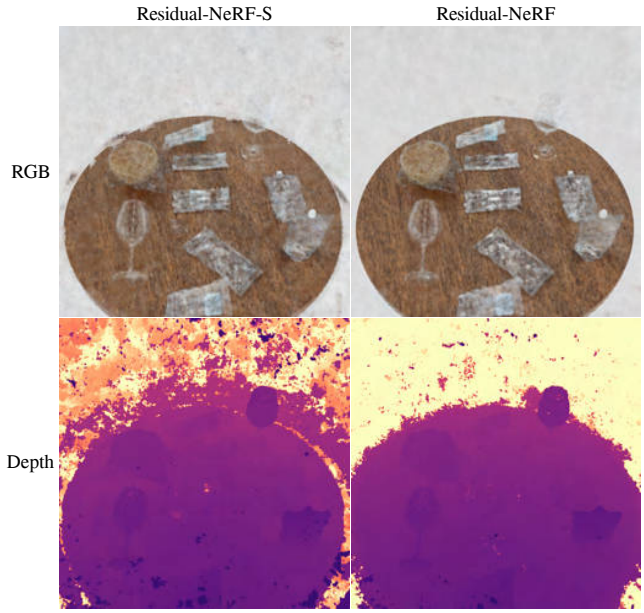


Figure 4.8: Naive Merging (Residual-NeRF-S) as introduced in Equation 4.8 compared against Mixnet (Residual-NeRF). The results suggest Mixnet helps view reconstruction and depth mapping when merging NeRFs.

4.6 Conclusion and Discussion

In this chapter, we introduce Residual-NeRF, a method designed to leverage a strong prior on the background scene to improve depth perception and speed up training. We study the problem of perceiving depth in mostly static scenes with transparent objects, as is typical of a robot's working area, e.g., shelves, desks, and tables.

Residual-NeRF begins by learning a *background NeRF* of the entire scene without transparent objects. Following this, a *residual NeRF* and *Mixnet* are trained to complement the *background NeRF*. The results suggest Residual-NeRF learns faster and achieves better depth mapping, compared to the baselines.

This chapter could be improved by comparing against more MVS methods non-specific to transparent objects. Future work may also include combining Residual-NeRF with recent advances in depth map completion. Future research could explore the performance across different transparent objects and environment conditions.

4.7 Acknowledgement

We thank the Center for Machine Learning and Health (CMLH) for their generous Fellowship in Digital Health. We also thank the Pittsburgh Supercomputing Center for their resources, and Uksang Yoo, Peter Schaldenbrand and Jean Oh for help with the robot experiments.

Chapter 5

DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation

ORIGINALLY APPEARED AS

Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, Jeffrey Ichnowski.
DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation [71].
Proceedings of Algorithmic Foundations of Robotics (WAFR), 2024.

Chapter in Context

Residual-NeRF improved static scene reconstruction using neural rendering. Robots benefit from an understanding of not just static scene geometry, but also how scenes evolve over time—their dynamics. This chapter extends neural rendering to obtain 3D point tracks from highly deformable scenes such as cloth. We show that this approach may yield millimeter-precise tracking. This advances SOLF's *obtainability* by recovering motion supervision in scenes with large deformations.

Teaching robots to fold, drape, or reposition deformable objects such as cloth will unlock a variety of automation applications. While remarkable progress has been made for rigid object manipulation, manipulating deformable objects poses unique challenges, including frequent occlusions, infinite-dimensional state spaces and complex dynamics. Just as object pose estimation and tracking have aided robots for rigid manipulation, dense 3D tracking (scene flow) of highly deformable objects will enable new applications in robotics while aiding existing approaches, such as imitation learning or creating digital twins with real2sim transfer. We propose DeformGS, an approach to recover scene flow in highly deformable scenes, using simultaneous video captures of a dynamic scene from multiple cameras. DeformGS builds on recent advances in Gaussian splatting, a method that learns the properties of a large number of Gaussians for state-of-the-art and fast novel-view synthesis.

DeformGS learns a deformation function to project a set of Gaussians with canonical properties into world space. The deformation function uses a neural-voxel encoding and a multilayer perceptron (MLP) to infer Gaussian position, rotation, and a shadow scalar. We enforce physics-inspired regularization terms based on conservation of momentum and isometry, which leads to trajectories with smaller trajectory errors. We also leverage existing foundation models SAM and XMem to produce noisy masks, and learn a per-Gaussian mask for better physics-inspired regularization. DeformGS achieves high-quality 3D tracking on highly deformable scenes with shadows and occlusions. In experiments, DeformGS improves 3D tracking by an average of 55.8 % compared to the state-of-the-art. With sufficient texture, DeformGS achieves a median tracking error of 3.3 mm on a cloth of 1.5 $\times$ 1.5 m in area. Website: `https://deformgs.github.io`

5.1 Introduction

Recent advances in robot learning have demonstrated impressive performance on challenging tasks, including rigid and deformable object manipulation. Scaling these approaches to deployment will require an improvement in robustness and learning from few demonstrations. A promising avenue for improving robot learning performance is the use of intermediate representations and foundation models, including 6D object pose estimation [57, 60, 179, 290, 345, 348, 362], semantic latent features [225], and 2D pixel-wise tracking [139, 350]. However, perception and representations that will lead to robust manipulation of deformable objects remain an open challenge, due to self-occlusions, shadows, and varying (or lack of) textures.

Three-dimensional dense point tracking, or *3D scene flow*, is a useful representation for robot manipulation, as it provides flexibility to represent high-dimensional dynamic state changes, while the deformable objects drop, deform, and drape during manipulation. In particular, dense 3D scene flow can be an input to imitation learning policies [20, 350], can be used to learn a transition model [271], to identify and track task-relevant key points, or to create a digital twin through real2sim transfer. Recent work in monocular tracking has seen improvements in performance on datasets such as TAP-Vid [62], but it remains unclear how to effectively lift from 2D tracking to 3D for robotic spatial understanding in challenging highly deformable scenes.

To overcome these limitations, Gaussian Splatting provides a promising avenue. Recent work demonstrated Gaussian Splatting [145, 147] can yield state-of-the-art novel-view synthesis and rendering speeds exceeding 100 fps. Concretely, 3D Gaussian Splatting uses a fast differentiable renderer to fit the colors, positions, and covariances of a set of 3D Gaussians. An extension of 3D Gaussian splatting [199] showed dynamic scenes can be modeled by explicitly optimizing the properties of Gaussians over time, resulting in novel-view synthesis and scene flow.

Explicitly optimizing the Gaussian properties as in Dynamic 3D Gaussians [199] may result in degraded performance with large deformations and shadows. The Gaussian properties may converge to local optima, especially in scenes with large deformations, strong

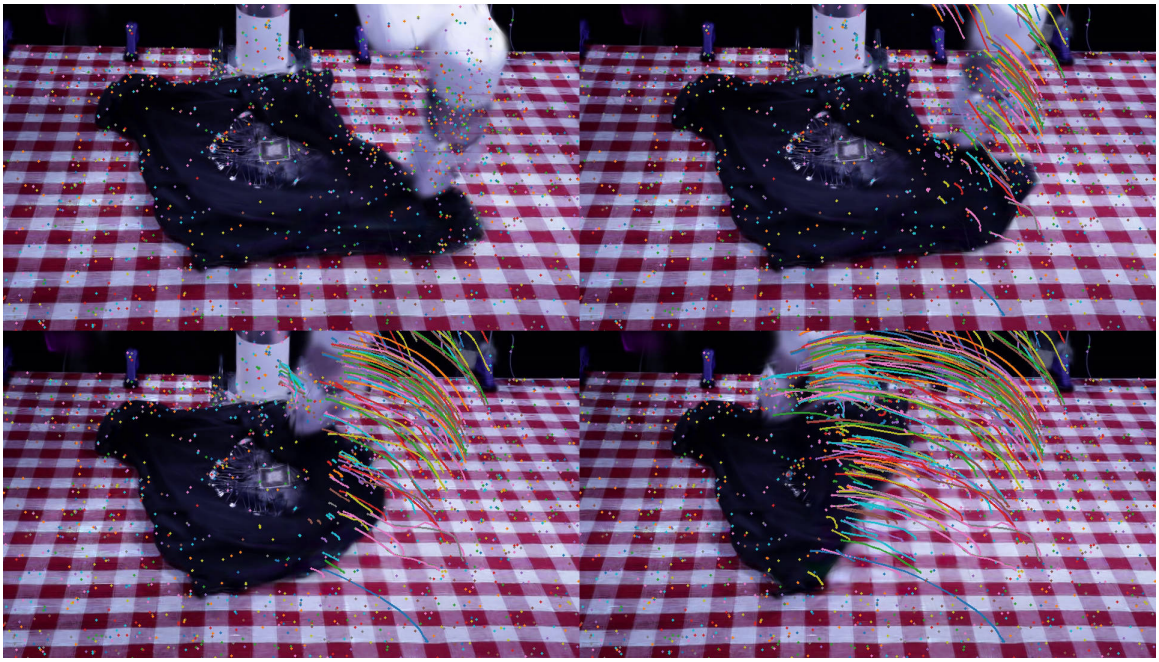


Figure 5.1: We propose DeformGS, a method that improves state-of-the-art methods for accurate 3D point tracking in highly deformable scenes. This figure shows DeformGS tracking in a real-world scene from the Robo360 [176] dataset.

shadows, and occlusions.

We propose DeformGS, a method that uses time-synchronized image frames from a calibrated multi-camera setup to track 3D geometries of deformable objects as they move through shadows and occlusions. DeformGS learns the canonical state of a set of Gaussians and a deformation function that maps the Gaussians into world space. This enables tracking by recovering scene flow, and novel-view rendering (through splatting) using a fast differentiable rasterizer.

We evaluate DeformGS in six photo-realistic synthetic scenes of varying difficulty. The scenes contain large deformations, shadows, and occlusions (Figure 5.1 shows tracking trajectories computed by DeformGS). Empirical results show that DeformGS infers 55.8 % more accurate 3D tracking results compared to previous state-of-the-art [199, 354]. In a scene with a 1.5 m $\times$ 1.5 m cloth (i.e., Scene 1), DeformGS can track cloth deformation with as low as 3.3 mm median tracking error.

We also evaluate DeformGS in the real world on the Robo360 [176] dataset. We show qualitative results for tracking rigid and deformable objects in cluttered scenes, and study two robotics applications: (1) real2sim transfer to create a digital twin, and (2) tracking task-relevant keypoints for downstream grasping applications.

In summary, our contributions are as follows:

- We provide the first approach to accurately perform 3D dense tracking of deformable objects using 4D Gaussians.

- We provide experiments that suggest state-of-the-art performance in 3D metric tracking of deformable cloth. DeformGS improves tracking accuracy by an average of 55.8% in synthetic experiments and demonstrates robust 3D tracking in the real world for deformable objects. The latter can be exploited as a representation for imitation learning and represents a new method for building digital twins.
- A set of six synthetic scenes with large deformations, strong shadows, and occlusions. The scenes and source code are available at the project website: `https://deformgs.github.io`

5.2 Related Work

5.2.1 Neural Rendering for Novel View Synthesis

DeformGS builds on prior work in novel-view synthesis, and uses photometric consistency as a signal to achieve 3D tracking. A popular novel view synthesis approach is NeRF [209]. It uses neural networks to learn scene representations that are capable of photo-realistic novel view reconstruction. Particle-based methods use a more explicit representation than typical NeRF-based approaches. DeformGS builds on 3D Gaussian Splatting [145, 147] which belongs to the latter category. 3D Gaussian Splatting [145] proposed a differential rasterizer to render a large number of Gaussian 'splats,' each with their state including color, position, and covariance matrix. Contrary to the NeRF-based approaches, Gaussian splatting achieves real-time rendering of novel views with state-of-the-art performance.

5.2.2 Dynamic Novel View Synthesis

The assumption of static scenes in neural rendering approaches prevents application to real-world scenarios with moving objects or humans, such as the dynamic and deformable scenes in this chapter. One line of work to address this assumption is adding a time dimension to NeRF modeling [66, 90, 172, 359]. Prior works either condition the neural field on explicit time input or a time embedding. Another line of work learns a deformation field to map 4D points into a canonical space [229, 239], i.e., every 4D point in space and time maps to a 3D point in a canonical NeRF. DeVRF [186] proposed to model the 3D canonical space and 4D deformation field of a dynamic, non-rigid scene with explicit and discrete voxel-based representations.

Several recent works extend the above approaches to 3D Gaussian splatting. Dynamic 3D Gaussians [199] explicitly models the position and covariance matrix of each Gaussian at each time step. This method struggles in dynamic scenes with large deformations, strong shadows, or occlusions. We build on another recent work, 4D Gaussian splatting [354], which uses feature encoding techniques proposed in HexPlanes [32] and K-planes [87], and learns a deformation field instead.

5.2.3 Point Tracking

Point tracking methods, usually trained on large amounts of data, aided previous 3D tracking approaches by providing a strong prior [186]. We also construct several baselines that include point tracking methods (Section 5.6). Prior work on point tracking often studies tracking 2D points across video frames, where a dominant approach is training models on large-scale synthetic datasets containing ground-truth point trajectories [62, 63, 107, 398] or dense optical flows [330]. Optical flow [17, 298] or scene flow [15, 103, 315, 317] can also be viewed as single-step point-tracking in 2D and 3D, respectively.

Another relevant line of work tightly couples dynamic scene reconstruction and motion estimation of non-rigid objects. A predominant setup is fusing RGBD frames from videos of dynamic scenes or objects [220]. Tracking or correspondence-matching methods see a progression from template-based tracking of objects with known shape or kinematics priors (such as human hand, face or body poses) [33, 223, 264], to more general shapes or scenes [25, 26, 406]. The main difference from these works is that we do not use depth input, and perform more rigorous quantitative evaluations on tracking specific points.

Most related to ours are the more recent methods that obtain tracking from neural scene rendering. DCT-NeRF [321] learns a coordinate-based neural scene representation that outputs continuous 3D trajectories across the whole input sequence. PREF [278] optimizes a dynamic space-time neural field with self-supervised motion prediction loss. Most recently, Luiten et al. [199] model dynamic 3D Gaussians explicitly across timestamps to achieve tracking. While the method presented in this chapter also leverages 3D Gaussians, in contrast to the explicit modeling in Dynamic 3D Gaussians [199], we learn a deformation function that scales much better with video length, and we focus on deformable objects that are more challenging than the ball-throwing videos used in [199].

5.2.4 Tracking for Robotics

A core motivation for studying point tracking is the potential it can unlock for robotics applications: for example, RoboTAP [314] shows pre-trained point-tracking models improve sample efficiency of visual imitation learning. It detects task-relevant keypoints, infers where those points should move to, and computes an action that moves them there. Any-point [350] learns to predict keypoint tracks, but conditioned on language inputs. Track2Act [20] builds on Any-point by learning a generalizable zero-shot policy, which only needs a few embodiment-specific demonstrations.

Rigid-body, or 6D, pose tracking and estimation have a rich history in robotics due to their foundational ability to model the world for a robot to manipulate [57, 60, 179, 214, 290, 345, 347, 348, 362]. In this chapter, we propose a deformable object analog of 6D pose tracking with the aim of extending successes to deformable object manipulation.

While existing methods leverage 2D tracking, and learn an additional policy to output robot actions, DeformGS provides a more powerful representation that allows for reasoning directly in 3D, instead of in the 2D image space.

5.3 Problem Statement

Given a set of timed image sequences captured from multiple cameras with known intrinsics and extrinsics, the objective is to learn a model that performs 3D tracking and novel view synthesis. Each image sequence is captured over the same time interval $t \in [0, H]$.

3D Tracking The primary goal is to recover the trajectory of any point in a dynamic scene by modeling the deformation of Gaussians over time. Thus, the objective is to find a function $x_t = Q(x_0, t_0, t)$, where $x_0 \in \mathbb{R}^3$ is the location of a point of interest at a chosen time $t_0 \in [0, H]$, while $x_t \in \mathbb{R}^3$ is the location of the same point at another chosen time $t \in [0, H]$. The function Q is valid for any point x_0 and any $t \in [0, H]$, allowing for tracking of any point in space.

Novel View Synthesis The secondary goal is to achieve accurate scene flow by using photometric consistency as a supervision signal. To achieve this, the objective is to recover novel views from arbitrary viewpoints. The extrinsics at any viewpoint can be captured by matrix P, with $P = K[R|T]$. Here K is the intrinsics matrix, R is the rotation matrix of a camera with respect to the world frame, and T is the translation vector with respect to the world frame. Concretely, the goal is to learn a function V such that $I_{P,t} = V(P, t)$, where $I_{P,t}$ is an image rendered from a camera with extrinsics P at time t. As with the tracking objective, the time parameter is valid for any $t \in [0, H]$.

5.4 Preliminary

5.4.1 Gaussian Splatting

3D Gaussian Splatting [145] deploys an explicit scene representation by rendering a large set of 3D Gaussians each defined by their mean position μ and covariance matrix Σ. Given $x \in \mathbb{R}^3$, its Gaussian is

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)},$$

Directly optimizing the covariance matrix Σ would lead to infeasible covariance matrices, as they must be positive semi-definite to have a physical meaning. Instead, Gaussian Splatting [145] proposes to decompose Σ into a rotation R and scale S for each Gaussian:

$$\Sigma = RSS^T R^T,$$

and optimize R, S, and the mean position.

Given the transformation W of a camera, the covariance matrix can be projected into image space as

$$\Sigma' = JW\Sigma W^T J^T,$$

where J is the Jacobian of the affine approximation of the projective transformation.

During rendering, we compute the color C of a pixel by blending N ordered Gaussians overlapping the pixel:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j).$$

where c_i is the color of each Gaussian and α_i is given by evaluating a Gaussian with covariance multiplied with a learned per-Gaussian opacity σ [145, 379]. This representation allows for fast rendering of novel views, and aims to reconstruct the geometry of the scene.

5.4.2 Deformation Fields for Dynamic Scenes

Prior work showed that a deformation function combined with a static NeRF in a canonical space can enable novel view synthesis in dynamic scenes. The deformation function $F_{\text{NeRF}} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ deforms a point in world coordinates (x', y', z') into a point in canonical space (x, y, z).

Prior work formulated F_{NeRF} as an MLP [239] and a multi-resolution voxel grid [84]. Wu et al. [354] applied a similar approach to arrive at Gaussian splatting of dynamic scenes. Given the state of a single canonical Gaussian, defined by $P = [\mu, S, R, \sigma, C]$ at time t, a deformation function is

$$P' = F_{\text{4DGS}}(P, t),$$

where F_{4DGS}, similar to the HexPlanes [32], contains a neural-voxel encoding in space and time. 4D-GS [354] starts with a *coarse* stage for initializing the canonical space, by setting $P' = P$, bypassing the deformation field and learning canonical properties directly. During the *fine* stage we learn the deformation function.

We propose DeformGS (Figure 5.2), based on 4D-Gaussians [354], to render novel views in dynamic scenes. The key differences with 4D-GS are: (1) we propose an intuitive method to track canonical Gaussians in world coordinates using a continuous deformation function, (2) the output of the deformation function is different, e.g., DeformGS infers shadows and does not alter opacity or scale over time, (3) using the method shown in Figure 5.3, we enforce physics-inspired regularization losses on the 3D trajectories of Gaussians, and (4) DeformGS uses dynamic object masks as input to learn a per-Gaussian mask.

5.5 Method

DeformGS achieves novel-view synthesis and high-quality 3D tracking using a canonical space of Gaussians and a deformation function to deform them to world space (Section 5.5.1). To incentivize learning physically plausible deformations, DeformGS introduces several regularization terms (Section 5.5.2). Finally, DeformGS learns 3D masks to focus regularization and Gaussian deformation on dynamic parts of the scene (Section 5.5.3).

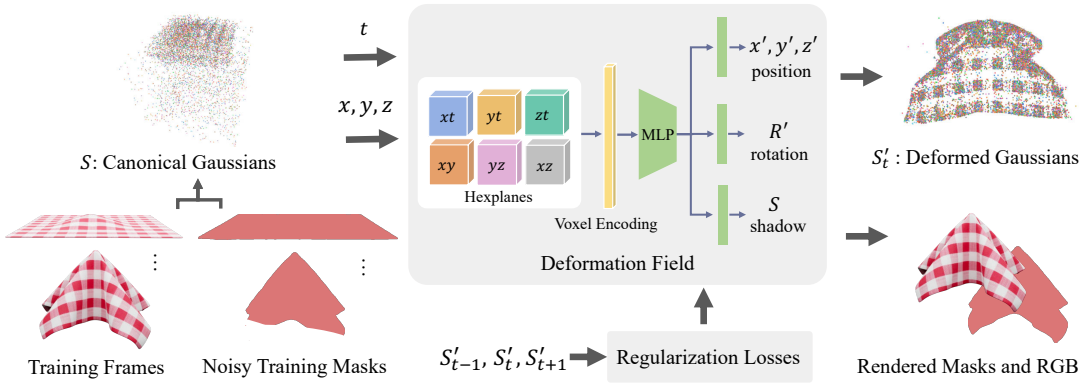


Figure 5.2: DeformGS maps a set of Gaussians with canonical properties to world space using a deformation function F. The deformation function takes in the position of a Gaussian (x, y, z) and a queried timestamp t, to infer shadow s, rotation R' and metric position (x', y', z'). During training, we use the metric positions and rotations to regularize the deformation function, considering the state at $t = \{i-1, i, i+1\}$ with Gaussian metric states P'_{t-1}, P'_t, P'_{t+1}.

5.5.1 4D Gaussian Splatting

Canonical Neural Voxel Encoding. As with prior work, DeformGS learns a deformation function F from a canonical space. We use a neural-voxel encoding to ensure F has sufficient capacity to capture complex deformations. Prior work [84, 87, 216, 354] showed that neural-voxel encodings improve the speed and accuracy of novel-view synthesis in dynamic scenes. We leverage HexPlanes [32, 354] to increase capacity for simultaneous 3D tracking and novel-view synthesis.

Figure 5.2 shows an overview of the canonical neural-voxel encoding. Each of the six voxel modules can be defined as $R(i, j) \in \mathbb{R}^{h \times lN_i \times lN_j}$. Here $\{i, j\} \in \{(x, y), (x, z), (y, z), (x, t), (y, t), (z, t)\}$, i.e., we adopt HexPlanes in all possible combinations. h is the size of each feature vector in the voxel, N_i, N_j are the sizes of the HexPlanes in each dimension, l is the upsampling scale. In every module, each plane has a different upsampling scale l. To query the multi-resolution voxel grids, we query each plane using bilinear interpolation to finally arrive at a feature vector used by the deformation MLP.

Deformation MLP. The deformation MLP takes in the voxel encoding and uses the encoding to deform the canonical Gaussians into world coordinates. Figure 5.2 shows the deformation MLP, which infers position, rotation, and a shadow scalar, given a feature vector from the neural voxel encoding. We choose this set of outputs to model rigid-body transformations of each Gaussian and changes in illumination. Modeling changes in illumination is critical in the presence of shadows. We multiply the RGB color of each Gaussian by the shadow scalar $s \in [0, 1]$, and the shadow scalar is in the range $[0, 1]$ by feeding the output of the MLP through a sigmoid activation function.

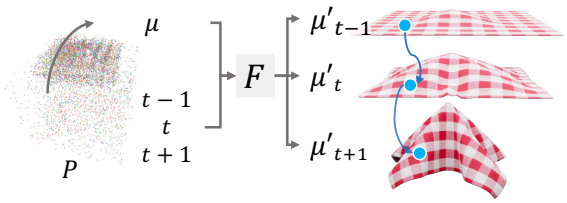


Figure 5.3: DeformGS uses three adjacent timesteps at every iteration to enforce physics-inspired regularization terms. All Gaussians are deformed to world space using the deformation function F, and rasterized to compute the photometric loss and its gradients. We use the positions of the Gaussians in world coordinates to compute the regularization terms based on local isometry and conservation of momentum (Section 5.5.2).

Next, we deform the Gaussians, modifying their mean positions μ and rotation R, and arrive at a set of Gaussians in the world space each with state P'. The differentiable rasterizer from Gaussian Splatting [145] then renders the Gaussians to retrieve gradients for regressing both the canonical Gaussian states and the parameters of the deformation function.

Unlike 4D-Gaussians [354], we propose to not infer opacity or scale using the deformation field. Optimizing for opacity and scale over time would allow Gaussians to disappear or appear instead of following the motion, which would make tracking less accurate. This design choice reduces the capacity of the deformation function, hence a lower view reconstruction quality as compared to 4D-Gaussians is expected.

5.5.2 3D Tracking using 4D Gaussians

Physics-Inspired Losses. Figure 5.3 shows the process of tracking Gaussians from the canonical space in world space. By querying the deformation function F, we can track the position of a Gaussian along the entire trajectory.

Without additional supervision, this approach will not necessarily converge to physically plausible deformations. Especially when objects include areas with little texture and uniform color, the solution space for all deformations is underconstrained by photometric consistency alone. To learn a more grounded deformation function, we propose regularization terms inspired by physics.

After empirically evaluating several combinations of regularization terms, we adopt the isometry loss proposed in [199] and add a conservation of momentum term. The first term captures a local isometry loss, which we compute based on the state of the k nearest neighboring (KNN) Gaussians.

Local Isometry Loss We incentivize the Gaussians to keep the relative position of the k nearest neighbors constant w.r.t. $t = 0$. With sufficient deformation, this assumption will be broken at a larger scale, but at a local scale, this regularization avoids drift from the ground-truth trajectory. The isometry loss is

$$\mathcal{L}_t^{\text{iso}} = \frac{1}{k|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \text{knn}_i} w_{i,j} \left| \|\mu_{j,0} - \mu_{i,0}\|_2 - \|\mu_{j,t} - \mu_{i,t}\|_2 \right|.$$

with

$$w_{i,j} = \exp\left(-\lambda_w \|\mu_{j,0} - \mu_{i,0}\|_2^2\right),$$

Here $\mathcal{P}$ is the set of all Gaussians.

Conservation of Momentum We add a term to incentivize conservation of momentum. Newton's first law states that objects without external forces applied, given some mass m and velocity vector $\mathbf{v}$, maintain their momentum $m \cdot \mathbf{v}$. We introduce the regularization term

$$\mathcal{L}_{i,t}^{\text{momentum}} = \|\mu_{i,t+1} + \mu_{i,t-1} - 2\mu_{i,t}\|_1.$$

This term incentivizes a constant-velocity vector and has the effect of imposing a low-pass filter on the 3D trajectories. It smooths out trajectories with many sudden changes of direction and magnitude (momentum).

5.5.3 Learning 3D Masks

Learning accurate 3D tracking in scenes with a mix of static and dynamic objects and rich textures poses significant challenges, mainly: (1) imposing physics-inspired regularization terms on all Gaussians may cause issues when dynamic and static objects interact, and (2) modeling millions of dynamic Gaussians can become a significant computational burden.

To address this, DeformGS takes noisy masks of dynamic scene components such as cloth, and learns which Gaussians are dynamic. More formally, we render a mask M by

$$M = \sum_{i \in N} m_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j).$$

where m_i is a per-Gaussian property, with $m_i \in [0, 1]$. We then add a regularization term to the loss function s.t. m_i is regressed to best reconstruct M. Finally, DeformGS uses m_i to select a subset of Gaussians to be dynamic, and applies regularization terms only to those Gaussians.

5.6 Experiments

We evaluate DeformGS on synthetic and real-world datasets of scenes with highly deformable objects. Section 5.6.1 provides details on the simulation experiment setup, evaluation metrics, and baseline methods. Section 5.6.2 reports evaluation results from the

compared methods and provides analysis. Section 5.6.3 lists the real-world evaluation setup, and finally in Section 5.6.4 we provide a qualitative evaluation of the performance of DeformGS in the real-world.

5.6.1 Simulation Experiment Setup

Dataset Preparation We use Blender to model dynamic cloth sequences and render photo-realistic images. We create 6 distinct scenes, each containing a different cloth with distinct visual and physical properties, and we render images from 100 different camera views and 40 consecutive time steps for training for a total of 4,000 images per scene. The cloth deformations are introduced by dropping each cloth over one or a few invisible balls onto a ground plane or by constraining the cloth at an attachment point. We obtain ground-truth trajectories by tracking the mesh vertices of deformable objects in Blender. Every scene contains a single deformable object and a rendered background.

Oracle Baselines We compare DeformGS to 2D tracking oracle models which have access to ground truth depth and trajectory information. While these methods were not designed for 3D tracking, they are well-known for their impressive 2D tracking performance. Their numbers aid in putting the tracking performance of the other baselines into context. We run RAFT [298] on all views, project tracking to 3D using ground truth depth, and report the mean results as the RAFT model. We provide two additional oracle methods which have access to the ground-truth trajectories as well. *RAFT Oracle* first evaluates on all views, to then output only the trajectories from the view with the lowest median trajectory error (MTE). We also report *OmniMotion Oracle*, which runs OmniMotion [330] on the viewpoint with the lowest MTE for RAFT. Training OmniMotion takes roughly 12–13 hours per viewpoint on an NVIDIA RTX 4090 GPU, making training on all 100 views impractical. The numbers from *RAFT Oracle* and *OmniMotion Oracle* are not an apples-to-apples comparison with the other methods, as to obtain their result they had to access privileged ground-truth trajectories.

Gaussian Splatting Baselines (1) Dynamic 3D Gaussians (*DynaGS*) [199], which also builds on 3D Gaussian splatting for dynamic novel-view synthesis, except it explicitly models the positions and rotations of each Gaussian at each time-step. This results in straightforward tracking of any point via finding the trajectory of the learned Gaussian closest to a queried point. Although the original paper assumes a known point cloud at the first frame, we instead use a randomly sampled point cloud for a fair comparison, with DynaGS and DeformGS both not using depth information.

(2) Finally, we compare to tracking using 4D-Gaussians [354] (*4D-GS*). We add the approach for 3D tracking of canonical Gaussians, as shown in Figure 5.3, to extract 3D trajectories from a novel view synthesis model. Comparing to *4D-GS* serves to show the impact of the changes made in the model architecture, the regularization terms, and learning per-Gaussian masks to arrive at DeformGS.

Training and Evaluation Setup We create a dataset of 6 dynamic cloth scenes, each with varying physical and visual properties (Figure 5.4). For DeformGS and 4D Gaussians, we perform 30,000 training iterations, and set point cloud pruning interval to 100, voxel plane resolution to [64, 64], and multi-resolution upsampling to levels $L = \{1, 2, 4, 8\}$. We set the regularization hyperparameters (Section 5.5.2) for all synthetic scenes to $\lambda_w = 2{,}000$, $\lambda^{\text{momentum}} = 0.03$, $\lambda^{\text{iso}} = 0.3$, and $k = 20$ for KNN. We keep all hyperparameters the same for real-world scenes, but increase the regularization terms for momentum and isometry loss. We generate the masks with Segment Anything (SAM) [150] for the initial frame, and use XMem [43] to propagate to future frames.

For DynaGS, we set $\lambda^{\text{rigid}} = 4$, $\lambda_w = 2{,}000$,$\lambda^{\text{iso}} = 2.0$, and $k = 20$, as in the open-source code.

We evaluate each compared method on 1,000 randomly sampled points on each cloth.

5.6.2 Simulation Results

3D Point Tracking Following prior work [199, 398], we report median trajectory error (MTE), position accuracy (δ), and the survival rate with a threshold of 0.5 [m] [199].

The results are summarized in Table 5.1. We make the following observations: (1) DeformGS outperforms baselines RAFT, DynaGS, and 4D-GS, by achieving a reduction in MTE of 55.8%–76.0% compared with the baselines. (2) The discrepancy between the RAFT oracle model and its averaged result demonstrates the difficulty arising from frequent self-occlusions. (3) The oracle models perform very well; this is in part thanks to the falling and short-horizon nature of these sequences, limiting self-occlusions. In the real-world we expect much larger errors due to noisy depth and more challenging occlusions in long-horizon tasks. It would also be unclear what viewpoint to choose without access to ground truth trajectories. This also points to future research avenues for additional supervision through optic flow and 2D tracking algorithms such as RAFT. (4) Scenes with less texture such as scene 3 perform significantly worse than scenes with strong texture.

Qualitative Results Figure 5.4 shows DeformGS results on all synthetic Blender scenes; Figure 5.5 shows ground truth and inferred trajectories in scene 5. The results show that especially DynaGS and 4D-GS introduce large errors as the cloth drapes down. RAFT improves over DynaGS and 4D-GS but requires accurate depth estimation.

5.6.3 Real-World Experiment Setup

Robo360 Data The Robo360 dataset [176] is a 3D omnispective multi-material robotic manipulation dataset. It covers many different scenarios, including manipulation by robot manipulators and humans captured by 86 calibrated cameras. These properties make it an ideal dataset to evaluate the effectiveness of DeformGS in the real world. We select three scenes: (1) a human folding a larger duvet, (2) a human folding a smaller cloth, and (3) a

Table 5.1: 3D tracking results on the deformable cloth dataset (Figure 5.4). For each metric, the methods above the solid line had access to privileged information, see [a][b] and Section 5.6.1 for more details. The results suggest that DeformGS outperforms the baselines in all averaged metrics, and is competitive with the oracle models. The results also suggest our novel deformation function architecture, learning per-Gaussian masks, and physics-inspired regularization losses improve the tracking performance compared to 4D-GS [354]. We do not consider the oracle methods to be fair baselines and therefore do not bold their results.

Metric	Method	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Mean
3D MTE [mm] ↓	RAFT [298] [a]	67.264	89.944	220.125	177.909	84.593	23.422	110.543
	RAFT Oracle [298][ab]	3.381	26.956	58.971	12.481	3.930	3.192	18.152
	OmniMotion Oracle [330] [ab]	0.535	14.513	39.958	4.556	2.487	2.011	10.677
	DynaGS [199]	24.233	81.119	464.64	54.074	34.985	36.101	115.859
	4D-GS [354]	4.645	95.032	223.839	27.441	14.091	11.619	62.778
	DeformGS	**3.373**	**45.33**	**88.369**	**14.022**	**7.257**	**8.173**	**27.754**
3D δ_{avg} ↑	RAFT [298] [a]	0.553	0.379	0.222	0.533	0.586	0.619	0.482
	RAFT Oracle [298][ab]	0.926	0.577	0.411	0.703	0.833	0.808	0.710
	OmniMotion Oracle [330] [ab]	0.987	0.693	0.549	0.864	0.885	0.849	0.805
	DynaGS [199]	0.624	0.31	0.042	0.435	0.533	0.527	0.4118
	4D-GS [354]	0.902	0.339	0.164	0.583	0.697	0.715	0.5667
	DeformGS	**0.929**	**0.522**	**0.322**	**0.71**	**0.856**	**0.787**	**0.688**
3D Survival ↑	RAFT [298] [a]	0.945	0.792	0.779	0.822	0.792	0.854	0.831
	RAFT Oracle [298][ab]	0.986	0.833	0.957	0.872	0.929	0.903	0.913
	OmniMotion Oracle [330] [ab]	1	0.963	1	0.985	0.963	0.933	0.977
	DynaGS [199]	0.999	0.99	0.483	0.988	0.992	0.992	0.907
	4D-GS [354]	**1**	0.967	0.834	**1**	0.999	**1**	0.967
	DeformGS	**1**	**1**	**1**	**1**	**1**	**1**	**1**

[a] Method had access to ground truth depth. [b] Method had access to ground truth trajectories to pick the best camera view.

robot arm folding a t-shirt. In the cloth folding scene, we exclude viewpoints where the entire person's body is visible to eliminate unnecessary complexity.

We also subsample the data to demonstrate DeformGS performance with fewer views. The duvet folding scene contains 17 training views, the cloth folding scene contains 20 training views, and the t-shirt folding robot scene was trained with 21 views.

5.6.4 Real-World Experiment Results

Real2Sim for Digital Twins Figure 5.6 shows the 3D tracking overlaid on rendered images, as well as the Gaussian points at each time step. The results suggest that DeformGS is able to successfully infer smooth and meaningful trajectories in the real world. While no ground truth is available, the trajectories appear to follow their geometry closely except for a few floating Gaussians. Hyperparameter tuning of the regularization functions, as well as discarding Gaussians with a low opacity, might help resolve this.

The point cloud included in this figure can be used to create a digital twin after recording

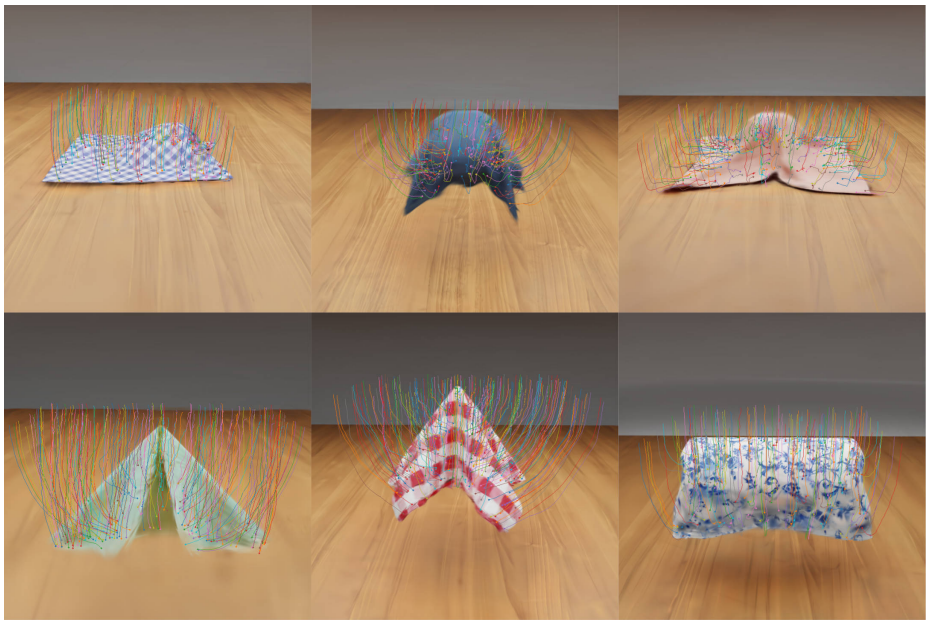
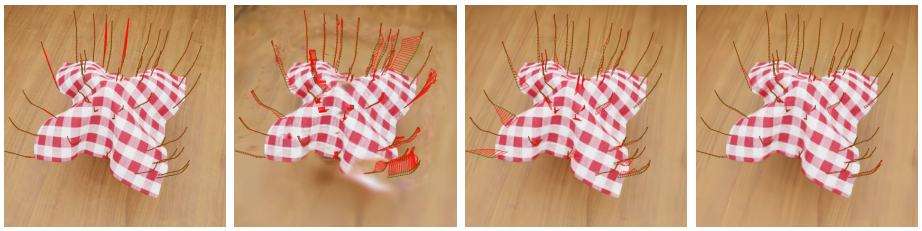


Figure 5.4: This figure shows the rendering and tracking of DeformGS in the six dynamic Blender scenes used for evaluation. We will refer to the scenes in this figure as Scenes 1, 2, 3, 4, 5 and 6 ordered from left to right, and top to bottom.



(a) RAFT [298] (b) DynaGS [199] (c) 4D-GS [354] (d) DeformGS

Figure 5.5: **Results on Scene 5**: randomly sampled ground-truth trajectories in green, inferred trajectories in red, and the error of corresponding points in red lines. Compared to the baseline methods, DeformGS results in smaller errors in 3D tracking.

the sequence. The digital twin of the duvet, and the entire environment, can then be used to create more dense supervision for imitation learning approaches.

Task-Relevant Keypoint Tracking Robotic manipulators can benefit from tracking task-relevant keypoints, such as the corner of a cloth or the edge of a jacket. Figure 5.7 shows a comparison between 4D-GS [354] and DeformGS in 3D point tracking, evaluated on both duvet and cloth scenes. The results suggest DeformGS leads to more smooth and overall useful trajectories. The trajectories from 4D-GS intertwine into more messy trajectories, and appear less physically plausible. This would hinder the adoption of 3D tracking into robot applications. Figure 5.1 shows DeformGS trajectories for a robot folding a t-shirt.

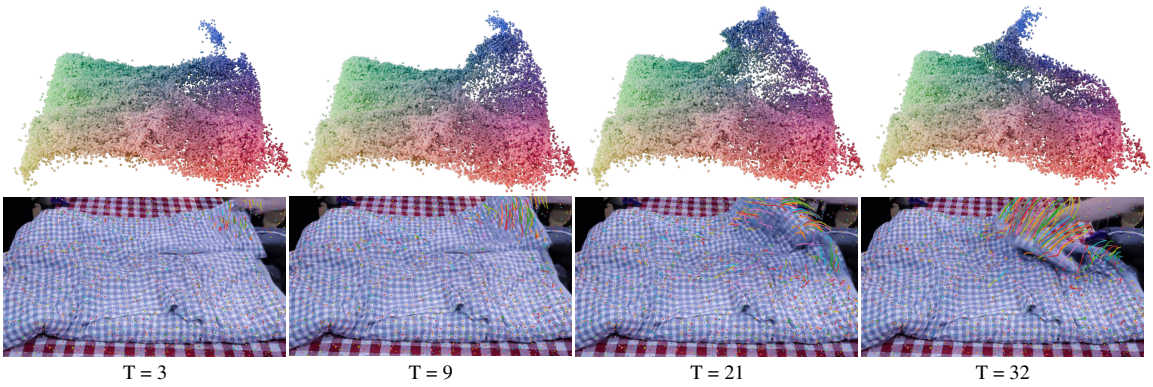


Figure 5.6: A person manipulating a duvet in the Robo360 [176] dataset, reconstructed using DeformGS. The top row shows the 4D Gaussians as point clouds, where the color represents dense correspondences. The bottom row shows rendered views overlaid with 3D trajectories projected to image space.



Figure 5.7: 3D point tracking on duvet (left) and cloth (right) folding scenes. Compared with 4D-GS (top), DeformGS (bottom) infers smoother, less tangled trajectories.

5.7 Conclusions

In this chapter, we address the challenging problem of 3D point-tracking in dynamic scenes with deformable objects. We introduced DeformGS, the first approach that learns continuous deformations for 3D tracking of highly deformable scenes. We empirically demonstrate that DeformGS outperforms baseline methods and achieves both high-quality dynamic scene reconstruction and high-accuracy 3D tracking on highly deformed cloth objects with occlusions and shadows, both in simulation and the real world. We also contribute a dataset of six synthetic scenes to facilitate future research.

Limitations and Future Work DeformGS, similar to prior work on dynamic novel view reconstruction, requires a setup of multiple synchronized and calibrated cameras, which may require a significant engineering effort in real-world scenarios. Additionally, significant innovation will be required to achieve the demonstrated results in real-time, as will be beneficial for scalable robot applications.

While DeformGS improves upon prior methods, we do observe Gaussians wandering off in some cases. We also notice the algorithm is relatively sensitive to the regularization hyperparameters ($\lambda^{\text{momentum}}$ and λ^{iso}); this might be resolved in the future by adding supervision from state-of-the-art point-tracking algorithms. These limitations point to promising directions for future research.

Acknowledgements This work was supported by the Center for Machine Learning and Health (CMLH) at CMU, and the Pittsburgh Supercomputing Center (PSC). We thank David Held for the productive discussions.

Chapter 6

Cloth-Splatting: 3D Cloth State Estimation from RGB Supervision

ORIGINALLY APPEARED AS

Alberta Longhini*, Marcel Büsching*, Bardienus P. Duisterhof, Jens Lundell, Jeffrey Ichnowski, Mårten Björkman, Danica Kragic.
* Equal contribution.
Cloth-Splatting: 3D Cloth State Estimation from RGB Supervision [195].
Conference on Robot Learning (CoRL), 2024.

Chapter in Context

DeformGS showed neural rendering can provide a powerful signal for 3D point tracking with multiple cameras. However, it required a large number of cameras, long training, and was sometimes fragile. This chapter presents Cloth-Splatting, which extends DeformGS by pre-training a generic cloth dynamics model, in addition to scene-specific fine-tuning. The result is better performance with fewer cameras and faster convergence. In SOLF, the learned dynamics prior makes cloth state more *obtainable* from fewer views and less optimization.

We introduce Cloth-Splatting, a method for estimating 3D states of cloth from RGB images through a prediction-update framework. Cloth-Splatting leverages an action-conditioned dynamics model for predicting future states and uses 3D Gaussian Splatting to update the predicted states. Our key insight is that coupling a 3D mesh-based representation with Gaussian Splatting allows us to define a differentiable map between the cloth's state space and the image space. This enables the use of gradient-based optimization techniques to refine inaccurate state estimates using only RGB supervision. Our experiments demonstrate that Cloth-Splatting not only improves state estimation accuracy over current baselines but also reduces convergence time by ~ 85 %. Code and videos available at: kth-rpl.github.io/cloth-splatting.

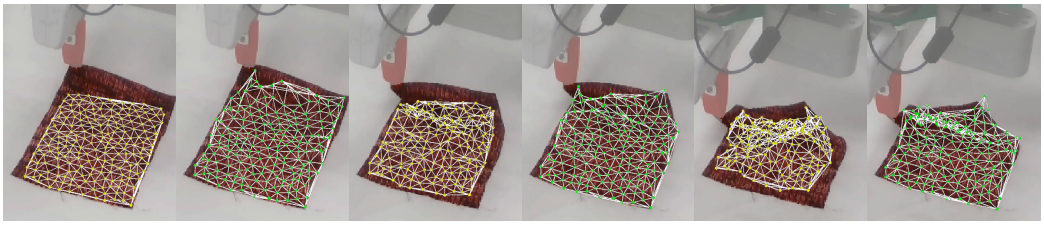
6.1 Introduction

Teaching robots to fold, drape, or manipulate deformable objects such as cloths is fundamental to unlocking a variety of applications ranging from healthcare to domestic and industrial environments [193]. While considerable progress has been made in rigid-object manipulation, manipulating deformables poses unique challenges, including infinite-dimensional state spaces, complex physical dynamics, and state estimation of self-occluded configurations [405]. Specifically, the problem of state estimation has led existing works on visual manipulation to either rely exclusively on 2D images, overlooking the cloth's 3D structure [7, 257, 368], or to use 3D representations that neglect valuable information in RGB observations [122, 181, 194].

Prior work on cloth state estimation often relies on 3D particle-based representations derived from depth sensors, including graphs [192, 258] and point clouds [288]. While point clouds effectively capture the object's observable state, they lack comprehensive structural information [194]. Alternative approaches rely on Graph Neural Networks (GNNs) trained on simulated data to predict graph representations using action-conditioned cloth dynamics [122, 181, 201]. However, these representations require online fine-tuning to bridge the sim-to-real gap. Using depth information to supervise the refinement of the cloth's 3D state estimates involves time-consuming simulation-based optimization techniques, as it lacks visual clues such as texture [123]. How to leverage RGB observations, which contain such information, remains an open problem.

We propose *Cloth-Splatting* (Figure 6.1), a method to estimate 3D states of cloth from RGB supervision by combining GS [145] with an action-conditioned dynamics model. The key idea of our method is to represent the 3D state of the cloth as a mesh and create a differentiable mapping between the cloth state space and the observation space using GS. This is achieved by populating the mesh faces with 3D Gaussians and expressing their positions relative to the mesh vertices. Given this, we can address the problem of estimating the 3D state of the cloth using a prediction-update framework akin to Bayesian filtering. Starting with a previous state estimate and a known robotic action, Cloth-Splatting predicts the next state using a learned dynamics model of the cloth. This prediction is then updated using RGB observations, leveraging the rendering loss provided by GS, allowing the refinement of the state estimate using visual clues such as texture and geometry.

We experimentally evaluate the 3D state estimation of cloth in simulated and real-world environments. Results show that Cloth-Splatting outperforms all 2D and 3D baseline tracking techniques, being $57\,\%$ more accurate and $\sim 85\,\%$ faster than the best-performing baseline. We further showcase how refining mesh-based representations enables closed-loop manipulation of cloths. Together, the results suggest that tracking deformable objects using only RGB observations is feasible, efficient, and accurate. The contributions of this chapter are 1) a mesh-constrained object-centric extension of GS; 2) a vision-supervised 3D state estimation for cloth with action-conditioned dynamics priors, and 3) experiments that suggest successful sim-to-real transfer.



(a) t=0 (b) t=7 (c) t=14

Figure 6.1: **Cloth-Splatting state estimation of a real-world cloth.** We introduce Cloth-Splatting, a 3D state estimation method for cloth from sparse RGB observations that combines GNN with GS. Given an initial sequence of 3D mesh predictions from the GNN (left), Cloth-Splatting (right) updates it using RGB observations, achieving state-of-the-art 3D tracking performance for deformables.

6.2 Related Work

Cloth Manipulation. Manipulating cloth in robotics involves tasks like folding, smoothing, lifting, and inserting objects into deformable bags [7, 257, 269, 368]. Manipulation approaches can be categorized into model-free methods, mapping cloth states or sensory data to actions [205, 339, 358], and model-based methods, utilizing cloth models to devise manipulation strategies [116, 123, 213]. Model-based approaches often rely on 3D representations focusing on optimizing pick-and-place actions. However, optimizing manipulation between pick and place in a closed-loop manner is crucial for handling planning errors and object variability [96, 113, 170, 194, 235], but remains underexplored, largely due to the challenges of real-world cloth state estimation [123]. This chapter addresses this gap, enabling feedback-loop manipulations using graph representations.

Cloth State Estimation. Cloth state estimation is a well-studied area in robotics, computer vision, and computer graphics [256, 283]. While vision-based methods for 2D and 3D cloth state estimation have been developed for static observations of cloths lying flat on a surface [44, 52, 133, 169], tracking these estimates over time is challenging due to the complex non-linear dynamics of deformables. A first solution to this problem proposes a self-supervised method that leverages an action-conditioned dynamics model of the cloth and test-time optimization to refine 3D state representations from point cloud observation [123]. However, this method requires aligning real-world states with simulated ones and disregards informative feedback provided by RGB observations. Unlike previous methods, the method presented in this chapter refines the 3D cloth states using RGB feedback. This is made possible by a differentiable GS map linking 3D states to image observations.

Vision-based tracking. Tracking points in 2D image space over time is a widely studied problem [62, 139, 272, 298, 330]. However, 3D visual trackers [300, 384] have gained popularity more recently, as they better address the occlusion challenges inherent in 2D tracking. Extensions of NeRF [209] to non-static scenes [81, 173, 229, 239] already allow us to track the 3D motion of dynamic scene content, but often lack applicability to real-world scenarios [91]. 3D Gaussian Splatting [145] based methods, like Dynamic 3D Gaussians (DynaGS) [199], track by explicitly modeling the position and covariance of each Gaussian over time. In contrast, 4DGS [354] learns a continuous deformation field to track Gaussian displacements. MD-Splatting [70] builds on 4DGS for cloth tracking using a shadow network and physics-inspired regularization. DeformGS [71] extends MD-Splatting by learning object-centric masks, and simplifying the regularization terms. However, these methods require dense observations (at least 50 cameras for MD-Splatting and DeformGS) and involve computationally costly per-scene optimizations (see Figure 6.4).

Mesh-constrained GS. Similar to Cloth-Splatting, several methods combine meshes and 3D Gaussians for deformable object representation [92, 136, 319]. However, like PhysGaussian [364], they rely on explicitly modeled dynamics and lack refinement with visual observations, limiting their tracking capabilities.

6.3 Problem Formulation

The problem addressed in this chapter is estimating the 3D state of a cloth $\mathbf{M}_{t+1}$ at time $t+1$ given the observations $\mathbf{Y}_{1:t+1}$ and the agent's actions $\mathbf{a}_{1:t}$. The state of the cloth is represented as an augmented mesh $\mathbf{M}_t = (\mathbf{V}_t,\ \dot{\mathbf{V}}_t,\ \mathbf{E}_t)$, where $\mathbf{V}_t \in \mathbb{R}^{\mathrm{N}\times 3}$ are the vertex positions, $\dot{\mathbf{V}}_t \in \mathbb{R}^{\mathrm{N}\times 3}$ are the velocities of the vertices, and $\mathbf{E}_t \in \mathbb{Z}_+^{\mathrm{L}\times 2}$ the edges. Only $\mathbf{V}_t$ and $\dot{\mathbf{V}}_t$ are estimated while the edges $\mathbf{E}_t$, that form a triangular mesh structure for connectivity, remain constant, i.e., $\mathbf{E}_0 = \mathbf{E}_1 = \cdots = \mathbf{E}_t$. The observations $\mathbf{Y}_{t+1} = \left\{\mathbf{I}_{t+1}^0,\ \ldots,\ \mathbf{I}_{t+1}^{\mathrm{K}}\right\}$ are a set of K multi-view RGB images $\mathbf{I}_{t+1}^{\mathrm{k}} \in \mathbb{R}^{\mathrm{w}\times\mathrm{h}\times 3}$ with unique camera matrices $\mathbf{P} = \{\mathbf{P}^0,\ \ldots,\ \mathbf{P}^K\}$ where $\mathbf{P}^k \in \mathbb{R}^{4\times 4}$. The actions $\mathbf{a}_t \in \mathbb{R}^3$ are Cartesian end-effector velocities.

We reframe this estimation problem as a Bayesian filtering problem, where the goal is to infer the joint posterior distribution $p(\mathbf{M}_{t+1}|\mathbf{Y}_{1:t+1}, \mathbf{a}_{1:t})$ recursively over time with a general prediction-update framework. The prediction step is assumed to have access to a transition probability function $p(\mathbf{M}_{t+1}|\mathbf{M}_t, \mathbf{a}_t)$, which allows us to compute a prior $p(\mathbf{M}_{t+1}|\mathbf{Y}_{1:t}, \mathbf{a}_{1:t})$ of the state of the system at time $t+1$:

$$p(\mathbf{M}_{t+1}|\mathbf{Y}_{1:t}, \mathbf{a}_{1:t}) = \int p(\mathbf{M}_{t+1}|\mathbf{M}_t, \mathbf{a}_t) p(\mathbf{M}_t|\mathbf{Y}_{1:t}, \mathbf{a}_{1:t-1}) d\mathbf{M}_t, \tag{6.1}$$

provided the history of observations $\mathbf{Y}_{1:t+1}$ and the agent's actions $\mathbf{a}_{1:t}$. Given a new observation $\mathbf{Y}_{t+1}$, the update of the prior estimate is proportional to the product of the measurement likelihood $p(\mathbf{Y}_{t+1}|\mathbf{M}_{t+1})$ and the predicted state. Thus, the joint posterior

distribution is obtained as follows:

$$p(\mathbf{M}_{t+1}|\mathbf{Y}_{1:t+1}, \mathbf{a}_{1:t}) = \frac{1}{\eta} p(\mathbf{Y}_{t+1}|\mathbf{M}_{t+1}) p(\mathbf{M}_{t+1}|\mathbf{Y}_{1:t}, \mathbf{a}_{1:t}), \tag{6.2}$$

where η is a normalization constant, ensuring that the posterior distribution integrates to 1. The state estimation problem then reduces to solving (6.2).

6.4 Method

Solving the problem of cloth 3D state estimation through a prediction-update framework requires modeling two key components: the transition probability function $p(\mathbf{M}_{t+1}|\mathbf{M}_t, \mathbf{a}_t)$ for the prediction step and the measurement likelihood $p(\mathbf{Y}_{t+1}|\mathbf{M}_{t+1})$ for the update step.

To handle the non-trivial deformable object dynamics, we approximate the transition probability function with a deterministic GNN $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, which is described in more detail in Section 6.4.1. This GNN allows us to predict the state of the cloth at time $t+1$, denoted as $\hat{\mathbf{M}}_{t+1}$.

For the update step, we model the measurement likelihood $p(\mathbf{Y}_{t+1}|\mathbf{M}_{t+1})$ using a measurement model $h(\mathbf{Y}_{t+1}|\mathbf{M}_{t+1})$, which maps the predicted state $\hat{\mathbf{M}}_{t+1}$ to the predicted observation $\hat{\mathbf{Y}}_{t+1}$. The likelihood is then expressed as a function of the measurement error $||\mathbf{Y}_{t+1} - \hat{\mathbf{Y}}_{t+1}||_2^2$. Still, modeling the highly non-linear measurement model h that maps from the state space of the cloth to the image space is challenging. The key insight of this chapter is to approximate the measurement model using GS, such that $\mathbf{Y}_{t+1} \approx h_{\text{GS}}(\mathbf{M}_{t+1}, \mathbf{P})$. Details on h_{GS} are provided in Section 6.4.2.

By combining the transition model $f_{\boldsymbol{\theta}}$ and the measurement model h_{GS}, we create a differentiable mapping between the cloth's state space and the observation space. This allows us to obtain the updated state $\tilde{\mathbf{M}}_{t+1}$ via gradient-based optimization by minimizing the measurement error in the observation space:

$$\mathcal{L}_{obs} = ||\mathbf{Y}_{t+1} - h_{\text{GS}}(\tilde{\mathbf{M}}_{t+1}, \mathbf{P})||_2^2, \tag{6.3}$$

as described in Section 6.4.3.

An overview of the full method is visualized in Figure 6.2, showing the prediction-update framework proposed in this chapter.

6.4.1 State Prediction

We model the cloth's action-conditioned transition function $f_{\boldsymbol{\theta}}$ using a GNN parameterized by $\boldsymbol{\theta}$ that is designed to effectively handle the mesh structure and complex cloth dynamics [258]. The input to the GNN consists of the last m states of the mesh $\mathbf{M}_{t-m:t}$ and the external robot action $\mathbf{a}_t$. To condition the GNN to the action, we assume the grasped particle is rigidly attached to the gripper, directly updating its position and velocity, similar to Wang et al. [338]. This constraint allows $f_{\boldsymbol{\theta}}$ to dynamically predict the next state of the cloth

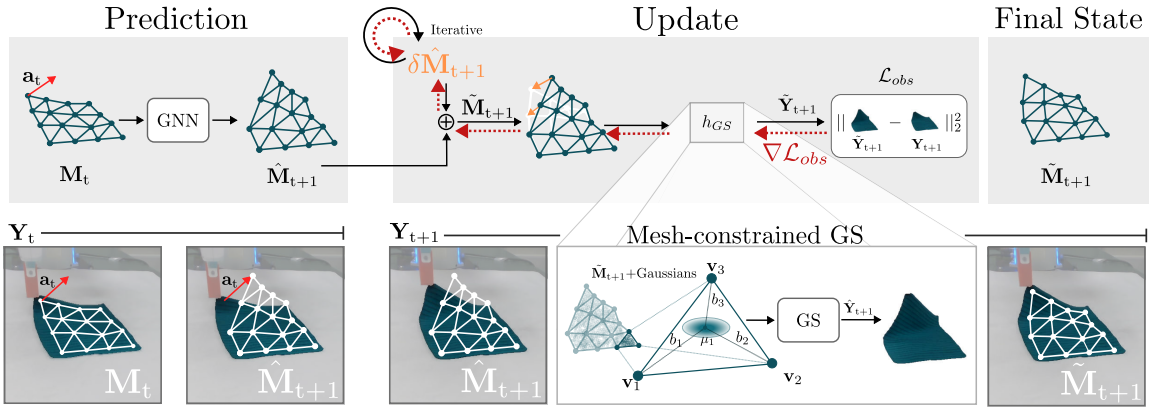


Figure 6.2: **Overview of Cloth-Splatting.** We estimate the 3D state of the cloth through a prediction-update framework. We use a graph neural network (GNN) to approximate the transition function that estimates the next state $\hat{\mathbf{M}}_{t+1}$ given an action $\mathbf{a}_t$ and the state $\mathbf{M}_t$. We then employ our proposed mesh-constrained Gaussian Splatting (GS) as a measurement model h_{GS} to obtain an estimate of the observation $\tilde{\mathbf{Y}}_{t+1}$ given the current refined state $\tilde{\mathbf{M}}_{t+1} = \hat{\mathbf{M}}_{t+1} + \delta\hat{\mathbf{M}}_{t+1}$. The differentiability of GS allows us to iteratively optimize the update $\delta\hat{\mathbf{M}}_{t+1}$ of the state estimate via the photometric consistency loss between the ground-truth observation $\mathbf{Y}_{t+1}$ and the rendered state $\tilde{\mathbf{Y}}_{t+1}$.

$\hat{\mathbf{M}}_{t+1} = f_{\boldsymbol{\theta}}(\mathbf{M}_{t-m:t}, \mathbf{a}_t)$ by propagating the effects of actions across the graph structure. Specifically, the network predicts the acceleration of each node, which we integrate using a forward-Euler integrator to compute the next velocity and position of each vertex. We train $f_{\boldsymbol{\theta}}$ on a one-step Mean Squared Error (MSE) loss between the predicted and ground-truth meshes.

6.4.2 Mesh-constrained Gaussian Splatting

We utilize GS as the measurement model $\hat{\mathbf{Y}}_{t+1} = h_{\text{GS}}(\hat{\mathbf{M}}_{t+1}, \mathbf{P})$, where h_{GS} is a differentiable map between the mesh representation and the image space.

Background. GS synthesizes images by projecting the centers $\boldsymbol{\mu}$ and covariances $\boldsymbol{\Sigma}$ of 3D Gaussians to the image plane and then uses α-blending to aggregate their colors $\mathbf{c}$ to pixel colors. To enforce the covariance matrix $\boldsymbol{\Sigma}$ to be positive and semi-definite, GS decomposes it into a rotation $\mathbf{R}$ and scale $\mathbf{S}$ for each Gaussian:

$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}. \tag{6.4}$$

Given the camera matrix $\mathbf{P}$, the covariance matrix can be projected into image space as:

$$\boldsymbol{\Sigma}' = \mathbf{J}\mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}, \tag{6.5}$$

where $\mathbf{J}$ is the Jacobian of the affine approximation of the projective transformation. During rendering, the color $\mathbf{c}^p$ of a pixel is computed by blending N ordered Gaussians overlapping the pixel:

$$\mathbf{c}^p = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{6.6}$$

where $\mathbf{c}_i$ is the color of each Gaussian and α_i is given by evaluating the projected 2D Gaussian with covariance multiplied with the learned opacity of each Gaussian.

Mesh-constrained GS. To connect GS to the mesh representation, we assign each Gaussian to a face of the initial mesh. The mean position $\boldsymbol{\mu}$ of each Gaussian is then defined in linear barycentric coordinates:

$$\mu = b_1 \mathbf{v}^1 + b_2 \mathbf{v}^2 + b_3 \mathbf{v}^3, \tag{6.7}$$

where $\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3 \in \mathbf{V}$ are the vertices of the assigned face and $b_1 + b_2 + b_3 = 1$. The rotation $\mathbf{R}$ of the Gaussian is then expressed relative to the assigned face. During a mesh deformation, the parameters of the Gaussians and the barycentric coordinates remain constant and represent the cloth's time-invariant appearance.

6.4.3 State Update

The goal of the state update is to improve the state prediction $\hat{\mathbf{M}}_{t+1}$ based on the observation $\mathbf{Y}_{t+1}$ such that the updated state $\tilde{\mathbf{M}}_{t+1}$ minimizes the measurement error (Equation 6.3).

One option to refine the state would be to directly optimize the transition function $f_{\boldsymbol{\theta}}$ so it yields $\tilde{\mathbf{M}}_{t+1}$ as an improved state estimate. However, $f_{\boldsymbol{\theta}}$ requires recursive roll-out for estimating future states. Coupled with the internal propagations of the GNN, this can lead to vanishing gradients.

To circumvent the previous issues, we instead define the updated state as the sum:

$$\tilde{\mathbf{M}}_{t+1} = \hat{\mathbf{M}}_{t+1} + \delta\hat{\mathbf{M}}_{t+1}, \tag{6.8}$$

where $\delta\hat{\mathbf{M}}_{t+1} = u_{\boldsymbol{\psi}}(t+1)$ is a learned residual state update parameterized by $\boldsymbol{\psi}$. The residual map $u_{\boldsymbol{\psi}}$ is modeled as a Multi-layer Perceptron (MLP), mapping the time to an offset from the state prediction, and is optimized using gradient descent on $\mathcal{L}_{obs}$. Intuitively, this approach keeps the prior estimate from the GNN fixed, refining the cloth state by minimizing the discrepancy between the observed image and the rendered cloth state. The residual map is optimized per scene with the initial output set close to zero. We also add the regularization loss $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{SSIM}} + \mathcal{L}_{\text{iso}} + \mathcal{L}_{\text{magn}}$, where $\mathcal{L}_{\text{SSIM}}$ is the Structural Similarity Index Measure (SSIM) loss [340], $\mathcal{L}_{\text{iso}}$ ensures an As-Rigid-As-Possible (ARAP) behavior [123, 279] in the cloth, and $\mathcal{L}_{\text{magn}}$ minimizes overall motion for numerical stability. Details on the regularization and the mesh-constrained GS appear in the supplementary material of the original publication [195].

6.5 Experimental Evaluation

The primary objective of the experiments is to evaluate how quickly and accurately Cloth-Splatting can estimate 3D cloth states from RGB images compared to other baselines. We further ablate different components of our method, provide qualitative results, and demonstrate the feasibility of using Cloth-Splatting for robotic cloth manipulation in both simulated and real-world scenarios.

6.5.1 Experimental Set-up

Dataset Generation. Due to the lack of ground-truth states in real-world scenarios, we generated a synthetic dataset composed of 75 different scenes to conduct quantitative evaluations. We included three different object categories: `TOWEL`, `SHORTS`, and `TSHIRT` with five mesh variations of each object and five different trajectories per variation. The mesh variations were generated following Lips et al. [185]. Each trajectory consists of random interactions with the cloths, performed using the NVIDIA Flex simulator [180, 202]. We recorded RGB-D images from 4 different camera views for each scene and time step, where the depth is only used for the baseline methods. We used Blender to render photo-realistic scenes. We further record robot actions and ground-truth cloth states.

Action-conditioned learned dynamics and mesh initialization. We implemented the action-conditioned models using the Graph Network Simulator (GNS) architecture proposed in [258], which consists of a GNN with a node encoder, a processor, and a decoder. The input to the model is a mesh reconstructed from a 3D point cloud of the cloth at t $= 0$. We use Delaunay triangulation to reconstruct the mesh [55]. The point cloud can be obtained from either depth observations or multi-view stereo reconstructions [335]. We trained the model exclusively on the `TOWEL` object and split the dataset into a training, validation, and test set.

Gaussian Splatting and Residual Model. For initialization, two randomly parameterized Gaussians are positioned on each face of the mesh, by sampling their barycentric coordinates from the distribution $\mathcal{N}(1/3, 0.05)$ and normalizing them so that they add up to 1. To model the appearance of the cloth, we first optimize the Gaussians using observations from $t = 0$ and the measurement loss $\mathcal{L}_{\text{obs}}$ without regularization for the first 1.5k iterations. Subsequently, we jointly optimize the Gaussians and the residual network for an additional 5.5k – 6.5k iterations for learning the residual deformation and previously unseen parts of the cloth, such as a differently colored backside.

Update. We implement two different update procedures. The `ITERATIVE` update predicts one step ahead, refines the prediction with GS, and then uses the refined state as input to the GNN for the next prediction. In contrast, `ROLLOUT` predicts future states by unrolling the GNN over the entire trajectory and then refines all states simultaneously with GS, resulting in a faster overall runtime. We apply `ROLLOUT` in the tracking experiments and `ITERATIVE` in the manipulation experiments. The supplementary material of the original publication [195] provides a runtime and accuracy comparison between the approaches.

6.5.2 Quantitative Results

To quantify the tracking performance, we use: *median trajectory error* (MTE) [398], which measures the distance between the estimated cloth and ground-truth tracks; *position accuracy* (δ) [62], which measures the percentage of tracks within the pre-defined distance thresholds 10, 20, 40, 80, and 160 mm to the ground truth; and the *survival rate*, which assesses the average number of frames until the tracking error exceeds a predefined threshold [398], which we set to 50 mm.

We compare Cloth-Splatting to these baselines: Dynamic 3D Gaussians (DynaGS) [199], which models each Gaussian's position and rotation; and MD-Splatting [70], which extends GS by projecting non-metric Gaussians into a metric space to track deformable objects better. We also include the unrefined GNN predictions to establish a performance baseline, as well as a 2D tracking method obtained by evaluating RAFT [298] on all views and then reporting only the trajectories from the view with the lowest MTE. We denote this as RAFT-Oracle as it has access to the ground-truth cloth state to select the best-performing view.

6.5.2.1 Comparative Tracking Evaluation

We assess the tracking accuracy of Cloth-Splatting across the 75 synthetic scenes against the baselines. The results in Table 6.1 show Cloth-Splatting outperforms the next best baseline MD-Splatting by 9.66 % in MTE on average, while also having the best overall position accuracy δ_{avg} and survival rate.

All methods show a varying performance between the different scenes, with all methods consistently being the worst on `SHORTS` which contain the largest, self-occluded deformation of the cloth with an example shown in Figure 6.3. Analyzing the unrefined GNN predictions shows a competitive performance, even on the shapes which are out-of-distribution from the training data, indicating that the GNN generalizes across different geometries.

Table 6.1: **Quantitative evaluation.** Comparison of Cloth-Splatting and the baselines in tracking quality. We report the mean and the standard deviation per metric ($\mu \pm \sigma$) and mark the best result **bold**.

Metric	Method	SHORTS	TOWEL	TSHIRT	Mean
3D MTE ↓ [mm]	RAFT-Oracle [298]	26.511 ± 33.854	17.951 ± 12.523	8.062 ± 14.990	18.324 ± 23.821
	DynaGS [199]	15.987 ± 17.972	7.352 ± 3.265	9.574 ± 4.324	10.924 ± 11.246
	MD-Splatting [70]	7.043 ± 9.265	1.755 ± 2.711	**2.109** ± 2.907	3.635 ± 6.235
	GNN	17.388 ± 15.824	9.157 ± 7.292	14.826 ± 12.295	13.853 ± 12.674
	Cloth-Splatting	**4.928** ± 5.240	**1.703** ± 1.213	3.159 ± 2.818	**3.284** ± 3.722
3D δ_{avg} ↑	RAFT-Oracle [298]	0.660 ± 0.166	0.661 ± 0.114	0.744 ± 0.133	0.683 ± 0.143
	DynaGS [199]	0.773 ± 0.141	0.829 ± 0.076	0.808 ± 0.083	0.804 ± 0.105
	MD-Splatting [70]	0.816 ± 0.098	0.870 ± 0.059	0.855 ± 0.082	0.847 ± 0.083
	GNN	0.720 ± 0.128	0.791 ± 0.076	0.731 ± 0.141	0.747 ± 0.121
	Cloth-Splatting	**0.851** ± 0.075	**0.879** ± 0.057	**0.858** ± 0.080	**0.862** ± 0.072
Survival rate ↑	RAFT Oracle [298]	0.666 ± 0.170	0.734 ± 0.119	0.752 ± 0.132	0.715 ± 0.145
	DynaGS [199]	0.795 ± 0.154	0.860 ± 0.072	0.850 ± 0.092	0.835 ± 0.113
	MD-Splatting [70]	0.869 ± 0.085	0.910 ± 0.059	0.881 ± 0.082	0.887 ± 0.077
	GNN	0.771 ± 0.138	0.860 ± 0.081	0.770 ± 0.138	0.800 ± 0.128
	Cloth-Splatting	**0.917** ± 0.067	**0.927** ± 0.059	**0.888** ± 0.084	**0.910** ± 0.072

6.5.2.2 Model Ablation

We report ablation studies on the mesh-constrained component of Cloth-Splatting in Table 6.2 on a SHORTS scene. In the following, we refer to each ablation by the corresponding identifier, e.g., (A1).

Using the state predictions of the GNN without the GS update (A1) results in a high tracking error, likely due to the previously unobserved shape of the cloth. Similarly, using Cloth-Splatting without the GNN (A2) and relying only on the initial mesh state for initialization leads to poor tracking accuracy, as the model must learn the entire deformation just from visual observations. Additionally, removing the regularization terms (A3) significantly reduces accuracy as it fails to enforce cloth structural properties. Consequently, parts of the cloth stretch into unnatural states to satisfy the visual observations rather than maintaining the underlying shape. Interestingly, reducing the number of camera views (A6, A7) does not degrade the performance as much as other ablations, although we experience a larger drop in performance for a monocular setup (A5).

6.5.2.3 Time Ablation

To quantify how the cloth model prior affects convergence time, we compare the MTE of Cloth-Splatting to the best-performing baseline throughout their training durations. The experiments were performed on an NVIDIA 4090 GPU and an Intel i9-14900K processor. Figure 6.4 shows that Cloth-Splatting achieves faster convergence times than the MD-Splatting and overall lower tracking error. The efficiency of Cloth-Splatting not only reduces

Table 6.2: **Model ablation.**

Ablation		3D MTE [mm]
(A1)	Only GNN	21.552
(A2)	No GNN	16.135
(A3)	No $\mathcal{L}_{reg}$	15.772
(A4)	No $\mathcal{R}_t$	10.799
(A5)	1 view	16.525
(A6)	2 views	9.535
(A7)	3 views	9.004
Full (4 views)		**8.923**

computational costs but also enables faster adaptation to new scenarios, making it a more practical choice for real-world online applications.

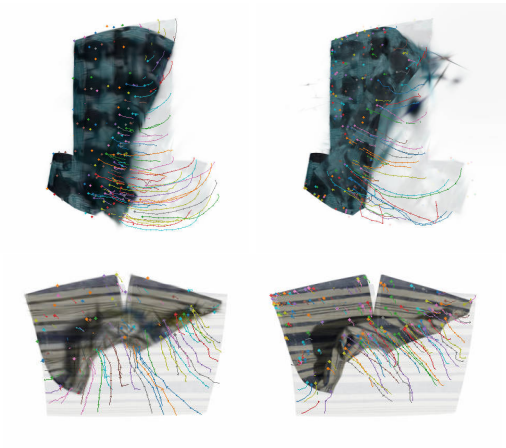
6.5.3 Qualitative Results

We present simulated qualitative rendering and tracking results in Figure 6.3 and compare these to MD-Splatting. Both methods capture the underlying dynamics of the scene. Nevertheless, the tracking results of MD-Splatting are negatively influenced by occasional tracking of visual artifacts. Although high-quality renderings are not the goal of Cloth-Splatting, the results show that the general visual appearance of the observed cloth is reproduced faithfully.

To showcase the transferability of Cloth-Splatting to the real world, we qualitatively evaluate the tracking of a real cloth folded by a 7-DOF Franka Emika Panda robot, observed by 3 extrinsically calibrated RealSense D435 cameras from which we only collect RGB images. Figure 6.1 shows qualitative real-world results for Cloth-Splatting and the GNN predictions. While the GNN predictions partially predict the correct mesh structure, they suffer from compounding errors from the roll-out predictions. Subsequently, Cloth-Splatting successfully updates the mesh to better model the cloth shape.

6.5.4 Robotic Manipulation Use Case

We further showcase that our state estimation process enables closed-loop optimization of folding trajectories using graph state representations. We focus on the half-folding task from Garcia-Camacho et al. [95], optimizing the trajectory between predefined pick-and-place positions. Our method refines the cloth's state estimate during each re-planning step, using model-predictive control (MPC) for planning and Cloth-Splatting for refinements (`MPC-CS`). We compare against a baseline using a predefined linear trajectory (`Fixed`), an open-loop (`MPC-OL`) baseline, and an oracle baseline (`MPC-ORACLE`) with ground-truth state access. The results in Table 6.3 show that our method outperforms all baselines, achieving results



(a) Cloth-Splatting (b) MD-Splatting

Figure 6.3: **Qualitative results (sim).** Tracking and rendering results on SHIRT (top) and SHORTS (bottom) scenes.

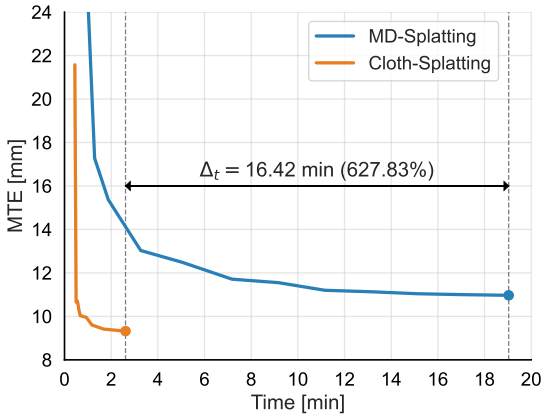


Figure 6.4: **Time ablation.** Convergence times of Cloth-Splatting and MD-Splatting on a SHORTS scene. The offset to the y-axis arises from the GNN predictions (1.36 s) and the 3D Gaussian initialization (26.55 s).

comparable to those of the oracle, underscoring the effectiveness of our method in refining state estimates for model-based closed-loop manipulation.

Table 6.3: **Manipulation results.** We report the mean and standard deviation of the MSE computed between the final cloth state and the goal state. Results are presented in units of 10^{-3}.

Method	TOWEL	TSHIRT
FIXED	2.2 $\pm$ 0.4	2.4 $\pm$ 0.4
MPC-OL	1.8 $\pm$ 2.1	7.3 $\pm$ 5.2
MPC-CS (us)	0.6 $\pm$ 0.6	1.2 $\pm$ 0.8
MPC-ORACLE	0.4 $\pm$ 0.2	0.8 $\pm$ 0.5

6.6 Conclusions

We introduced Cloth-Splatting, an approach for 3D state estimation of cloths using RGB supervision. Cloth-Splatting integrates an action-conditioned cloth dynamics model with Gaussian Splatting to enhance the accuracy of 3D state representations based solely on RGB feedback using sparse (3-4) camera views. We experimentally validated the quality of these estimates and showcased the computational efficiency of our framework. These advancements position Cloth-Splatting as a more practical choice for real-world applications compared to existing baselines.

Limitations. While Cloth-Splatting shows significant improvements over the baselines, its speed is still insufficient for real-time applications. Additionally, the need for a calibrated multi-camera set-up may present challenges in real-world scenarios, although scalable computer vision techniques for calibration, such as those in DUSt3R [335], can help mitigate this issue. Since Cloth-Splatting assumes a static visual appearance, dynamic appearance changes, such as shadows or changing lighting conditions, can occasionally lead to the tracking of visual artifacts. Furthermore, the initial mesh initialization requires occlusion-free observations of the cloth. To address this limitation, recent advances in template-based reconstruction [336] of crumpled cloth present a promising direction for future research.

Part II

Generative Spatiotemporal Priors

Chapter 7

RaySt3R: Predicting Novel Depth Maps for Zero-Shot Object Completion

ORIGINALLY APPEARED AS

Bardienus P. Duisterhof, Jan Oberst, Bowen Wen, Stan Birchfield, Deva Ramanan, Jeffrey Ichnowski.
RaySt3R: Predicting Novel Depth Maps for Zero-Shot Object Completion [73].
Conference on Neural Information Processing Systems (NeurIPS), 2025.

Chapter in Context

Thus far we have reconstructed observed structure in static and dynamic scenes. But humans understand viable object structure from just a single viewpoint. Can we instill a similar capability into robots? With RaySt3R, we formulate object completion as novel view synthesis by predicting novel depth maps. In SOLF, predicting geometry rather than appearance provides a more *focused* objective for learning object shape.

3D shape completion has broad applications in robotics, digital twin reconstruction, and extended reality (XR). Although recent advances in 3D object and scene completion have achieved impressive results, existing methods lack 3D consistency, are computationally expensive, and struggle to capture sharp object boundaries. The method presented in this chapter (RaySt3R) addresses these limitations by recasting 3D shape completion as a novel view synthesis problem. Specifically, given a single RGB-D image and a novel viewpoint (encoded as a collection of query rays), we train a feedforward transformer to predict depth maps, object masks, and per-pixel confidence scores for those query rays. RaySt3R fuses these predictions across multiple query views to reconstruct complete 3D shapes. We evaluate RaySt3R on synthetic and real-world datasets, and observe it achieves state-of-the-art performance, outperforming the baselines on all datasets by up to 44 % in 3D Chamfer distance. Project page: `rayst3r.github.io`

This work was generously supported by the Center for Machine Learning and Health (CMLH) at CMU, the NVIDIA Academic Grant Program, and the Pittsburgh Supercomputing Center.

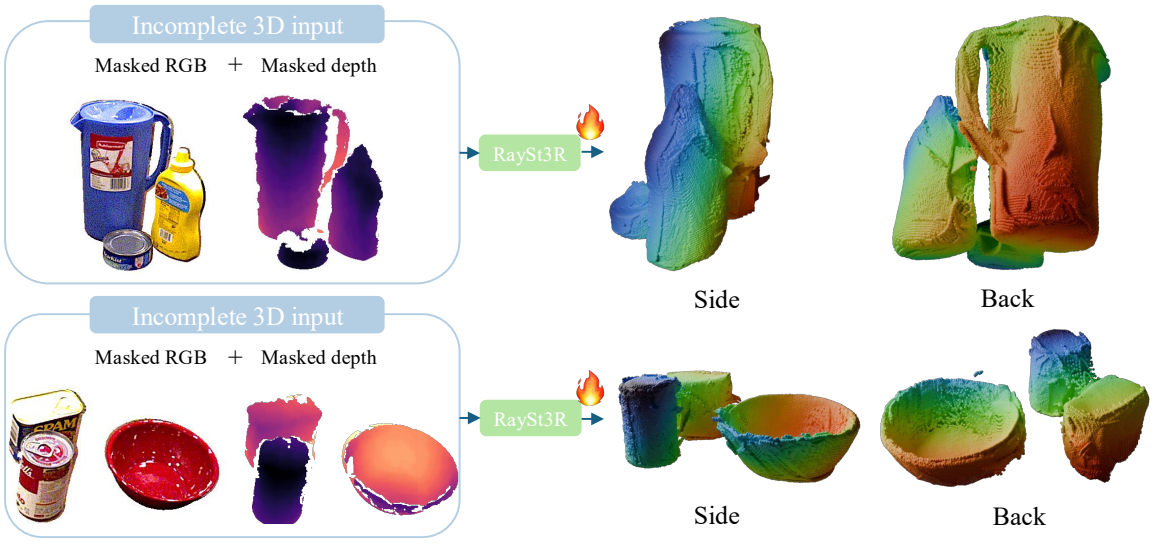


Figure 7.1: RaySt3R is a method for zero-shot 3D shape completion from a single foreground-masked RGB-D image. It predicts depth maps, object masks, and per-pixel confidence scores for novel viewpoints, and fuses them to reconstruct a complete 3D shape. RaySt3R is able to recover the geometry of full objects in cluttered real-world scenes, despite only being trained on synthetic data.

7.1 Introduction

3D shape completion is an enabling tool for visual reasoning and physical interaction with partially visible objects, and facilitates a wide range of downstream tasks such as robot grasping in cluttered environments [204, 346], obstacle avoidance [211, 294], mechanical search [119], digital-twin reconstruction, and Augmented or Virtual or Extended Reality (AR/VR/XR) applications.

Challenges. We focus on the robotics-driven setting where an RGB-D image is provided as input for multi-object shape completion. While object-centric methods achieve high reconstruction quality for single objects, multi-object scenes require instance segmentation and alignment procedures [1] that tend to be brittle in practice. Generative approaches use 2D image generation models [341, 381] to generate images from novel viewpoints, but these can be sensitive to large viewpoint changes and are computationally inefficient at inference time, which hinders robot and XR deployment. Other methods scale up 3D prediction on abundant synthetic scene data [129], but the resolution for the 3D representations (such as 3D MAE voxel grids) is too coarse to capture sharp object shapes with high-frequency geometry details.

Approach. We propose Ray Stereo 3D Reconstruction (RaySt3R), a novel method for addressing the above challenges. Given a single masked RGB-D image as input, our key insight is to recast shape completion as a novel view-synthesis task, then aggregate multiple view predictions to generate a complete 3D shape. Our approach draws inspiration from recent work that casts 3D reconstruction as point map regression via multi-view transformers [68, 335]. We similarly use a vision transformer (ViT) [64] architecture defined over visual DINOv2 [225] features extracted from the input image. However, instead of requiring a second image as an additional input, we input the novel view to be synthesized in the form of a camera ray map. Specifically, RaySt3R is trained to predict depth maps, confidence maps, and foreground masks for each queried ray via cross-attention. We then merge RaySt3R's geometric predictions from multiple novel views using the per-ray confidence and mask predictions.

Data. Since RaySt3R can be seen as a view-synthesis engine, we can train at scale on pairs of RGB-D images without requiring volumetric 3D supervision (as required by prior work [129]). We train RaySt3R on a large-scale augmented synthetic dataset with 1.1 million scenes and 12 million views. Across synthetic and real-world benchmarks, RaySt3R outperforms prior art by up to 44 % (in shape completion accuracy). Despite never being trained on real data, RaySt3R generalizes well to real-world cluttered scenes.

This chapter contributes:

- RaySt3R, a method for view-based 3D shape completion that learns confidence-aware depth maps and object masks from novel views, and uses a novel formulation of merging multi-view prediction.
- A new curated large-scale dataset with 12 million novel depth maps and masks, which we will open-source to facilitate future research.
- Evaluations of RaySt3R on synthetic and real-world datasets that show RaySt3R achieves state-of-the-art accuracy for 3D shape completion, and successfully generalizes to real-world cluttered scenes after training on only synthetic data.

7.2 Related work

3D shape completion has seen impressive progress over the last years. We explore related works categorized by their reasoning space, i.e., volumetric approaches and view-based approaches.

Volumetric reasoning Volumetric methods operate directly in 3D space and provide strong geometric priors. Some approaches directly predict point clouds [83, 206, 312], while others rely on implicit geometry representations. The latter infer 3D structure at test time by querying the representation with a spatial point and a partial observation (e.g., an RGB or RGB-D scan). Prior work uses signed distance functions for such representations [155, 228] or voxel occupancy grids [24, 26, 118, 160, 207, 232, 353]. OctMAE [129] builds on

the idea of MAE [110] from the image synthesis domain, and applies it to next-token prediction natively in 3D. Although these methods yield promising results, their resolution is constrained by the cubic cost of their volumetric resolution, leading to a coarse grid and smoothed structures lacking fine details.

Another strategy decomposes the scene at the object level [1, 376]. SceneComplete [1] constructs a 3D scene by chaining together foundation models for object segmentation, occlusion inpainting, 3D shape retrieval, and pose estimation. Despite its modular design, our experiments (Section 7.5) suggest that this reliance on multiple components introduces brittleness and several single points of failure.

Recent work like TRELLIS [361] instead learns a 3D latent representation from text or image, which can be decoded into various formats such as meshes. However, as shown in Section 7.5, experiments suggest it struggles in real-world multi-object scenes. In contrast, RaySt3R adopts a view-based strategy that is specifically designed for robust 3D completion in cluttered environments.

View-based reasoning Diffusion models [114] and their extensions [115, 252, 255] have enabled unprecedented performance in generative tasks such as image synthesis, inpainting, and video prediction. Several works leverage off-the-shelf generative models for 3D generation tasks [93, 188, 318, 356, 381]. ViewCrafter [381] leverages these advances by iteratively completing a scene point cloud using a point-conditioned video diffusion model. More recently, Li et al. [162] predict layered depth maps for constructing object-level and scene-level 3D geometries. While this method yields promising results on single-object shape completion, it struggles with predicting accurate geometries in real-world scenes containing multiple objects. Unique3D [356] tries to strike a balance between fidelity and inference speed by predicting multi-view images, generating corresponding normal maps and a textured mesh within 30 seconds.

While these models often yield visually appealing results, they lack geometric consistency, especially for cluttered real-world environments. The inference time of large diffusion models may also hinder deployment in robotics or XR settings. In contrast, RaySt3R predicts geometrically accurate depth maps from novel views, for fast and accurate 3D shape completion in cluttered real-world scenes.

7.3 Problem statement

Given a single RGB-D image, $I^{\text{input}} \in \mathbb{R}^{H \times W \times 3}$, $D^{\text{input}} \in \mathbb{R}^{H \times W}$, foreground mask $M \in \{0,1\}^{H \times W}$, and a camera with known intrinsics, $K^{\text{input}} \in \mathbb{R}^{3 \times 3}$, the goal is to predict the full 3D surface geometry of all masked foreground objects.

We frame the prediction goal as a set of points $Q \in \mathbb{R}^{N \times 3}$ that is both *accurate* and *complete* w.r.t. the ground-truth points $Q^{\text{gt}} \in \mathbb{R}^{S \times 3}$, sampled on the surfaces (e.g., meshes) of all objects in the scene.

We measure accuracy as the shortest distance from a predicted point to the nearest

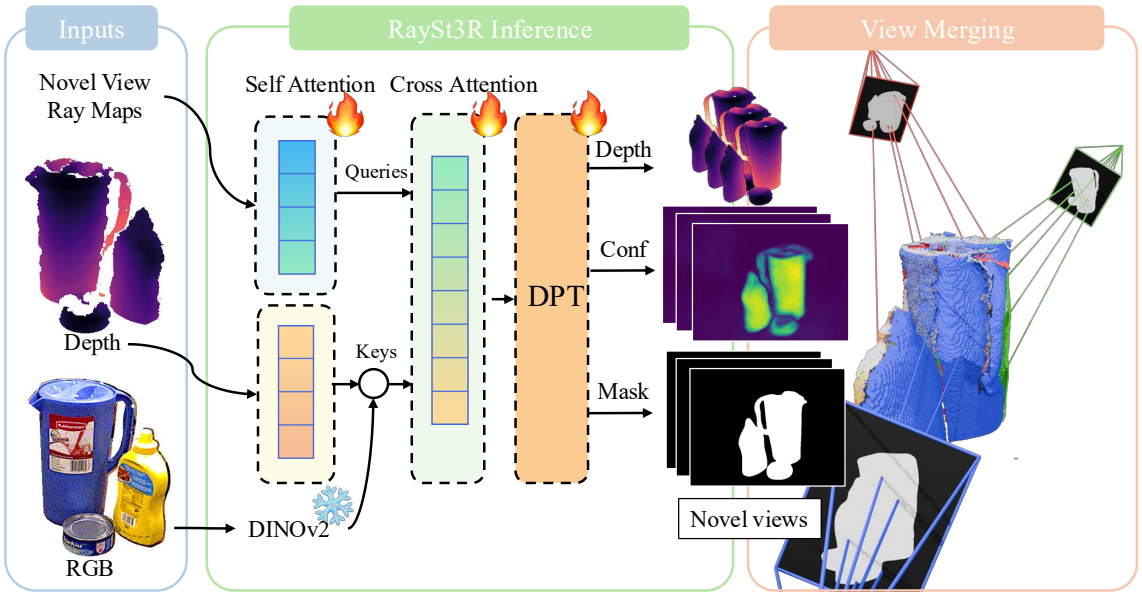


Figure 7.2: The architecture of RaySt3R. RaySt3R takes a single RGB-D image and foreground mask as input, and predicts depth maps, object masks, and per-pixel confidence scores for novel views. First, we apply the foreground mask to the RGB and point map input. Next, we use self-attention layers for the point map and ray map inputs, and feed the RGB image into the frozen DINOv2 [225] encoder. We feed all features into cross-attention layers followed by two separate DPT heads [244] for depth and mask predictions. Finally, we provide a confidence- and occlusion-aware multi-view merging formulation.

ground-truth point, averaged over all predicted points. We measure completeness as the shortest distance from a ground-truth point to the nearest predicted point, averaged over all ground truth points.

7.4 Methods

An overview of our approach is illustrated in Figure 7.2. We propose to train a transformer that, given the partial capture from a single RGB-D image and foreground mask, predicts depth maps and per-pixel confidence scores, and foreground masks for novel views. We first present the model architecture (Section 7.4.1), then the training objectives (Section 7.4.2). We then describe the procedure for querying novel views (Section 7.4.3) and conclude with the prediction merging strategy (Section 7.4.4).

7.4.1 Network architecture

The RaySt3R network architecture is inspired by DUSt3R [335] and successors [325, 331, 341]. Here, we leverage a ViT with point map, ray map, and depth map representations for 3D object completion.

The inputs to RaySt3R (Figure 7.2) are a foreground-masked RGB-D image and a novel query view. First, we unproject the input depth map D^{input} to a point map $X^{\text{input}} \in \mathbb{R}^{H \times W \times 3}$ using the given input image intrinsics K^{input}, thus

$$X^{\text{input}}_{i,j} = (K^{\text{input}})^{-1}[iD^{\text{input}}_{i,j},\ jD^{\text{input}}_{i,j},\ D^{\text{input}}_{i,j}]^{\mathsf{T}}.$$

We convert the query view into a ray map $R \in \mathbb{R}^{H \times W \times 2}$, with $R_{i,j} = [(i - c_x)/f_x,\ (j - c_y)/f_y]^{\mathsf{T}}$, where c_x, c_y are the image-center coordinates and f_x, f_y are the focal lengths of the novel target view.

Because we would like to pass information between the input and query views via cross attention, we transform the input point map X_{input} into the target camera coordinate frame to ease information sharing. That is, $X^{\text{context}} = P^{\text{query}}_{\text{input}} h(X^{\text{input}})$, where $P^{\text{query}}_{\text{input}} \in \mathbb{R}^{3 \times 4}$ transforms from the input to query camera coordinate frame, and $h : (x, y, z) \mapsto (x, y, z, 1)$.

We compute features for the point map and ray query with L layers of self-attention (SA).

$$F^{\text{point_map}} = \text{SA}(X^{\text{context}}), \quad F^{\text{ray}} = \text{SA}(R) \tag{7.1}$$

We mask out the background in X^{context} and replace it with a single learned background token. We process the RGB image by first masking out the background, and subsequently passing it through a frozen DINOv2 [225] encoder. Recent work has shown that a combination of features from different layers of a pre-trained ViT is useful for downstream tasks [79], hence we concatenate the features from intermediate layers of the DINOv2 encoder, and use a linear layer to project them to F^{DINO}.

For the cross-attention (CA) layers, we construct the keys of the first layer by concatenating $F^{\text{point_map}}$ and F^{DINO}. The queries are the ray features F^{ray}.

$$G = \text{CA}(F^{\text{ray}}, \text{concat}(F^{\text{point_map}}, F^{\text{DINO}})) \tag{7.2}$$

Finally, we use a DPT head [244] to predict depth maps and their confidence scores. A separate DPT head predicts the object mask.

7.4.2 Training objectives

We train RaySt3R to predict confidence-aware depth maps and object masks.

Depth loss Inspired by DUSt3R [335], we define the confidence-aware depth loss:

$$\mathcal{L}_{\text{depth}} = \sum_{i\in[0,W-1]} \sum_{j\in[0,H-1]} M^{\text{gt}}_{i,j} \left(C_{i,j} \left\| d_{i,j} - d^{\text{gt}}_{i,j} \right\|_2 - \alpha \log C_{i,j} \right). \tag{7.3}$$

Here, $C_{i,j}$ is the confidence score of each pixel in the predicted depth map, α is a hyperparameter, $d_{i,j}$ is the predicted depth, $d^{\text{gt}}_{i,j}$ is the ground-truth depth, and $M^{\text{gt}}_{i,j}$ is the ground-truth mask. The confidence scores are enforced to be strictly positive by setting $C_{i,j} \leftarrow 1 + \exp(C_{i,j})$. This enables a confidence estimate without explicit supervision.

We also predict binary object masks from novel viewpoints, and use a binary cross-entropy loss to supervise them during training.

$$\mathcal{L}_{\text{mask}} = \sum_{i\in[0,W-1]} \sum_{j\in[0,H-1]} \left(-m^{\text{gt}}_{i,j} \log(m_{i,j}) - (1 - m^{\text{gt}}_{i,j}) \log(1 - m_{i,j}) \right) \tag{7.4}$$

Here, $m_{i,j}$ is the predicted object mask after a sigmoid operation, and $m^{\text{gt}}_{i,j}$ is the ground-truth object mask. Finally, we combine the depth and mask losses with a sum weight λ_{mask}.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{depth}} + \lambda_{\text{mask}} \mathcal{L}_{\text{mask}} \tag{7.5}$$

7.4.3 View sampling

To construct a set of query views to sample, we fit a tight bounding box to the input view point map and sample points on a sphere with radius $\lambda_{bb} r_{bb}$ around the box's center. Here, λ_{bb} is a tunable parameter, and r_{bb} is half the length of the bounding-box diagonal. We found degenerate cases where the bounding box is too small, thus we clip the radius to be at least $\lambda_{\text{cam}} r_{\text{cam}}$, where r_{cam} is the distance from the camera to the center of the bounding box and λ_{cam} is a tunable hyperparameter. We sample points evenly on a cylindrical equal-area projection of the sphere to improve the coverage of the scene. We don't include the input point map in our predictions as it likely contains noise and artifacts. Instead, we query RaySt3R with the input view and include it in our predictions.

7.4.4 Merging predictions

After predicting depth maps and object masks for all novel views, we merge them to produce a complete 3D shape. We merge the depth maps by accounting for occlusions, RaySt3R's predicted masks, and the confidence scores.

Occlusion handling: First, we filter the points in each novel view to only parts of the scene that were not visible in the input image (i.e., those points occluded by the input view's foreground mask M^{input} and depth map D^{input}). Each point $q_{n,i,j}$ is defined as the point predicted by the n-th novel view at pixel (i, j). Its projection in the input view is given by

$p_{n,i,j} = K_{\text{input}} P_n^{\text{input}} h(q_{n,i,j})$, where P_n^{input} transforms points from the n-th novel view to the input view. We define each entry of the mask as:

$$m_{n,i,j}^{\text{occ}} = \begin{cases} 1 & \text{if } (p_{n,i,j})_z > D_{i,j}^{\text{input}} \text{ and } M_{i,j}^{\text{input}} = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{7.6}$$

RaySt3R predicted masks: Even with the occlusion constraint, the object mask from a novel view is largely unknown. For example, any observed surface could be a thin plate or a rich 3D object. We use the predicted mask from RaySt3R to filter the points, by thresholding the predicted mask $m_{i,j}^{\text{RaySt3R}} \in [0, 1]$ at 0.5.

Confidence scores: RaySt3R's architecture enables unsupervised confidence scores for each pixel in the predicted depth maps. Confidence scores are typically used to reduce edge-bleeding in dense ViT predictions [325, 335], or to exclude out-of-distribution objects such as specularities. With the same objective, we threshold $c_{i,j}^{\text{RaySt3R}}$ at τ for all experiments; more analysis is provided in Section 7.5.8.

Final mask: The final valid mask for a given novel view is obtained by setting

$$m_{n,i,j} = m_{n,i,j}^{\text{occ}} \cdot \mathbf{1}\left[m_{n,i,j}^{\text{RaySt3R}} > 0.5\right] \cdot \mathbf{1}\left[c_{n,i,j}^{\text{RaySt3R}} > \tau\right] \tag{7.7}$$

We obtain our final 3D reconstruction by aggregating valid points across all novel views.

7.5 Results

7.5.1 Training dataset

RaySt3R's training procedure requires a large number of camera pairs to scale zero-shot to the real world. It requires an RGB image for the input view and depth maps, intrinsics, and extrinsics for both cameras. We leverage existing synthetic datasets from FoundationPose [348] and OctMAE [129]. OctMAE [129] places GSO [65] and Objaverse [54] objects in synthetic scenes, and provides a single rendered image and depth map for each scene. FoundationPose has separate GSO and Objaverse splits; we only use the GSO split. For both datasets, we use the Objaverse and GSO meshes to render depth maps from novel views. Our dataset spans 1.1 million scenes with 12 million views in total.

7.5.2 Evaluation datasets

We evaluate RaySt3R on synthetic and real-world datasets. Following OctMAE [129], we evaluate on subsets of evaluation splits of the YCB-Video [362] (900 frames), HOPE [309] (50 frames), and HomebrewedDB [142] (1,000 frames) datasets. They are real-world 6D

pose estimation datasets with noisy depth maps and imperfect masks, including common objects such as boxes and cylinders, as well as items of complex geometries such as metal parts. For results on synthetic data, we evaluate on evaluation split of the OctMAE [129] (1,000 frames) dataset test split. We notice edge artifacts in the masks introduced due to data compression in the original work [129].

7.5.3 Data augmentation

Synthetic training data lacks noise and other artifacts as present in the real world. We therefore apply data augmentation during training to better bridge the sim-to-real gap. Inspired by [51, 345, 348], we apply a set of augmentations to the input views at training time. For depth maps, we randomly apply Gaussian noise, add holes, and shift the pixel coordinates [13, 51]. For the RGB image, we randomly vary brightness and contrast, and apply per-channel salt-and-pepper noise and Gaussian noise.

7.5.4 Implementation details

We train RaySt3R on $8\times$ 80-GB A100 GPUs for 18 epochs, totaling approximately 20 million scene iterations. We set the batch size to 10 per GPU, and a learning rate of 1.5×10^{-4} with a half-cosine learning-rate schedule, starting with one warm-up epoch and using an AdamW optimizer [196]. We use a ViT-B model with patch size 16, embedding dimension 768, 12 heads, 12 cross-attention layers, but 4 self-attention layers to save on compute. We select the ViT-L with registers for DINOv2 [225].

We set $\lambda_{bb} = 1.3$ and $\lambda_{\text{cam}} = 0.7$ for all real-world datasets, and $\lambda_{bb} = 2.5$ and $\lambda_{\text{cam}} = 1.2$ for the OctMAE dataset. The parameters are chosen to be larger for the OctMAE dataset, as the input view is typically placed very close to the objects with severe occlusions. We set the confidence threshold $\tau = 5$ for all experiments, and sample 22 views in total. During training we set the confidence parameter $\alpha = 0.2$, $\lambda_{\text{mask}} = 0.1$. Inference takes less than 1.2 seconds on a single RTX 4090 GPU, and can be further reduced by querying fewer views.

7.5.5 Baselines

We compare RaySt3R against the state-of-the-art in 3D shape completion. OctMAE introduced a novel 3D MAE algorithm, and also trained prior shape completion models on their novel dataset. We compare against OctMAE, and the prior works they trained, i.e., VoxFormer [167], ShapeFormer [370], MCC [353], ConvONet [232], POCO [24], AICNet [160], Minkowski [47], and OCNN [329].

We also compare against SceneComplete [1], which uses a combination of foundation models to produce complete geometry. The authors leverage a VLM and Grounded-SAM [249] to produce object-level masks, image inpainting to fill in occluded regions, an image-to-3D model to produce a 3D mesh, and finally FoundationPose for 3D alignment.

We also benchmark against Unique3D [356] and TRELLIS [361], which are recent image to 3D models. Finally, we compare against 'Layered Ray Intersections' (LaRI) [162], which introduced the concept of layered point maps to predict multiple points on each camera ray. Unique3D, Trellis, and LaRI predict points in canonical coordinates; we align the predictions with the ground truth 3D using first a brute-force search for a similarity transform, followed by ICP [277]. Note that we do not perform such a registration for RaySt3R, but provide this to baselines to give them the benefit of the doubt. While LaRI [162] and Unique3D [356] do not require foreground masks for reconstructing objects, we observe that TRELLIS [361] tends to reconstruct the entire scene. Therefore, we compare TRELLIS with a masked image input and a raw image input. We also attempted to evaluate ViewCrafter [381] on this task, but the images produced by the video diffusion model were of too poor quality to perform the evaluation. We provide more details in the supplementary material of the original paper [73].

7.5.6 Quantitative results

Following prior work [129], we evaluate the zero-shot generalization performance of all methods using Chamfer distance (CD) and F1-Score@10mm (F1). Detailed formulations of the metrics are provided in the original paper's supplementary material. We present the quantitative results of this evaluation in Table 7.1. RaySt3R consistently outperforms all baselines across both synthetic and real-world scenes. The strongest baseline, OctMAE [129], performs competitively; however, RaySt3R surpasses it across all metrics, by $20\,\%$ to $44\,\%$ in CD. SceneComplete [1] proves to be a fragile pipeline, therefore not yielding competitive results. We were unable to produce SceneComplete results on HOPE, as the pipeline requires an intractable amount of VRAM for cluttered scenes. LaRI [162], Unique3D [356], and TRELLIS [361] show better performance than SceneComplete [1], but also do not produce competitive results. Feeding masked RGB images to TRELLIS [361] outperforms raw image inputs on all datasets except HomebrewedDB [142]. We show common failure modes in Section 7.5.7.

We also compute the standard deviation of the Chamfer distance across all real-world datasets for each method and observe that our model exhibits the lowest standard deviation ($1.74\,\mathrm{mm}$), followed by OctMAE [129] ($2.38\,\mathrm{mm}$) and Unique3D [356] ($8.11\,\mathrm{mm}$).

7.5.7 Qualitative results

Figure 7.3 shows qualitative results of RaySt3R on real-world datasets. The results suggest that RaySt3R is capable of generating high-quality 3D predictions, generating more consistent and complete results compared to the baselines. The most competitive baseline, OctMAE [129], produces viable but oversmoothed object shapes due to its low-resolution 3D MAE grid. Furthermore, TRELLIS [361], Unique3D [356] and LaRI [162] struggle with relative object placement and aspect ratios. They also fail for certain out-of-distribution objects, and occasionally fail to register to the ground truth point cloud. Interestingly,

Table 7.1: Quantitative evaluation of multi-object scene completion on synthetic and real-world datasets. We evaluate on the test split of OctMAE [129], and the BoP benchmarks YCB-Video [362], HOPE [309], and HomebrewedDB [142]. We report Chamfer distance (CD) [mm] and F1-Score@10mm (F1). The first section contains numbers copied from OctMAE [129], the second section contains recent works we evaluated. For alignment of LaRI [162], Unique3D [356], and TRELLIS [361] with the ground truth mesh, we apply brute force search followed by ICP [277]. We evaluate TRELLIS [361] with masked and unmasked RGB inputs. SceneComplete [1] runs out of VRAM on HOPE [309]. The results suggest RaySt3R outperforms all baselines.

	Synthetic		Real					
	OctMAE [129]		YCB-Video [362]		HB [142]		HOPE [309]	
Method	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑	CD↓	F1↑
VoxFormer [167]	44.54	0.382	30.32	0.438	34.84	0.366	47.75	0.323
ShapeFormer [370]	39.50	0.401	38.21	0.385	40.93	0.328	39.54	0.306
MCC [353]	43.37	0.459	35.85	0.289	19.59	0.371	17.53	0.357
ConvONet [232]	23.68	0.541	32.87	0.458	26.71	0.504	20.95	0.581
POCO [24]	21.11	0.634	15.45	0.587	13.17	0.624	13.20	0.602
AICNet [160]	15.64	0.573	12.26	0.545	11.87	0.557	11.40	0.564
Minkowski [47]	11.47	0.746	8.04	0.761	8.81	0.728	8.56	0.734
OCNN [329]	9.05	0.782	7.10	0.778	7.02	0.792	8.05	0.742
OctMAE [129]	6.48	0.839	6.40	0.800	6.14	0.819	6.97	0.803
LaRI [162]	39.22	0.283	11.41	0.658	22.23	0.414	18.64	0.528
Unique3D [356]	44.62	0.244	17.56	0.468	25.41	0.329	26.37	0.322
TRELLIS (w/ mask) [361]	61.74	0.227	22.94	0.443	35.29	0.354	20.25	0.443
TRELLIS (w/o mask) [361]	65.74	0.225	31.74	0.338	30.71	0.348	22.05	0.416
SceneComplete [1]	81.57	0.289	96.63	0.359	85.81	0.416	N/A	N/A
RaySt3R (ours)	**5.21**	**0.893**	**3.56**	**0.930**	**4.75**	**0.889**	**3.92**	**0.926**

TRELLIS [361] may predict table surfaces even for masked input images with no visible table. SceneComplete [1] proves to be brittle to single-point failures such as missing object masks.

7.5.8 Ablation studies

Training ablations: Table 7.2 summarizes our training ablation results, with all models trained under the same setup for roughly 20 million scene iterations. Training a ViT-S model (384-dim, 6 heads) leads to a performance drop. Disabling data augmentation further degrades zero-shot generalization to real-world data, highlighting its importance. We also compare training on 100k uniformly sampled scenes versus the full 226k GSO set. The smaller but more diverse 100k set performs better, emphasizing the value of data diversity. Finally, removing DINOv2 [225] inputs causes an additional decline, underscoring the benefit of pre-trained features.

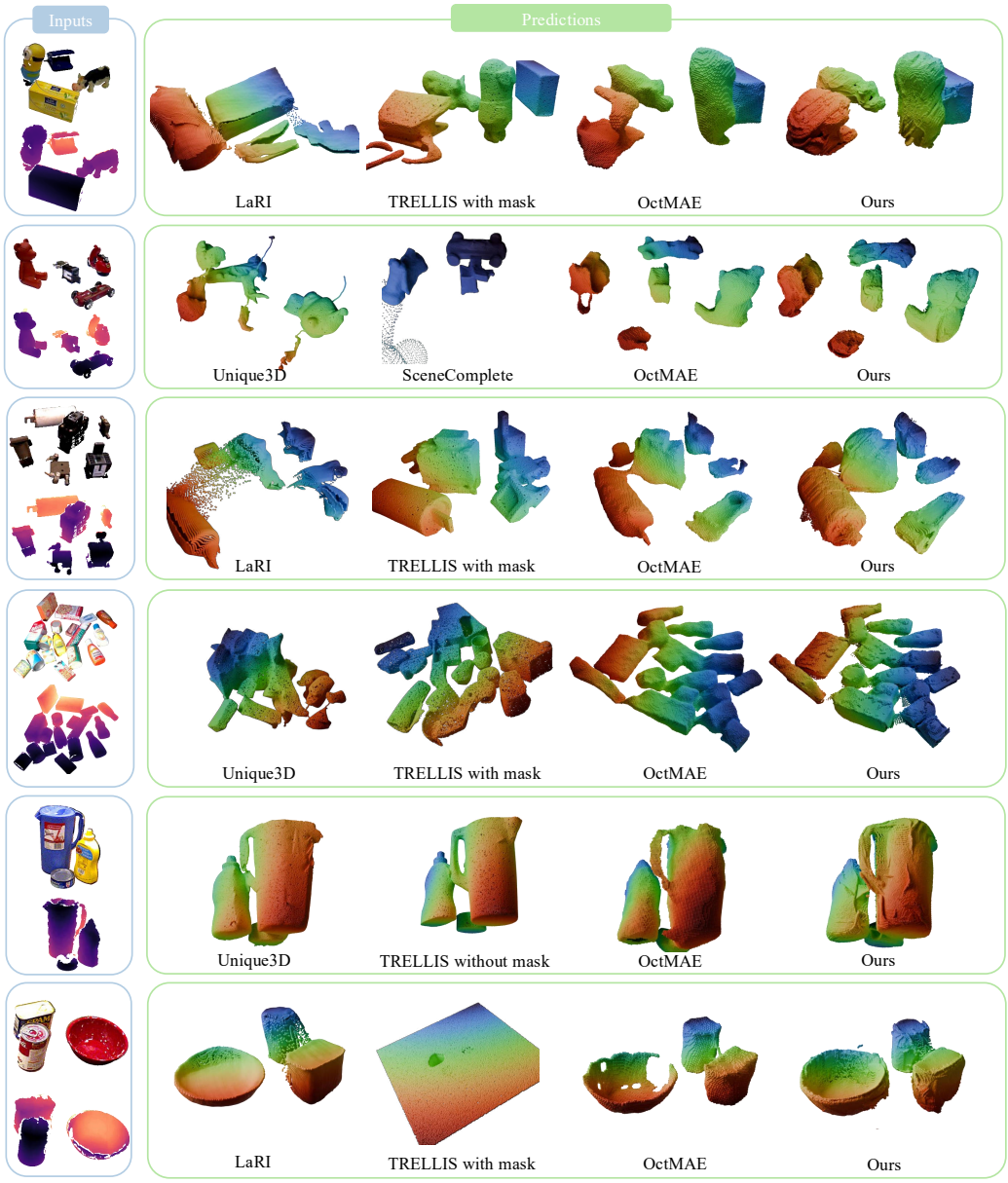


Figure 7.3: Qualitative results of RaySt3R in real-world multi-object scenes. For each scene, we select a subset of methods to preserve visual clarity, but we share all method predictions in the supplementary material of the original paper [73]. The results suggest RaySt3R produces the most geometrically accurate shapes compared to the baselines. The most competitive baseline, OctMAE [129], tends to predict softer shapes as a result of its coarse 3D MAE grid. We observe TRELLIS [361], Unique3D [356], and LaRI [162] struggle with relative object placing, aspect ratios, and out-of-distribution objects, occasionally leading to incorrect registration to the ground truth points. Finally, SceneComplete [1] proves to be brittle to single points of failure such as missing object masks.

Table 7.2: Ablation study on RaySt3R training on the YCB-Video dataset [362]. The results suggest data scale, data diversity, DINOv2 [225] features, data augmentation, and model size affect RaySt3R's performance.

Method name	CD ↓	F1 ↑
RaySt3R (proposed)	**3.56**	**0.930**
ViT-S	3.70	0.920
No data augmentation	3.89	0.916
Train on 100k scenes	4.30	0.894
w/o DINOv2 [225]	4.81	0.877
Train on 226k GSO scene	5.34	0.864

Table 7.3: Ablation study on RaySt3R view merging on the OctMAE test dataset [129]. We ablate querying the input view (Section 7.4.3), occlusion masking with the input mask, and finally RaySt3R's predicted masks. The results suggest all steps contribute to performance, especially the predicted masks.

Query Input	Occ. Mask	Pred. Mask	CD↓	F1↑
✓	✓	✓	**5.21**	**0.893**
✗	✓	✓	7.55	0.836
✓	✗	✓	7.69	0.855
✓	✓	✗	10.12	0.825
✗	✗	✗	73.17	0.641

View merging ablations: Table 7.3 shows the ablations on our view merging formulation. We ablate querying the input view (Section 7.4.3), using the input mask to detect occluded regions, and finally the use of RaySt3R's predicted mask. The results suggest that each component of our formulation contributes to shape completion performance, especially the predicted masks.

Confidence RaySt3R predicts a per-pixel confidence value, which can be used to filter the predictions. To understand the impact of the confidence threshold, we report Chamfer distance, Completeness, and Accuracy for a range of thresholds, as depicted in Figure 7.4. The results suggest that confidence is a good proxy for error and that confidence can be effectively used to trade off accuracy and completeness. Depending on the application, the threshold can therefore be tuned accordingly, as some applications may be less tolerant to outliers requiring high accuracy, while others may require more complete predictions. For all prior experiments, we set confidence threshold τ to 5 for a balance between accuracy and completeness. Figure 7.5 shows a visualization of the impact of changing the confidence threshold on the predicted 3D points.

7.6 Conclusion and future work

We present RaySt3R, a novel approach to 3D shape completion from a single RGB-D image and foreground mask. RaySt3R learns to predict depth maps, object masks, and per-pixel confidence scores for novel viewpoints, which are fused to reconstruct complete 3D shapes. We benchmark RaySt3R on real-world and synthetic datasets, and compare it to the state-of-the-art in volumetric and view-based methods. The results suggest that RaySt3R

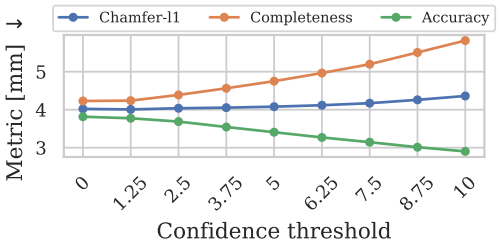


Figure 7.4: Confidence threshold vs error metrics averaged over all real-world datasets. The results suggest increasing the confidence threshold improves accuracy and degrades completeness.



Figure 7.5: The impact of the confidence threshold on the predicted 3D points. This experiment suggests increasing the confidence threshold can aid in reducing the edge bleeding issue.

is capable of generating high-quality 3D predictions, outperforming the baselines across the board.

RaySt3R's results could be further improved by training on real-world data and scaling up compute. The current synthetic dataset might be too limited to capture increasingly complex objects. Future work may also explore scaling up compute by exploring other architectures like diffusion transformers.

Chapter 8

Modality Forcing for Scalable Spatial Generation

ORIGINALLY APPEARED AS

Bardienus Pieter Duisterhof, Deva Ramanan, Jeffrey Ichnowski, Justin Johnson, and Keunhong Park. *Modality Forcing for Scalable Spatial Generation [74].* arXiv:2606.13676, 2026.

Chapter in Context

RaySt3R's posed depth maps can be used for object completion, but this kind of data is not cheaply available at scale. To scale at a lower cost, can we use image-only pre-training with text-to-image (T2I)? In Modality Forcing, we study this question by post-training T2I models to produce monocular depth. We find that depth performance scales with T2I performance, and contribute a state-of-the-art monocular depth predictor. This addresses SOLF's *obtainability* by learning geometric priors from abundant images with limited depth supervision.

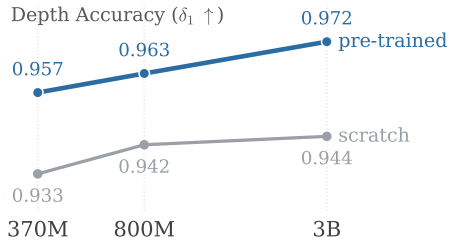
Text-to-image (T2I) models contain rich spatial priors. Synthesizing photorealistic, cluttered scenes requires an understanding of geometry, including perspective and relative scale. Prior works adapt T2I models to leverage this prior for depth prediction, but they require dense depth data and involve complex recipes. We propose Modality Forcing, a simple, scalable post-training recipe for joint image-depth generation using a single DiT trained on sparse depth data. Modality Forcing enables conditional and joint generation of image and depth in any permutation by assigning separate noise levels per modality. Per-modality decoders let us train on sparse, real-world depth and achieve strong, generalizable depth prediction. We further show that Modality Forcing inherits the scalability of T2I pre-training: by training a set of T2I models from scratch (370M to 3.3B parameters), we find that larger models trained on more image data produce more accurate depth. Our strongest model is competitive with state-of-the-art monocular depth estimators and reduces AbsRel by 57% relative to existing joint image-depth generative models. These results provide strong evidence that image generation is a scalable pre-training objective for spatial perception.



(a) Joint Generation: Text ⇒ Image + Depth



(b) Conditional Generation: Image ⇌ Depth



(c) Scales with T2I model size

Figure 8.1: We present Modality Forcing, a post-training recipe to extract spatial priors from text-to-image (T2I) models. A single DiT models the joint distribution over images and depth, enabling joint and conditional generation in arbitrary combinations. We demonstrate depth predictions improve with increasingly capable T2I pre-training, suggesting T2I is a scalable objective for spatial generation.

8.1 Introduction

Over the past several years, simple architectures and formulations fed with plentiful data have outperformed complex human-designed pipelines in many tasks. Examples include DUSt3R [335] for 3D reconstruction, SAM [150] for segmentation, and SAM-3D [295] for mesh generation. A central lesson is that general methods capable of absorbing more data are more effective than complex heuristics—often called the *bitter lesson* [289].

Models trained on task-specific data have led to considerable advances; however, 3D data is scarce, which makes it difficult to scale to the quantities required for large models. One approach to ease this is to combine multiple synergetic tasks in a unified multi-task formulation. Language modeling is the clearest example, where a single generalist model trained on next-token prediction [29] exceeds specialist systems across translation, question answering, and coding.

Text-to-image (T2I) models contain similarly rich representations, but adapting them to new spatial tasks is difficult. First, they denoise latent tokens in an image VAE which is not designed for arbitrary spatial modalities. Depth as images limits training to dense (usually



Figure 8.2: Modality Forcing generates rich RGB-Depth from text prompts. Unprojecting the points to 3D, Modality Forcing generates faithful and sharp geometry. The same checkpoint enables monocular depth estimation, and depth-to-image generation competitive with the best specialist models.

synthetic) samples and precludes encoding other spatial modalities such as meshes or point clouds. Second, post-training T2I models for spatial tasks may destroy their pre-trained backbones and limit the pre-training benefit. Post-training generative models to produce spatial modalities requires a simple and scalable recipe.

We introduce Modality Forcing, a post-training recipe that enables a pre-trained T2I model to predict depth. Modality Forcing combines latent RGB tokens and pixel-space depth tokens into a single DiT to model the joint image-depth distribution. Pixel-space depth diffusion enables learning from sparse real-world depth annotations and achieves state-of-the-art results. Once trained, Modality Forcing allows setting RGB or depth as a target or conditioning, enabling joint generation, or conditional image-to-depth (I2D) or depth-to-image (D2I) generation.

Prior work has also modeled the joint RGB-depth distribution, but relied on learned adapters and was limited to dense depth data [30, 390]. We show that post-training a single DiT on the joint distribution unlocks training on real-world data and improves depth predictions by 57% on average. Concurrent work explored native multimodal pre-training [371] and instruction fine-tuning [89], representing depth as images. These models show encouraging results across visual tasks but share the same challenges with data, and leave it unclear whether depth quality scales with T2I capability.

We contribute a controlled scaling study to investigate the scalability of Modality Forcing with T2I model capability. We train a family of DiTs from scratch, spanning from 370M

to 3.3B parameters, and apply the same Modality Forcing recipe to each. We find that increasingly capable T2I models, larger models trained on more images, produce more accurate depth predictions. This suggests T2I is a scalable pre-training objective for spatial generation.

Contributions.

- **Modality Forcing**, a simple post-training recipe that unifies three spatial tasks—monocular depth estimation (I2D), depth-to-image (D2I), and joint image-depth generation with one set of weights.
- A **controlled scaling study** from 370M to 3.3B parameters and from zero to 1.92B images, showing that depth predictions scale with T2I model and training size.
- **State-of-the-art results**: applied to FLUX.2-klein-9B, Modality Forcing competes with the best depth estimators and outperforms joint image-depth baselines by a margin of 57%.

8.2 Related Work

Text-to-image. Text-to-image (T2I) models have advanced rapidly over the past few years [82, 154, 252, 296]. The Stable Diffusion family of models began with latent diffusion models [252], which perform denoising in the latent space of a pre-trained VAE rather than in pixel space; this became the dominant template for efficient image synthesis. A major architectural shift came with Stable Diffusion 3 [82], which replaced the U-Net with a Diffusion Transformer (DiT) [230] trained under a rectified-flow objective. The current state of the art for open-source models is largely set by FLUX [154] and Z-Image [296] — both flow transformers operating in the latent space of an image VAE and trained on large volumes of curated data. Whether this growing T2I capability reflects a correspondingly stronger underlying *spatial* prior remains untested; Modality Forcing answers this by post-training a controlled family of T2I checkpoints into joint image-depth generators under one recipe.

Monocular depth estimation. Monocular depth estimation has recently been transformed by foundation models trained on large corpora [333, 334, 373, 374]. Depth Anything V2 [374] scales by distilling a large teacher trained on synthetic data through pseudo-labeled real images, yielding strong zero-shot depth. Other works repurpose T2I models as depth priors: Marigold [143] fine-tunes Stable Diffusion on synthetic RGB-D pairs. The DiT architecture itself has also been adapted for depth: Pixel-Perfect Depth [366] shows that diffusing depth in pixel space avoids edge artifacts. Most recently, MoGe [333] reframes the problem as direct prediction of an affine-invariant 3D point map from which depth and intrinsics can be derived, and MoGe-2 [334] extends this to metric scale. Together, these lines of work establish that foundation-style learning on large 3D datasets yields high-quality depth and that T2I models carry exploitable spatial priors. Despite the rapid progress of T2I

models, the extent to which their spatial priors transfer to depth has not been systematically studied.

Joint Image-Depth Generation. A growing body of work tackles jointly generating images and depth, motivated by the observation that a single joint distribution over RGB and depth unifies conditional and joint generation. LDM3D [280] retrains a Stable Diffusion VAE and U-Net on six-channel RGB-D inputs. JointNet [390] duplicates the pre-trained U-Net into a parallel depth branch tightly coupled to a frozen RGB branch, supporting bidirectional prediction via channel-wise inpainting. UniCon [165] generalizes this with LoRA and disentangled per-branch noise schedules, recovering depth-conditioned generation, depth estimation, and joint sampling in a single model. JointDiT [30] ports the paradigm to diffusion transformers atop FLUX.1, introducing unbalanced per-modality timestep sampling. Orchid [152] instead trains a joint VAE over color, depth, and normals, paired with a single latent diffusion model. All of these methods rely on dense RGB-D supervision, typically from synthetic data or curated captured datasets. In contrast, Modality Forcing is a post-training recipe that scales with the base T2I model and removes the requirement for dense supervision.

8.3 Method

Modality Forcing (Figure 8.3) post-trains a pre-trained text-to-image DiT to model the joint distribution over RGB and depth. The core idea is to give each modality its own noise level, so that a single set of weights supports joint generation and either conditional direction. Tokenizing depth in pixel space further lets the model learn from sparse, real-world depth supervision.

8.3.1 Problem Definition

We unify three generation tasks under a single model: joint image and depth generation, depth to image, and image to depth. Let $\mathbf{c}$ denote a text prompt, $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ an RGB image, and $\mathbf{d} \in \mathbb{R}^{H \times W}$ a depth map. We learn a joint generator

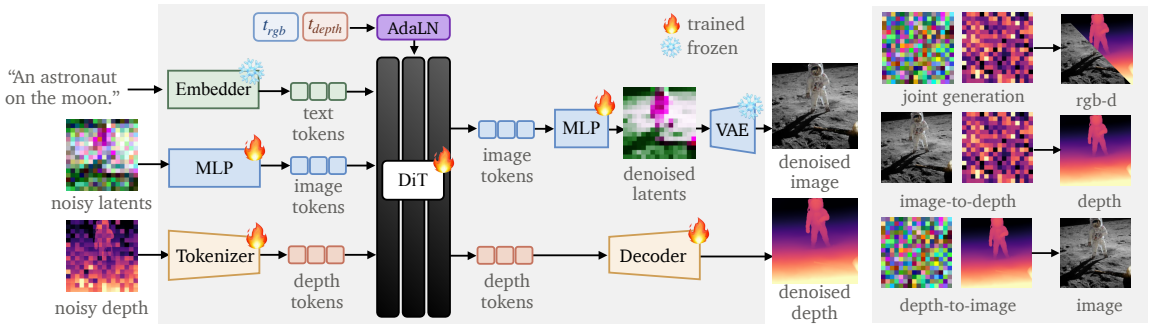
$$p_\theta(\mathbf{x}, \mathbf{d} \mid \mathbf{c}) \tag{8.1}$$

from a set of $(\mathbf{c}, \mathbf{x}, \mathbf{d})$ triples. A single model p_θ then supports three tasks during inference:

$$(\mathbf{x}, \mathbf{d}) \sim p_\theta(\mathbf{x}, \mathbf{d} \mid \mathbf{c}) \qquad \text{Joint RGB-D generation} \tag{8.2}$$

$$\mathbf{d} \sim p_\theta(\mathbf{d} \mid \mathbf{x}, \mathbf{c}) \qquad \text{Image-to-depth (I2D)} \tag{8.3}$$

$$\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{d}, \mathbf{c}) \qquad \text{Depth-to-image (D2I)} \tag{8.4}$$



(a) Training (b) Inference

Figure 8.3: Modality Forcing is a recipe to post-train image-generation models for depth prediction. We encode RGB using a pre-trained VAE and depth using a per-pixel tokenizer. During training (a), each modality is independently noised and trained with separate losses. At inference (b), each modality's noise level can be controlled to yield image→depth, depth→image, or joint generation.

Preliminaries. Diffusion models are an expressive class of generative models for high-quality generation. We discuss flow-matching with v-prediction [184] and x-prediction [164]. Let $\mathbf{x}_0 \sim q(\mathbf{x})$ denote a clean data distribution and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ Gaussian noise. We interpolate linearly between the two with a scalar $t \in [0, 1]$ to create training samples,

$$\mathbf{x}_t = (1 - t)\,\mathbf{x}_0 + t\,\boldsymbol{\epsilon}, \tag{8.5}$$

so that $\mathbf{x}_0$ is clean and $\mathbf{x}_1$ is pure noise. The constant velocity along this path is $\mathbf{v} = \mathbf{x}_0 - \boldsymbol{\epsilon}$. With *v-prediction* we parameterize the output as the velocity and regress a network $g_\theta(\mathbf{x}_t, t)$ directly:

$$\mathcal{L}_{\text{v-pred}} = \mathbb{E}\big[\|g_\theta(\mathbf{x}_t, t) - \mathbf{v}\|_2^2\big]. \tag{8.6}$$

Recent work has shown that v-prediction can struggle in high-dimensional spaces [164], instead suggesting to parameterize the output as the clean sample, $\hat{\mathbf{x}}_0 = g_\theta(\mathbf{x}_t, t)$. The implied velocity $\hat{\mathbf{v}} = (\hat{\mathbf{x}}_0 - \mathbf{x}_t)/t$ is regressed to $\mathbf{v}$:

$$\mathcal{L}_{\text{x-pred}} = \mathbb{E}\left[\left\|\big(g_\theta(\mathbf{x}_t, t) - \mathbf{x}_t\big)/t - \mathbf{v}\right\|_2^2\right]. \tag{8.7}$$

This is well-conditioned when the data lies on a low-dimensional manifold (such as natural images or depth maps). Both methods sample the learned vector field from $t = 1$ to $t = 0$ with an ODE solver.

8.3.2 Modality Forcing

We introduce Modality Forcing, a diffusion algorithm with per-modality noise levels. Assigning separate noise levels allows conditional and joint generation in any permutation: an image with no noise conditions, while a full noise image is a generation target. Several

methods use this idea, but differ in the axis along which the noise varies. Teacher Forcing [352] runs along sequence position, holding past tokens clean while denoising the next. Diffusion Forcing [36] varies noise level per token. Latent Forcing [8] denoises DINO [225] latents ahead of raw pixels. Modality Forcing extends this idea to the axis of modality: RGB and depth carry their own noise levels within a diffusion process, so that fixing a modality at $t = 0$ makes it conditioning. The schedule alone selects joint, image-to-depth, or depth-to-image generation.

We post-train existing T2I models to support all sampling schemes. During training, we sample per-modality noise levels in three different ways to support our three generation tasks. Joint RGB-D generation samples $t_{\text{rgb}}, t_{\text{depth}} \in [0, 1]$; I2D fixes $t_{\text{rgb}} = 0$ and samples $t_{\text{depth}} \in [0, 1]$; and D2I samples $t_{\text{rgb}} \in [0, 1]$ and fixes $t_{\text{depth}} = 0$.

Depth tokenizer. Real-world videos typically only contain sparse depth annotation, since depth is estimated, e.g., using multi-view stereo (MVS) pipelines. To allow for post-training on these sparse annotations, we denoise depth directly in pixel space rather than the latent VAE space. This is in contrast to prior work [30, 143, 165] which tokenizes depth through a pre-existing or new image VAE, with no obvious mechanism to accommodate partial depth supervision. We fill missing pixels with isotropic Gaussian noise to signal to the model that depth is not available at those locations, equivalent to how a fully missing depth map is encoded.

Per-modality timestep conditioning. Because each modality is denoised at its own noise level, every token must be modulated by its own timestep. We give RGB and depth separate timestep embedders: the RGB stream reuses the pre-trained embedder, while depth receives a freshly initialized one. Since joint and conditional generation couple the two noise levels, we additionally let each stream's modulation observe the other modality's timestep through a lightweight cross-stream mixing module—one small embedder per direction. We initialize both to zero, so the embedding begins as an exact identity with no cross-communication and learns the coupling over training.

Depth detokenizer. The DiT blocks are followed by a depth detokenizer with n layers of self-attention, followed by a final linear layer that maps the depth tokens back to pixel space. The extra depth blocks help create depth-specific capacity without disrupting the RGB stream. We normalize depth supervision by scaling it to have unit mean, followed by spatial contraction [14] to ensure $d \in [0, 2]$.

Initialization. We warm-start the depth pathway rather than training it from scratch. We initialize the depth stream by cloning the pre-trained image-stream weights; the remaining depth-specific modules—the pixel-space tokenizer, the depth timestep embedder, the detokenizer, and the cross-stream mixers—are initialized from scratch.

8.3.3 Self-Distillation

T2I models are trained on billions of images; post-training them on millions of less diverse images may erode the rich prior. To preserve it, we introduce a self-distillation loss that penalizes the student for drifting from the original T2I checkpoint (similar in spirit to 'Learning without Forgetting' [174]). At train time, we pass the current noisy RGB through the frozen T2I model and record its predicted velocity $\mathbf{v}_{\text{t2i}}$, then penalize the deviation of the student's RGB velocity $\mathbf{v}$:

$$\mathcal{L}_{\text{dist}} = (\lambda_{\text{hi}} t_{\text{depth}} + \lambda_{\text{lo}}(1 - t_{\text{depth}}))||\mathbf{v}_{\text{t2i}} - \mathbf{v}||_2^2, \tag{8.8}$$

where $||\mathbf{v}_{\text{t2i}} - \mathbf{v}||_2^2$ is the L2 loss between the predicted velocity and the T2I-predicted velocity. We set $\lambda_{\text{hi}} > \lambda_{\text{lo}}$ so the penalty is strongest at $t_{\text{depth}} = 1$: when depth is fully noised it carries no information. As depth is denoised toward $t_{\text{depth}} = 0$, it supplies context the T2I model never sees, so we relax the penalty rather than force agreement with a now-uninformed teacher.

8.4 Results

We evaluate Modality Forcing across joint and conditional RGB-Depth tasks. First, we train a suite of T2I models from scratch to study how depth generation scales with T2I model and training dataset size. Next, we apply Modality Forcing to FLUX.2-klein-9B, and benchmark it against the best specialist models. Our experiments show that depth quality scales reliably with T2I model size and data, and that Modality Forcing is competitive with the best depth models.

8.4.1 Scaling experiment.

We conduct a controlled scaling experiment to answer: *does depth performance scale with T2I capability?* We train three T2I models on a large and diverse set of web images at 256×256 resolution and post-train with Modality Forcing. We evaluate depth performance across model and training dataset sizes. There should be improvements across both axes if our hypothesis is true.

Each T2I model is a DiT trained with a flow-matching objective. For each training pair $(\mathbf{x}, \mathbf{c})$, we encode the image into a clean latent $\mathbf{z}_0$ with the frozen FLUX.2 VAE [154]. We sample a timestep $t \in [0, 1]$ and add noise to the latent along the linear interpolant $\mathbf{z}_t = (1 - t)\,\mathbf{z}_0 + t\,\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The network output $g_\theta(\mathbf{z}_t, t, \mathbf{c})$ is parameterized as a clean-latent prediction $\hat{\mathbf{z}}_0$, and we regress the implied velocity to the true velocity $\mathbf{v} = \mathbf{z}_0 - \epsilon$:

$$\mathcal{L}_{\text{T2I}} = \left\|(g_\theta(\mathbf{z}_t, t, \mathbf{c}) - \mathbf{z}_t)/t - \mathbf{v}\right\|_2^2. \tag{8.9}$$

The resulting checkpoints are the starting points for Modality Forcing post-training.

Table 8.1: **Training data.** We train on twelve real-world and simulated datasets totaling 17M frames across 58k scenes, spanning indoor scanning, outdoor driving, and synthetic rendering.

Dataset	# Scenes	# Frames
Argoverse 2	700	108,139
Aria Project Sim	5,022	2,424,158
ARKitScenes	5,041	744,200
Blended MVS	493	113,209
FoundationStereo	40,936	40,936
Hypersim	457	74,519
MegaDepth	113	19,447
ParallelDomain	1,520	239,840
ScanNet v2	1,200	94,751
TartanAir v2	1,122	8,563,146
Taskonomy	520	4,385,534
Waymo Open	796	159,000
Total	**57,920**	**16,966,879**

Table 8.2: Base T2I model sizes used for scaling experiment, ranging from 370M to 3.3B parameters.

Param Count	Token Size	FFN Dim	Depth
370M	1024	3,072	30
800M	1536	4,608	30
3.3B	3072	9,216	30

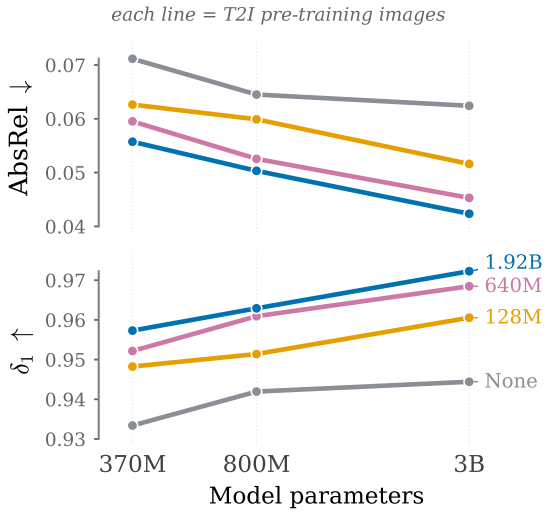


Figure 8.4: Scaling experiments. Depth accuracy (δ_1, ↑, bottom) and AbsRel (↓, top) by T2I model size. Each line represents a T2I pre-training dataset size (none, 128M, 640M, 1.92B). Training larger T2I models on more image data yields better depth performance.

We vary model size from 370M to 3.3B (Table 8.2), and T2I pre-training data size from zero to 1.92B samples. For depth, we use ≈17M training frames, a combination of real and synthetic images (Table 8.1), and evaluate depth across NYUv2 [273], DIODE [313], ETH3D [268] and ScanNet [50].

Figure 8.4 shows depth performance across model and pre-training data size. The results suggest that the performance of Modality Forcing scales with T2I model size and training data. Larger T2I models and larger training data result in improved depth quality. This suggests Modality Forcing is a scalable recipe for depth generation. The results show that T2I pre-training and not just model size enables better depth quality, presenting direct evidence of the spatial prior present in T2I models.

Implementation Details The T2I trunk is a flat-token DiT [230] with parallel attention and MLP branches per block, pre- and post-RMSNorm, grouped-query attention, and RoPE positional encoding [282]. Latents are produced by a frozen FLUX.2 VAE [154] and tokenized by a fresh patch embedding; text prompts are encoded by a frozen UMT5-XXL [48] into embeddings, and the trunk attends to these text tokens at every block. We use logit-normal sampling for RGB, and plateau logit-normal sampling for depth, both with a timestep shift [82] of $\mu = 1.1$. Additionally, we set $p_{\text{i2d}} = 0.2$ and $p_{\text{d2i}} = 0.2$, to focus the network on the case of a fully denoised image or depth map.

Captioning follows protocols similar to those in SD3 [82] and WAN [320]. Each model in the family has the same number of layers but a different token size (Table 8.2).

The metrics are averaged across NYUv2 [273], DIODE [313], ETH3D [268] and ScanNet [50] datasets and take 256×256 center crops. We use the robust affine-invariant alignment introduced in MoGe-2 [334].

8.4.2 FLUX-Based Modality Forcing

After showing Modality Forcing scales with T2I capability, we now evaluate its performance when paired with a SoA T2I model. We post-train FLUX.2-klein-9B with the Modality Forcing recipe and evaluate it on conditional and joint generation tasks. We post-train this model using the real-world and synthetic depth data shown in Table 8.1. The training recipe is largely unchanged from Section 8.4.1, only now we apply the self-distillation loss to retain more of the T2I prior (Section 8.3.3).

Image-to-depth generation We evaluate the monocular depth estimation capability by evaluating affine-invariant depth on NYUv2 [273], DIODE [313], ETH3D [268], ScanNet [50] and KITTI [99]. We evaluate the Modality Forcing recipe against state-of-the-art depth estimation models. The comparison includes discriminative depth estimators, generative depth models, and existing joint image-depth generators. Discriminative depth estimators include the most performant models such as MoGe-2 [334] and Depth Pro [23]. We also include generative depth estimators such as Pixel-Perfect Depth (PPD) [366]. Finally, most comparable to Modality Forcing, we directly compare against methods which jointly generate image and depth such as JointDiT [30].

The results in Table 8.3 suggest that Modality Forcing convincingly outperforms existing joint image-depth models and generative depth models. We attribute the performance delta to the scalability of our recipe to real-world depth data, the strong prior from the FLUX.2-klein-9B backbone, and our post-training recipe. Modality Forcing is competitive even with the strongest depth models such as PPD [366], Depth Anything V2 [374] and MoGe-2 [334]. Modality Forcing outperforms all prior baselines on NYUv2 [273] and ETH3D [268]. Our performance on ScanNet [50] is also strong, but confounded by the ScanNet training split in our data mixture.

Qualitatively, Figure 8.5 shows that the predicted depth is consistent with ground truth. We find that JointDiT [30] infers predictions that violate geometric constraints such as

Table 8.3: **Affine-invariant depth estimation.** Comparison of monocular depth estimation methods on five benchmarks. The results suggest Modality Forcing outperforms all models built for joint RGB-Depth generation, and generative depth estimators. Modality Forcing approaches the performance of the very best depth models such as MoGe-2 [334], scoring better on NYUv2 [273] and ETH3D [268]. The overall best are underlined, within-type best are **bolded**.

		NYUv2 [273]		KITTI [99]		ETH3D [268]		ScanNet [50]		DIODE [313]	
Method Type	Method	AbsRel↓	δ_1 ↑	AbsRel↓	δ_1 ↑	AbsRel↓	δ_1 ↑	AbsRel↓	δ_1 ↑	AbsRel↓	δ_1 ↑
Discriminative depth estimation	ZoeDepth [21]	4.76	97.3	5.59	95.1	7.27	94.2	–	–	7.80	90.9
	MASt3R [157]	4.67	96.7	5.79	95.1	4.64	97.0	–	–	5.79	94.1
	DA-v2 [374]	4.16	97.9	6.77	94.3	4.63	97.2	–	–	5.41	94.6
	Depth Pro [23]	3.67	98.2	5.12	96.8	4.97	96.4	–	–	4.66	95.6
	UniDepth V2 [237]	2.96	**98.6**	3.85	**98.1**	2.95	98.5	–	–	4.05	96.5
	MoGe-2 [334]	**2.89**	**98.6**	**3.75**	**98.1**	**2.80**	**99.1**	–	–	**3.14**	**97.4**
Generative depth estimation	Marigold [143]	4.88	96.8	9.05	90.3	4.90	97.2	5.89	95.2	6.13	94.5
	Lotus [109]	4.31	97.3	8.89	90.1	5.25	96.7	5.01	96.5	6.70	93.8
	GeoWizard [88]	4.89	96.6	10.97	86.6	6.15	95.3	5.49	95.7	6.37	94.0
	PPD [366]	**3.82**	**97.7**	**6.57**	**95.1**	**4.02**	**98.3**	**4.04**	**97.8**	**4.97**	**95.6**
I2D from Joint Model	JointNet [390]	11.92	86.7	13.74	81.3	12.63	87.1	14.81	81.8	20.02	82.7
	UniCon [165]	8.75	91.7	18.67	73.0	8.77	92.9	9.23	91.0	13.57	90.8
	JointDiT [30]	5.12	96.9	10.90	88.4	5.32	97.0	5.76	96.3	10.22	93.9
	Ours	**2.52**	**98.9**	**5.37**	**96.6**	**2.37**	**99.3**	**2.32**	**98.9**	**3.35**	**97.7**

straight lines and vertical walls. We also observe JointDiT's quantization artifacts for distant predictions.

Joint Generation Joint image-depth generation is a key capability enabling prompt-conditioned scene generation for artists, architects and embodied agents. Figure 8.6 shows a comparison between Modality Forcing and the existing SoA joint image-depth generator JointDiT [30]. JointDiT produces stunning images, but the depth predictions are not as robust as the Modality Forcing recipe.

Depth-to-Image Generation (D2I) Depth-to-image generation allows for generating various samples from a single depth map, conditioned on a text prompt. This unlocks generating various scenes with identical layout for embodied AI agents, or assets with identical geometry but different appearance. We follow the evaluation protocol from JointDiT [30] and UniCon [165] by evaluating on 6,000 images in the OpenImages dataset with Depth Anything V2 [374] depth annotations. We conduct two experiments: (1) generate all 6,000 images conditioned on DAv2 depth, and compute the FID vs GT images, (2) compute AbsRel between DAv2(generated images) and DAv2(GT images); this is a proxy for depth-following.

Table 8.4 shows the results. Modality Forcing generates images with the lowest FID, considerably better than all the baselines. We find D2I performance does not match JointDiT [30], as our recipe appears to more loosely follow depth instructions for some scenes.

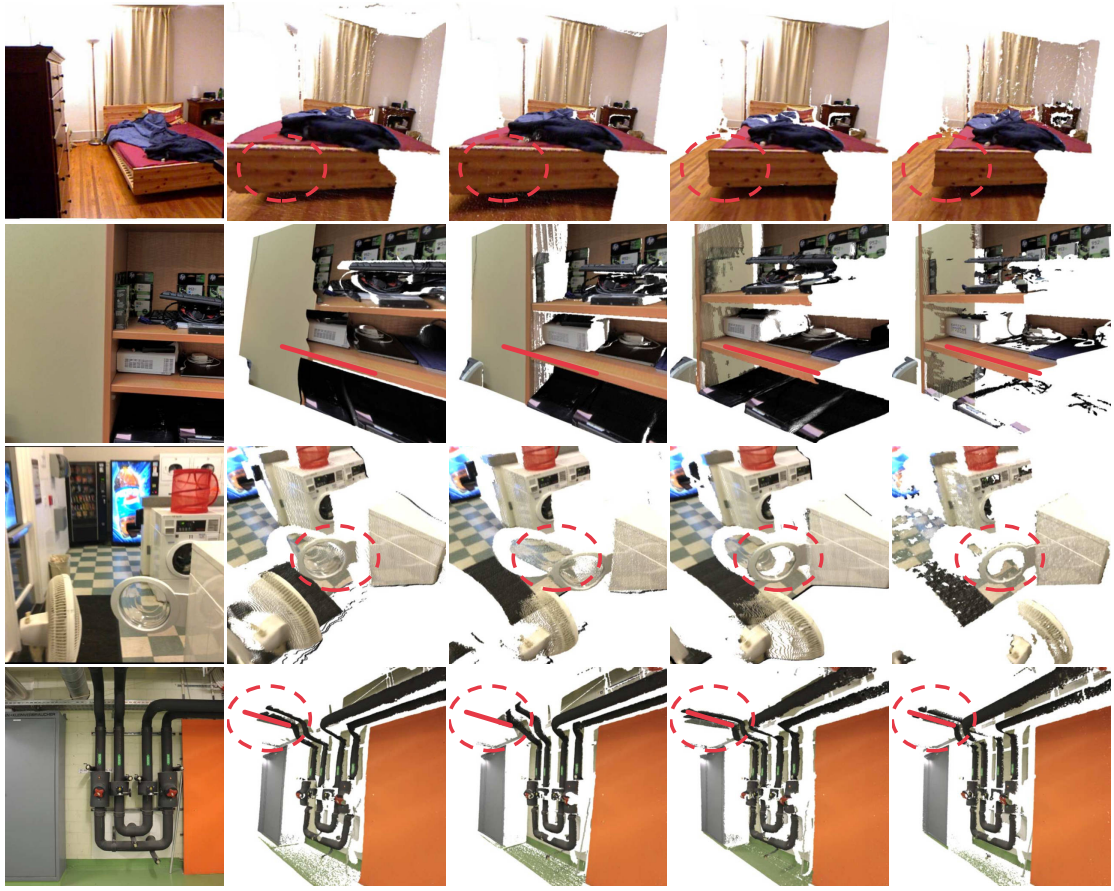


Figure 8.5: Qualitative image-to-depth generation results. Modality Forcing generates robust and sharp depth maps yielding plausible 3D point clouds. We find JointDiT [30] sometimes fails catastrophically, missing structure or misestimating scale. PPD [366] produces much closer results, but Modality Forcing produces more robust results.

Implementation Details Unlike other T2I models, FLUX.2 uses double-stream layers (an image and text stream). We append a depth stream for those layers and freeze the RGB weights for the double stream layers. We train this model at $512 \times x$ variable aspect ratio, where $x \leq 512$.

We use logit-normal sampling for RGB, and plateau logit-normal sampling for depth, both with a timestep shift of $\mu = 1.1$. Additionally, we set $p_{\text{i2d}} = 0.2$ and $p_{\text{d2i}} = 0.2$, to focus the network on the case of a fully denoised image or depth map.

We use the robust and optimal alignment solver (ROE) [334], as well as the ensemble approach documented by Marigold [143] with $N = 10$. We rerun the joint and generative models to match the protocols; we copy the discriminative results from the MoGe-2 paper as they match our protocol.



Figure 8.6: Qualitative joint image-depth generation results. Modality Forcing samples RGB and geometry jointly, and proves to produce consistent and compelling point clouds. JointDiT [30] inherits compelling image generations from FLUX, but struggles with depth.

Table 8.4: Depth-conditioned image generation on 6,000 OpenImages samples.

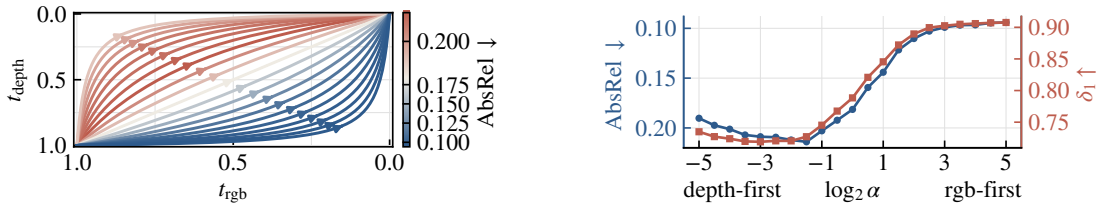
	OpenImages 6K	
Method	FID↓	AbsRel↓
Readout-Guidance	18.72	23.19
ControlNet	13.68	9.85
UniCon [165]	13.21	9.26
JointDiT [30]	12.62	**6.99**
Modality Forcing (Ours)	**11.41**	9.26

8.4.3 Denoising Trajectory Ablations

Modality Forcing allows setting arbitrary per-modality noise levels at inference time, a flexibility we previously showed enables competitive image-to-depth and depth-to-image generation. Here we leverage the same mechanism to extend our analysis to partial depth conditioning and to the model's behavior across a range of different generation trajectories (Figure 8.7).

Depth Quality Encoding per-modality timesteps independently allows for arbitrary denoising trajectories. We study how RGB and depth generations are affected by various trajectories through the noise landscape – prioritizing image or depth tokens early on in the denoising process. We follow Latent Forcing [8] by parameterizing the denoising trajectory according to $f_\alpha(t) = \frac{\alpha t}{1+(\alpha-1)t}$, with $t_{\text{depth}} = f_\alpha(t_{\text{rgb}})$. For $\alpha > 1$, we denoise RGB first, while $\alpha \in [0, 1)$ means depth is prioritized first. We sweep $\alpha \in [2^{-5}, 2^5]$ and complete inference from the OpenImages 6k prompts. We then compute depth metrics between MoGe-2 [334] on the generated image and our own generated depth map. The results

suggest that denoising RGB first yields more consistent depth. This suggests that generating natively in the latent VAE space may be an easier space to traverse than raw depth pixels, similar to what was found in Latent Forcing [8].



(a) Inference-time trajectories studied.

(b) Denoising RGB first leads to better depth prediction.

Figure 8.7: Modality Forcing inference-time analysis. We find that denoising RGB first acts as a kind of 'scratch pad' in latent space, leading to higher-quality depth predictions.

Partial Depth Conditioning In some creative applications, a user may want to control the conditioning strength between modalities — using a depth map to loosely fix coarse object pose, or to dictate precise geometry. Arbitrary denoising trajectories enable this partial conditioning: at every step we set $\mathbf{x}_{\text{depth}} = (1 - t_{\text{depth}})\mathbf{x}_{\text{cond}} + t_{\text{depth}}\epsilon$. Denoising depth earlier means more of RGB generation happens with clean depth, yielding stronger conditioning. Figure 8.8 shows that this mechanism effectively controls depth conditioning strength, and yields compelling and viable results. This simple test-time implementation adds another capability to the same model.

8.5 Conclusion and Limitations

We present Modality Forcing, a simple post-training recipe that turns a T2I model into a unified image-depth generator. The central idea is to assign each modality its own timestep and loss while sharing a single DiT backbone, which enables joint and conditional generation in any combination. Across controlled scaling experiments, the results suggest that stronger T2I pre-training transfers to stronger spatial prediction. Our strongest model based on FLUX.2-klein-9B achieves I2D performance competitive with the very best depth models. This supports the broader view that web-scale generative image models contain spatial structure that can be exposed and continues to scale.

The current model also leaves clear directions for improvement in future work. First, our scaling study is limited to at most 3B parameters and does not derive a full scaling law. Future work may explore scaling to larger T2I models and study scaling behavior in greater detail. Second, our experiments use relatively few depth samples, several orders of magnitude fewer than what is seen during T2I pre-training. Following recent work [326], we will likely see another leap in performance by scaling up depth data and the T2I backbone to

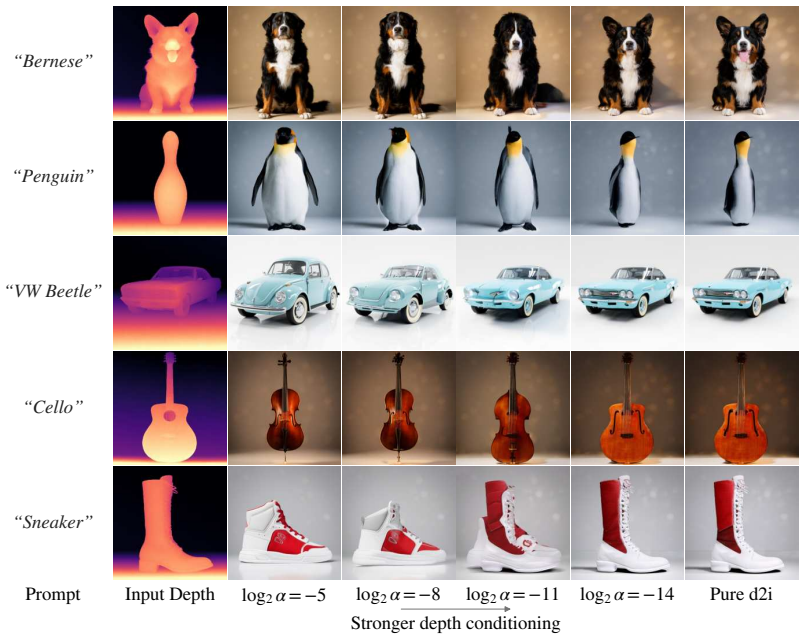


Figure 8.8: The denoising trajectory across depth and RGB dictates the strength of modality conditioning. Denoising more of depth early on means RGB will more rigidly match it. Here we show that this enables a new level of controllability of depth-conditioned image generation.

$>$ 9B parameters. Finally, architectural adjustments may further reduce artifacts or enable prediction of metric depth. With these extensions, we believe Modality Forcing provides a broad foundation for scalable generation across modalities. Applied to depth, Modality Forcing competes with the very best specialist models and is shown to continue to scale.

Chapter 9

PointZero: 3D Point Track Completion for Learning Transferable 3D Dynamics

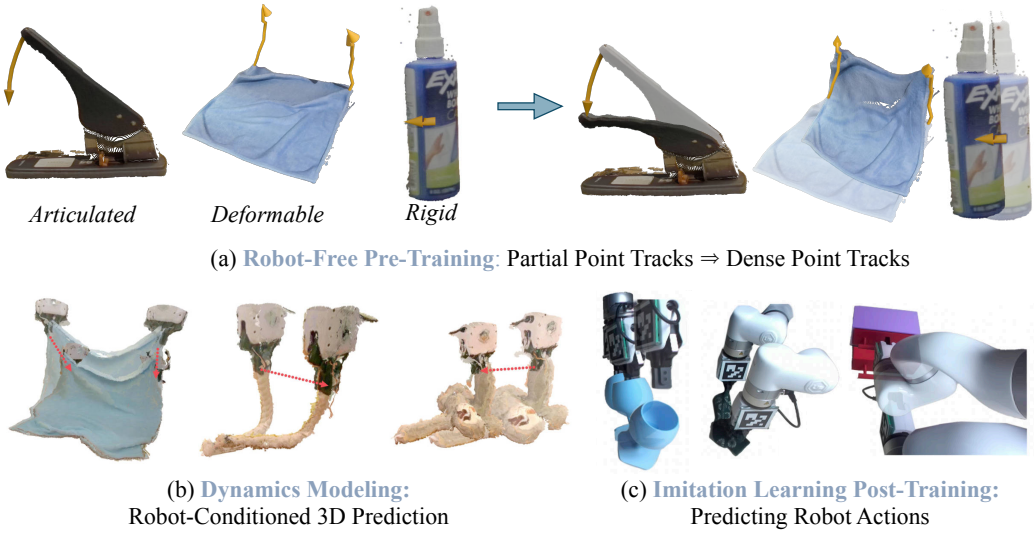
ORIGINALLY APPEARED AS

Bardienus P. Duisterhof, Kaifeng Zhang*, Adam Hung*, Bowen Wen, Stan Birchfield, Yunzhu Li, Deva Ramanan, and Jeffrey Ichnowski.
* Equal contribution.
PointZero: 3D Point Track Completion for Learning Transferable 3D Dynamics [75]. Conference on Neural Information Processing Systems (NeurIPS), 2026.

Chapter in Context

Thus far we have trained generative priors for static scenes, but humans can reason about dynamics from limited observations. Prior work learned similar priors from robot interaction data. Can we pre-train dynamics understanding without robot data? In PointZero, we propose point track completion as a pre-training objective for dynamics understanding. We show that it improves downstream robot manipulation tasks. In SOLF, point track completion provides a *focused* dynamics objective with *obtainable* supervision that requires no robot action labels.

World models endow perceptual systems with the ability to predict how scenes evolve under interaction. They are most beneficial when trained on diverse volumes of data, to instill a rich prior into downstream applications. Existing methods typically require robot action labels to learn action-conditioned 3D dynamics, which excludes web video data from the training pool. We study *3D point track completion* as a pre-training objective for learning transferable 3D dynamics without robot data. Given a single RGB-D observation and sparse partial 3D trajectories (tracks), we predict future 3D tracks of all observed points. We show this objective produces a rich 3D dynamics prior, without requiring robot action labels. We contribute a diverse dataset of 2.9 million synthetic frames spanning deformable, articulated, and rigid objects, and use it to train PointZero. We show that a flexible and expressive transformer, PointZero, outperforms prior methods on the same data. We demonstrate the utility of our pre-training objective by post-training PointZero for two downstream applications: (1) action-conditioned 3D dynamics prediction and (2) imitation learning.



(a) **Robot-Free Pre-Training**: Partial Point Tracks ⇒ Dense Point Tracks

(b) **Dynamics Modeling:** Robot-Conditioned 3D Prediction

(c) **Imitation Learning Post-Training:** Predicting Robot Actions

Figure 9.1: PointZero proposes *3D point track completion* as a pre-training objective for learning rich 3D dynamics priors. Given one RGB-D image and one or a few point trajectories (shown in orange), PointZero predicts the future 3D tracks of all observed scene points. PointZero is a robot-free pre-training objective for instilling 3D dynamics understanding in world models. Post-trained for dynamics modeling [392] and robot manipulation [125], it outperforms *application-specific* baselines.

When fine-tuned to condition on end-effector pose, PointZero outperforms the baselines on the recent PGND 3D dynamics benchmark. When fine-tuned to predict robot actions and 3D tracks, PointZero outperforms or matches the baselines on 6/7 simulated and real-world robot manipulation tasks. We furthermore evaluate training PointZero from scratch to isolate the benefits of our proposed architecture from those of our proposed pre-training objective and dataset. We release the dataset, checkpoints, and full training recipe.

9.1 Introduction

Humans intuitively anticipate how the world responds to interaction: pulling a drawer, folding a sock, or sliding a cup. Endowing perception systems with comparable predictive abilities promises to unlock applications across robotics, simulation, and AR/VR/XR.

Challenges. Learning a generic 3D dynamics model is inherently challenging due to the vast *diversity* of physical phenomena, the *underobservability* of many aspects of physics, and the difficulty of characterizing the space of all possible *interactions*. Despite such challenges, humans readily interact with articulated, rigid, and deformable objects, and can predict plausible dynamics "zero-shot" before interacting with a particular object of interest. Prior methods learn robot-action-conditioned 3D dynamics models by fitting a separate model per scene from hundreds of interactions with the target object [120, 392].

Even hundreds of interactions may be insufficient to learn complex dynamics from scratch, motivating pre-training objectives that provide transferable priors over 3D scene dynamics.

Pre-training. Large-scale pre-training on video data has led to dramatic improvements in data efficiency and generalization capabilities of robot learning systems [355, 397]. However, most such approaches pre-train 2D visual representations or video prediction models, which cannot explicitly represent metric 3D scene motion.

Recent 3D world models have begun scaling dynamics prediction in 3D, but rely on robot interaction data with action labels [121]. Requiring robot action labels makes data collection considerably more expensive and excludes the vast amount of available web video. This raises the question, *can we learn 3D dynamics priors without action annotations?*

3D Point Track Completion. We propose 3D point track completion (Figure 9.1) as a robot-free training objective for learning transferable 3D dynamics. Given a single RGB-D observation and 3D point trajectories for a few sparse points, we predict the 3D point tracks of all observed points. Three properties make the objective scalable. *Robot-free supervision:* The labels and conditioning features are both just 3D point tracks, available from simulation and in principle from any video via point tracking [140, 141]. No robot- or embodiment-specific annotations are needed. *Task-agnostic supervision:* As with other dynamics-modeling objectives, training examples need not correspond to successful task completions. Arbitrary play data or free-form interactions with objects can all provide useful supervision. *Flexible representation:* 3D point tracks can represent arbitrary objects, including those with high-dimensional state such as cloth.

Post-training. We study the utility of our pre-training objective by post-training pre-trained models for two practical applications. First, we adapt the model to condition on robot pose instead of partial point tracks, and train it on scene-specific action-conditioned dynamics modeling using the PGND benchmark [392]. Second, we adapt PointZero for robot manipulation by adding a small head to predict robot actions and supervise with robot manipulation demonstrations.

Zero-shot generalization. We further evaluate whether PointZero learns a *generalizable* dynamics prior by testing the model's zero-shot capabilities. PointZero's architecture consistently outperforms other 3D dynamics-modeling architectures when all methods are trained on our synthetic dataset and evaluated zero-shot on PGND scenes [392] and on a custom test set of unseen real-world objects.

Contributions.

- **Problem formulation:** We introduce 3D point track completion as a flexible and transferable pre-training objective for learning arbitrary 3D dynamics without requiring action labels.
- **Performant task transfer:** When post-trained for *action-conditioned dynamics modeling* and *robot manipulation*, PointZero outperforms strong *application-specific* baselines and training the PointZero architecture from scratch. This suggests that our pre-trained model learns broadly useful 3D dynamics priors that transfer effectively to robotic applications.
- **Dataset:** We release a diverse 4D synthetic dataset with dense per-point trajectory annotations spanning deformable, articulated, and rigid object interactions. We also release the training and inference code for pre- and post-training, as well as pre-trained checkpoints.

9.2 Related Work

Scene-specific Dynamics Models. A common problem setting in prior work is to fit a dynamics model to interaction data collected in a specific scene. Methods grounded in physics such as mass-spring systems [134, 189], the Finite Element Method (FEM) [97], Position-Based Dynamics (PBD) [202], and the Material Point Method (MPM) [166, 285] perform well only when state estimation and system identification are successful. The difficulty in properly identifying system parameters under partial observability and noise has led researchers to explore learning-based approaches. Graph-Based Neural Dynamics (GBND) [171, 258, 391] has shown great promise in simulating complex, high-dimensional deformable objects. However, its reliance on manually designed graph topologies and local message passing limits its applicability across a broader range of objects, particularly to rigid and articulated objects where dynamics involve abrupt changes and constrained motion. Several works explored particle-based predictions without relational edges [120, 302, 351, 392] that can learn dynamics models successfully on challenging materials, but come with a limitation that the methods are only trained on single objects or scenes with predefined physical parameter types. In contrast, PointZero combines a simple transformer architecture with a 3D point track completion objective to learn representations that adapt to scene-specific dynamics in post-training, outperforming these baselines on their own benchmarks.

Large-scale 3D Representation Learning. Large-scale pre-training has transformed 3D computer vision, enabling models to generalize across diverse scenes, objects, and viewpoints. DUSt3R [335] and follow-up works [68, 144, 325] have substantially advanced 3D reconstruction, while other efforts have improved object pose estimation [348], stereo matching [349], and scene completion [73].

Dynamics models trained at scale have primarily focused on image next-state prediction, either in latent [400] or pixel [36] space (e.g., video generation). Scalable 3D dynamics learning has remained largely unexplored, owing to the scarcity of large 3D temporal

datasets.

Concurrent with the research presented in this chapter, PointWorld [121] studies scalable 3D dynamics prediction and can model scene dynamics without scene-specific interaction data or explicit object geometry priors. However, it requires calibrated real-world robot interaction data to train, which limits scalability. PointZero instead trains on a point track completion objective, which provides action-like conditioning without requiring any robot annotations. We imagine future work will leverage diverse sets of data, and we show that 3D data without action annotations is useful.

Pre-training for Robotics. Prior works investigated pre-training visual representations for downstream manipulation policies. One approach is to learn generic visual features using self-supervised learning methods such as masked autoencoding [110, 242] or DINO [35], and then fine-tune on robotics tasks. Rising in popularity is predicting task-conditioned futures [126, 369, 377] across modalities such as video, DINO, point tracks and depth. Other methods learn representations through explicit supervision over manipulation-relevant signals, such as video-language alignment [218], human actions [270], or point trajectories [125, 350]. These works suggest that structuring representation learning around features relevant to manipulation can improve downstream policy learning. Our 3D point track completion objective follows a similar principle: by predicting dense 3D point trajectories, the model is encouraged to learn representations that capture task-relevant object geometry, motion, and dynamics. The closest setting to ours is 3PoinTr [125], which uses task-specific 3D point track prediction as pre-training for downstream manipulation; we compare against 3PoinTr and other baselines in our imitation-learning experiments.

9.3 Problem Statement

We introduce the pre-training objective of *3D point track completion*.

Point cloud observation. Starting from a single RGB-D observation, we use the RGB image $I \in \mathbb{R}^{H\times W\times 3}$, depth map $D \in \mathbb{R}^{H\times W}$, foreground mask $M \in \{0,1\}^{H\times W}$, and camera intrinsics $K \in \mathbb{R}^{3\times 3}$ to unproject the masked depth into an observed 3D point cloud $P^{\text{obs}} \in \mathbb{R}^{N_p\times 3}$, where N_p is the number of foreground pixels.

Partial point track conditioning. We condition the model on a small number of partial 3D trajectories: we select N_a conditioning points over T timesteps, represented as $A \in \mathbb{R}^{T\times N_a\times 3}$, where empirically $N_a \in \{1, 2, 3\}$.

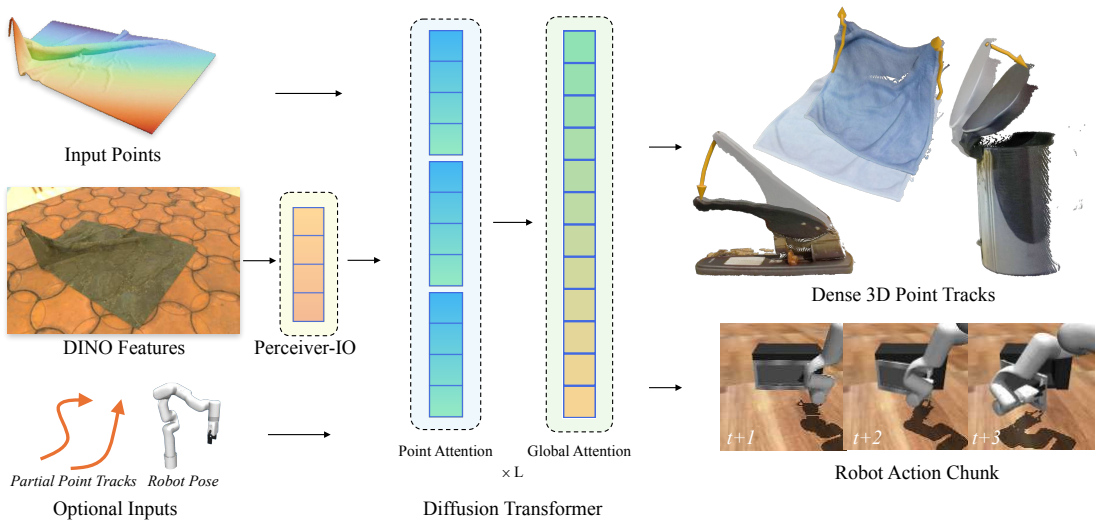


Figure 9.2: PointZero architecture overview. PointZero takes in a single RGB-D image and a small number of complete point tracks, and predicts dense 3D point trajectories. PointZero can also be fine-tuned to condition on robot pose instead of partial point tracks, or to output robot action chunks in addition to point track predictions. PointZero combines a Perceiver-IO [131] architecture with self-attention and cross-attention to encode diverse dynamics.

Prediction. Given the initial observation P^{obs}, visual features extracted from I, and the conditioning A, PointZero completes point tracks $P \in \mathbb{R}^{T \times N_p \times 3}$, with $P_{1,i} = P_i^{\text{obs}}$ and predicted positions at frames $\tau \in \{2, \ldots, T\}$.

The point-level formulation provides a unified dynamics representation for arbitrary object types. The pre-training objective can be used or adapted for several applications, including **partial point track completion:** the trained model directly predicts dense point tracks, given one or more conditioning point tracks. **Post-training for action-conditioned dynamics:** replace the partial point track conditioning with robot end-effector pose conditioning. **Post-training for robot manipulation:** attach a lightweight action prediction head to the pre-trained representation and supervise it with observation-action pairs from expert robot demonstrations.

9.4 Methods

We propose PointZero (Figure 9.2), a single-transformer architecture for modeling action-conditioned 3D dynamics. The model combines (i) point cloud geometry, (ii) action trajectories, and (iii) visual features into a single diffusion transformer (DiT) that predicts the target state from noise.

Point tokens. Given the observed point cloud P^{obs}, we repeat each initial position over the $T-1$ future frames to form $r_i \in \mathbb{R}^{3(T-1)}$, and embed it using a shared MLP E_P, yielding $X_P \in \mathbb{R}^{N_p \times d}$.

Partial point track tokens. For each time step $\tau \in \{1, \ldots, T\}$ and trajectory index $j \in \{1, \ldots, N_a\}$, we embed the 3D coordinate $A_{\tau,j} \in \mathbb{R}^3$ using a shared MLP E_A. We inject positional information using a time embedding $e_t(\tau) \in \mathbb{R}^d$, produced by an MLP from the scalar τ, and a trajectory-index embedding $e_a(j) \in \mathbb{R}^d$, looked up from a learned table of N_a vectors. The resulting action token is $(X_A)_{\tau,j} = E_A(A_{\tau,j}) + e_t(\tau) + e_a(j)$, and we flatten these tokens into a single action sequence $X_A \in \mathbb{R}^{(TN_a) \times d}$.

Visual tokens. We extract dense image features using DINOv2 [225], yielding tokens $F_{\text{DINO}} \in \mathbb{R}^{\lfloor H/14 \rfloor \times \lfloor W/14 \rfloor \times d}$. To reduce memory, we compress them into N_V visual tokens using a Perceiver-IO module [131]: starting from learned latent queries $Z \in \mathbb{R}^{N_V \times d}$, we apply L layers of self-attention over Z and cross-attention from Z to F_{DINO}. This produces $X_V = \text{Perceiver}(Z, F_{\text{DINO}}) \in \mathbb{R}^{N_V \times d}$.

Diffusion transformer (DiT). We use a diffusion transformer with alternating layers of self-attention and cross-attention. For each point i, the query concatenates the current noisy future trajectory $p_{i,t} \in \mathbb{R}^{3(T-1)}$ with r_i, the initial XYZ position repeated $T-1$ times, and projects the result with a shared MLP $E_Q : \mathbb{R}^{6(T-1)} \to \mathbb{R}^d$. This yields $X_{Q,i} = E_Q(\text{concat}(p_{i,t}, r_i))$, with $X_Q \in \mathbb{R}^{N_p \times d}$; a learned diffusion-time embedding is added to each query.

The cross-attention keys and values are formed by concatenating the point, action, and visual embeddings: $X_K = \text{concat}(X_P, X_A, X_V) \in \mathbb{R}^{(N_p + TN_a + N_V) \times d}$.

9.4.1 Training Objectives

We supervise the full trajectory per point to provide a rich supervision signal. We index the $T = 10$ physical frames by $\tau \in \{1, \ldots, T\}$, with the initial observation at $\tau = 1$, and define the flattened future trajectory $p_i = \text{flatten}(P_{i,2:T}) \in \mathbb{R}^{3(T-1)}$, where $P_i \in \mathbb{R}^{T \times 3}$ denotes the ground-truth trajectory of point i in P. For all losses below, $\|\cdot\|_w^2$ denotes the coordinate-averaged, temporally weighted squared error defined in Section 9.7.1; losses are averaged over points and training examples.

Regression. Given the dynamics model g_θ parameterized by θ, we directly regress the trajectory

$$\mathcal{L}_{\text{reg}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| g_{\theta,i}(P^{\text{obs}}, A, I) - p_i \right\|_w^2 . \tag{9.1}$$

Flow matching (v-prediction). For flow matching [184], we sample an isotropic Gaussian source point $p_i^{\text{src}} = \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, 0.2^2\mathbf{I}_{3(T-1)})$ in normalized coordinates, and interpolate between source and target as $p_{i,t} = (1-t)p_i^{\text{src}} + tp_i$ for diffusion time $t \in [0,1]$. The target velocity is $v_i^\star = (p_i - p_{i,t})/(1-t) = p_i - p_i^{\text{src}}$. Let $P_t \in \mathbb{R}^{N_p \times (T-1) \times 3}$ be the reshaped tensor corresponding to $\{p_{i,t}\}_{i=1}^{N_p}$. We train the model to predict $v_i^\star$:

$$\mathcal{L}_{\text{FM}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| g_{\theta,i}(P_t, P^{\text{obs}}, A, I, t) - v_i^\star \right\|_w^2 . \tag{9.2}$$

JiT-style (x-prediction). Flow matching and diffusion models for image generation are often applied in learned autoencoder latent spaces, enabling generation in a more compact and smooth representation [53, 252]. In contrast, Just image Transformers (JiTs) [164] perform generation directly in pixel space. JiT argues that natural images lie on a low-dimensional manifold within pixel space, and therefore trains the model to directly predict the denoised image x_0.

Similarly, PointZero does not use a low-dimensional VAE to encode 3D flow, and we hypothesize that the data also lies on a low-dimensional manifold. Accordingly, we follow JiT and train the model to perform x-prediction: g_θ outputs the trajectory $\hat{p}_i = g_\theta(P_t, P^{\text{obs}}, A, I, t)_i$. Following the JiT training objective, we supervise the implied velocity $\hat{v}_i = (\hat{p}_i - p_{i,t})/(1-t)$ against the target velocity $v_i^\star$:

$$\mathcal{L}_{\text{JiT}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| \frac{g_{\theta,i}(P_t, P^{\text{obs}}, A, I, t) - p_{i,t}}{1-t} - v_i^\star \right\|_w^2 . \tag{9.3}$$

Following JiT, we clip the denominator $1-t$ to a minimum of 0.05 when computing both target and predicted velocities during training.

9.4.2 Dataset

A major bottleneck for scaling 3D dynamics learning is the absence of standardized datasets that capture action-conditioned, physically grounded interactions across diverse object types. Unlike video prediction, 3D dynamics learning requires temporally consistent geometry, explicit object states, and controllable interventions, making data collection particularly costly and fragmented across prior work. To this end, we contribute a new synthetic dataset spanning deformable, articulated, and rigid objects with motions driven by randomized interactions (Figure 9.3). Details regarding our dataset generation are provided in Section 9.7.3.

9.4.3 Post-training Recipes

We study the utility of 3D point track completion as a pre-training objective for downstream tasks.



Figure 9.3: We contribute a novel dataset to learn diverse dynamics in simulation and the real world. The dataset contains 2.9 million synthetic image frames with deformable, articulated and rigid objects. We also collect a dataset with 14 objects and 124 interactions in the real world for evaluation of zero-shot sim-to-real transfer.

Post-training on end-effector state. For accurate scene-specific dynamics, we replace the conditioning on sparse partial point tracks with conditioning on robot end-effector (EEF) pose. Concretely, we swap the action tokens X_A for EEF tokens that encode the 6-DoF pose and gripper state at each timestep, embedded by a new MLP E_{EEF}. We initialize compatible components from the pre-trained model and fine-tune on the target scene.

Imitation-learning post-training. For policy learning, we remove partial point track conditioning and add an action head π_ϕ that predicts an EEF trajectory. With point-track supervision, we jointly predict EEF and point trajectories. See Section 9.5.7 and Sections 9.7.4 and 9.7.9 for task-specific recipes.

9.5 Results

9.5.1 Implementation Details

Normalization and augmentations. We normalize each point cloud using the initial-frame centroid and 99^{th}-percentile radius, and invert this transformation before evaluating errors in physical units. We then apply random rotations and Gaussian noise to both the point clouds and partial point track conditioning points to enhance robustness to noisy real-world measurements. We also apply visual augmentations to image observations, including random changes in brightness and contrast, salt-and-pepper noise, and Gaussian noise.

Training. We train PointZero on a node of 8× H100 GPUs for approximately 2 days per variant. Following JiT [164], we use logit-normal time sampling and additionally set $t{=}0$ with probability $p_{t=0} = 0.2$. At test time, both flow matching and JiT use four forward-Euler updates on a deterministic logit-normal quantile time grid with parameters $\mu = -3$ and $\sigma = 1$. We adopt a ViT-Base backbone (768 token dimension, 12 heads, 12 layers) with a 3-layer Perceiver-IO encoder and 4 query tokens in the pre-training checkpoints. Post-training omits DINO visual conditioning for efficiency.

9.5.2 Baselines

We implement several strong existing learning-based dynamics methods, all *trained on our training dataset* with identical augmentations and a prediction horizon of $H = 10$. All methods use the same inputs, conditioning signals, output targets, and supervision; in particular, each baseline is adapted to condition on the same partial point tracks as PointZero.

Graph-Based Neural Dynamics (GBND) [393]: GBND downsamples the point cloud to 100 particles per scene, connects edges based on top-k nearest neighbors ($k = 5$) to form a spatial graph, and applies a GNN to predict next-step per-particle velocities. GBND is rolled out sequentially for longer predictions.

ParticleFormer [120]: ParticleFormer uses the same input as GBND but uses a transformer for particle encoding and predicted velocity decoding.

Particle-Grid Neural Dynamics (PGND) [392]: PGND uses PointNet [240] and a grid-based representation to predict next-step particle velocities.

Point Transformer v3 (PTv3) [357]: PTv3 predicts next-step particle motions and performs sequential prediction in a similar way to PGND, but with a transformer architecture.

9.5.3 Evaluation Metrics

Following prior work [120, 392, 393], we evaluate prediction accuracy using the final-timestep point clouds. Given ground-truth points G and predicted points P, we report the mean squared error (MSE), mean distance error (MDE), the bi-directional Chamfer Distance (CD), and the Earth Mover's Distance (EMD). Formal definitions of these metrics are provided in Section 9.7.2.

Table 9.1: Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in 10^{-2} m^2. **Bold** marks the best result in each column, and underlining indicates the second-best result.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND [393]	17.38	6.15	7.45	16.47	11.15	6.46	7.38	11.11	155.72	759.93	93.30	154.44
ParticleFormer [120]	17.35	6.15	7.42	16.44	10.86	6.46	7.11	10.83	155.04	760.43	92.76	153.77
PGND [392]	9.01	1.36	4.10	7.86	8.70	2.13	5.40	8.29	67.58	123.79	33.56	64.02
PTv3 [357]	9.20	1.59	3.91	8.28	8.25	2.31	5.25	8.00	61.97	114.77	29.87	58.57
PointZero-FM-mean-10	3.59	0.25	1.55	3.18	2.40	0.36	1.77	2.34	27.26	34.25	14.17	25.41
PointZero-FM-oracle-10	**2.80**	**0.15**	**1.24**	**2.41**	1.96	0.25	1.48	1.91	19.91	18.80	10.39	18.43
PointZero-Regression	4.08	0.32	1.74	3.63	3.85	0.66	2.57	3.68	32.14	44.58	16.43	29.78
PointZero-JiT-mean-10	3.62	0.26	1.56	3.22	2.36	0.37	1.70	2.30	28.06	39.25	14.65	26.35
PointZero-JiT-oracle-10	2.85	0.16	1.26	2.47	**1.95**	**0.24**	**1.44**	**1.89**	**19.46**	**17.89**	**10.30**	**17.99**

9.5.4 Synthetic Evaluation

We evaluate on held-out samples of our training dataset: approximately 32k scenes across all object categories. We also provide ablations over model and data size in Section 9.7.5.

Comparison against baselines. Table 9.1 shows that every PointZero variant outperforms every baseline on every metric and object category. This suggests that a flexible but expressive transformer, generating entire motion sequences, is highly performant for learning diverse dynamics.

Training objectives. Generative models are capable of modeling arbitrary distributions given incomplete observations. We generate ten seeded predictions and report the lowest error independently for each scene and metric. This is a ground-truth oracle. The generative objectives, FM and JiT, outperform the regression variant, suggesting that they capture a multimodal distribution over plausible point trajectories. More comprehensive statistics and analysis are available in Section 9.7.6.

In our experiments, JiT does not consistently outperform flow matching (PointZero-FM) when both models are trained for sufficiently long.

9.5.5 Real-World Zero-shot Partial Point Track Completion

We evaluate zero-shot generalization to real-world objects: methods are trained only on our simulation dataset and tested on novel real-world scenes without fine-tuning.

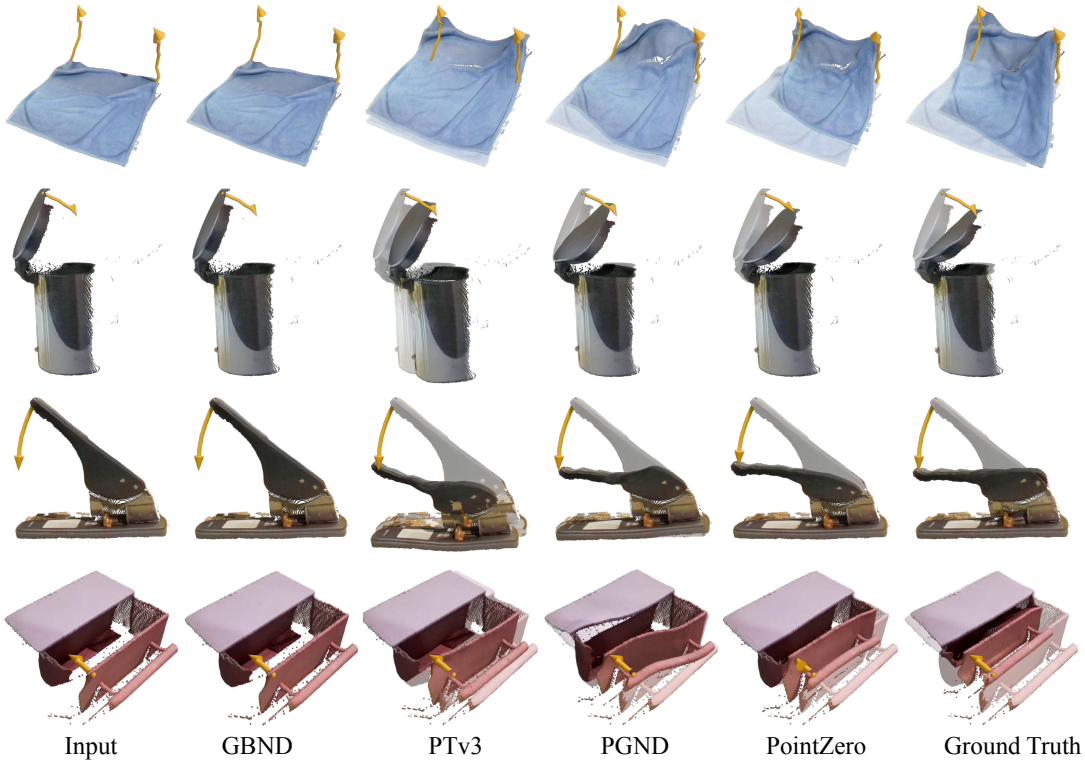


Figure 9.4: Real-world qualitative 3D point track completion evaluation. All methods are trained with identical inputs and conditioning, and with identical data. We find GBND often predicts zero motion, while PTv3 and PGND struggle to recover rigid (articulated) motions.

Data collection. We collect 124 interactions across 14 objects: 6 articulated, 5 deformable, and 3 rigid objects, where a human operator manipulates each object. We obtain action and point-track trajectory labels using FoundationStereo [349] and CoTracker3 [140]. To obtain action labels, we manually annotate one human-object contact point per human hand in the first frame and forward-propagate the trajectories.

Analysis. We display qualitative results in Figure 9.4. Compared to baselines, PointZero excels at adhering to the input trajectory actions (orange arrows) it is conditioned on. PointZero point track completions are also more faithful to object-specific physical structure, with better preservation of cloth surface area and less deformation of rigid bodies. Table 9.9 shows that both FM and JiT outperform all baselines on 11 of 12 real-world metrics; PTv3 wins on rigid-object MSE. This suggests that for learning diverse dynamics, simple transformer architectures beat networks with inductive biases such as spatial grids [392].

Table 9.2: Dynamics prediction errors on the PGND benchmark [392] (centimeters). The PGND method is trained only on scene-specific interactions. We evaluate PointZero both trained from scratch on the scene-specific interactions (PointZero-Scratch), and also fine-tuned (PointZero-FT) on the scene-specific interactions from the pre-trained checkpoint. Both PointZero rows report a per-scene, per-metric best-of-10 oracle (Section 9.5.6). **Bold** marks the best result in each column.

Method	Bread			Paperbag			Cloth			Box			Rope			Sloth		
	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓
GBND [391]	3.1	3.1	1.6	3.0	4.2	1.6	7.7	8.3	3.5	4.5	6.2	3.2	6.2	7.3	3.6	7.8	6.4	3.2
PGND [392]	2.0	1.8	1.0	**1.6**	2.1	0.9	4.5	4.3	2.2	**2.2**	**1.5**	1.6	3.9	3.8	2.1	4.3	3.3	1.7
PointZero-Scratch	4.2	3.9	1.9	4.2	4.4	1.9	5.6	4.7	2.1	3.7	3.9	2.1	9.3	10.8	5.0	7.3	5.9	2.8
PointZero-FT	**1.5**	**1.1**	**0.6**	**1.6**	**2.0**	**0.8**	**4.0**	**3.4**	**1.5**	2.4	2.7	**1.4**	**3.3**	**3.3**	**1.6**	**3.9**	**3.0**	**1.5**

9.5.6 Post-training: Robot-Conditioned 3D Dynamics Prediction

3D point track completion is an incomplete approximation of contact-rich real-world dynamics, as is common in robot manipulation. We post-train PointZero (Section 9.4.3) to predict 3D dynamics conditioned on robot end-effector state, and study the performance of its pre-trained spatial prior. We evaluate PointZero on the six-scene benchmark from PGND [392] and compare against strong *application-specific* baselines. For each scene, we post-train a separate instance of PointZero using the same training set used to train the PGND baseline. We post-train with the JiT objective and LoRA. We also train PointZero-Scratch with the same objective, random initialization, and all parameters trainable. The results in Table 9.2 suggest two things: (1) PointZero convincingly outperforms state-of-the-art baselines on 4/6 scenes, and (2) pre-training is critical to its strong predictions.

We additionally evaluate PointZero in a zero-shot setting on the PGND benchmark. It achieves competitive performance with scene-specific baselines, *despite never having interacted with the objects.* We provide these results and additional details in Section 9.7.8.

9.5.7 Post-training: Imitation Learning for Robot Manipulation

We test whether the trained representation effectively adapts to robot manipulation using the experimental evaluation protocol from 3PoinTr [125]. We independently post-train PointZero to predict robot actions in three simulation tasks and four real-world tasks. For each task, we use 20 expert demonstrations with robot action labels, and 100 additional videos of expert demonstrations without any action labels. We provide details about the tasks and policy-training setup in Section 9.7.9.

Table 9.3: Imitation learning success (%). PointZero uses 20 labeled demos and 100 actionless videos per task.

Task	3PoinTr	DP3	DP	ATM	PointZero
Simulation					
Blockstack	90.9	44.4	45.4	40.0	**99.8**
Microwave	80.8	31.4	32.0	30.4	**93.1**
Glass	95.2	74.9	57.5	3.9	**95.9**
Real-world					
Drawer	90.0	70.0	–	30.0	**100.0**
Cup	**100.0**	80.0	–	30.0	**100.0**
Paper	**90.0**	10.0	–	0.0	70.0
Sock	**90.0**	20.0	–	0.0	**90.0**

Table 9.4: Pre-training **with** downstream track supervision. Success (%); 20 demos, no extra videos.

Recipe	Blockstack	Microwave	Glass	Avg.
Scratch	**93.5**	81.4	66.6	80.5
Pre-trained	93.2	**91.9**	**79.5**	**88.2**

Table 9.5: Pre-training **without** downstream track supervision. Success (%); 20 demos, no extra videos.

Recipe	Blockstack	Microwave	Glass	Avg.
Scratch	83.8	89.5	49.1	74.1
Pre-trained	**85.2**	**90.1**	**64.6**	**80.0**

Baselines. **DP3** [386] and **Diffusion Policy** [45] are performant behavior cloning methods that encode observations and then use conditional diffusion models to generate action chunks. **3PoinTr** [125] and **ATM** [350] pre-train point-track prediction models from videos, and then condition policies on the point-track predictions.

Analysis. PointZero achieves the highest or joint-highest success rate on six of seven manipulation tasks (Table 9.3). Using only 20 action-labeled demonstrations per simulation task and no additional actionless videos, pre-training increases average success from 80.5% to 88.2% with auxiliary point-track supervision (Table 9.4); without it, the pre-trained recipe achieves 80.0% versus 74.1% from scratch (Table 9.5). The latter recipe freezes the pre-trained point-processing stream, whereas Scratch trains the full model (Section 9.7.4). Both comparisons use 1,000 rollouts per simulation task; bold entries mark the better variant. The results suggest that 3D point track completion as a pre-training objective helps downstream imitation learning applications. In particular, the isolated pre-training experiments in Table 9.4 and Table 9.5 suggest 3D point-track completion directly improves performance.

9.6 Conclusion

We present PointZero, a 3D dynamics model trained to do *3D point track completion*: given a single RGB-D observation and a small number of sparse point trajectories, the model predicts dense future 3D tracks for all observed points. This objective provides a scalable way to learn dynamics priors without requiring embodiment-specific annotations, using a metric point representation that can represent deformable, articulated, and rigid objects.

Across synthetic and real-world evaluations, PointZero substantially improves dense

point-track completion over adapted 3D dynamics baselines. More importantly, the learned representation transfers effectively to downstream tasks: after post-training, PointZero improves scene-specific action-conditioned dynamics prediction and achieves strong performance on robot manipulation tasks. These results suggest that 3D point track completion can serve as a useful bridge between scalable robot-free dynamics pre-training and downstream embodied intelligence applications.

Limitations and Future Work. PointZero remains limited by the coverage and realism of its pre-training data. Although our synthetic dataset spans several object classes, it does not capture the full diversity of real-world materials, contact-rich hand-object interactions, cluttered scenes, or long-horizon dynamics. Future work should augment simulation with pseudo-annotations from real-world videos using 3D reconstruction and point tracking methods. PointZero also uses relatively simple conditioning signals: partial point tracks during pre-training and end-effector pose during action-conditioned post-training. Richer interaction representations, such as contact locations and forces, may further improve accuracy. Finally, our downstream evaluations remain task-specific and relatively small-scale; future work should study multi-scene and multi-task transfer across a broader range of applications.

9.7 Supplementary Details

9.7.1 Training Loss Weighting

We give more weight to later timesteps to emphasize longer-horizon predictions, where uncertainty is greater. For future frames $\tau \in \{2, \dots, T\}$, we use $\|u_i\|_w^2 = \frac{1}{3(T-1)} \sum_{\tau=2}^{T} [0.1 + 0.9((\tau-1)/T)^2] \|u_{i,\tau}\|_2^2$.

9.7.2 Dynamics Prediction Evaluation Metrics

The definitions below apply to PointZero's corresponding, equal-cardinality final-frame point sets; $\Pi(P, G)$ denotes permutation matrices.

Mean Squared Error (MSE):

$$\mathcal{L}_{\text{MSE}}(P, G) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|P_i - G_i\|_2^2. \tag{9.4}$$

Mean Distance Error (MDE):

$$\mathcal{L}_{\text{MDE}}(P, G) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|P_i - G_i\|_2. \tag{9.5}$$

Bi-Directional Chamfer Distance (CD):

$$\mathcal{L}_{\text{CD}}(P, G) = \frac{1}{2N_p} \sum_{p \in P} \min_{g \in G} \|p - g\|_2 + \frac{1}{2N_p} \sum_{g \in G} \min_{p \in P} \|g - p\|_2. \tag{9.6}$$

Earth Mover's Distance (EMD):

$$\mathcal{L}_{\text{EMD}}(P, G) = \min_{\pi \in \Pi(P,G)} \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \pi_{ij} \|P_i - G_j\|_2. \tag{9.7}$$

For the PGND benchmark, EMD is computed using more points than the other metrics, which can yield lower reported distances.

9.7.3 Dataset Generation Details

Our dataset comprises deformable, articulated, and rigid object interactions. For all datasets, we randomize camera location and intrinsics and use a single-view RGB-D observation. This models the partial visibility and various viewpoints encountered in the real world.

Deformables. We follow prior work [185] to achieve procedural cloth generation.

- **Mesh Generation**: We generate *towels*, *t-shirts* and *shorts* meshes. We randomize both the scale and the proportions.
- **Physics**: We use NVIDIA FleX [202] to model deformable physics. We apply one or two action trajectories to the cloth, modeling interactions such as *fold*, *lift*, *drop*, or *push*. Concretely, given constraints on location of action points, we solve for the location of all other points that would produce static equilibrium. To enhance motion diversity, we randomize physics parameters such as stiffness and drag.
- **Appearance**: We render the scenes with Blender, with random cloth and background textures from PolyHaven [238].

Articulated Objects We use the PartNet-Mobility dataset [360] to retrieve articulated objects, and place them in Genesis [6] for simulating motion. We randomly pick a revolute or prismatic joint to articulate, and set a random configuration in its viable range to produce static equilibrium. This alone may result in an invisible articulation, or very small motions. To address this issue, we reject samples where the 2D optical flow of the sequence is small using privileged information in simulation.

Rigid Objects Finally, we aim to instill an understanding of rigid body dynamics and collision into PointZero. We modify the Kubric engine [100] to include ground-truth 3D trajectories and fewer objects (up to 3 instead of 23 objects).

Sampling Action Trajectories For partial point trajectory completion training, we automate the extraction of partial point trajectories. For deformable objects, we save the action trajectories that were used as kinematic constraints in NVIDIA FleX [202] to be used as training input. For articulated and rigid bodies, we pick 3D point tracks from the k largest displacements.

Dataset Mixing. The recorded training weights are 0.33 each for shorts, T-shirts, and towels, 1.0 for articulated objects, and 0.33 for rigid objects, giving category probabilities of approximately 42.67%, 43.10%, and 14.22%, respectively. This gives an approximate category ratio of $3:3:1$, reflecting our emphasis on learning more complex non-rigid and articulated dynamics while still exposing the model to rigid-body motion and collisions.

9.7.4 Additional Manipulation Comparisons

Tables 9.4 and 9.5 report the pre-training ablations using only 20 demonstrations per simulation task, with no additional actionless videos. In Table 9.4, PointZero initializes from JiT pre-training and scratch initializes randomly; both train all parameters with point-trajectory regression and an $\ell_{2,1}$ action loss, using a separate action branch with learned queries. In Table 9.5, both use only the action loss; PointZero keeps its pre-trained point-processing stream fixed while training the remaining parameters, whereas scratch trains the full model. With downstream track supervision, pre-training improves two tasks and the average (80.5% to 88.2%), while Blockstack changes from 93.5% to 93.2%. Under this protocol, Diffusion Policy [45] and DP3 [386] achieve average success rates of 45.0% and 50.2%, respectively; their per-task results are listed in Table 9.3. Without downstream point-track supervision, pre-trained PointZero outperforms its scratch variant on all three tasks, with the largest gap on Glass (64.6% vs. 49.1%). DP3 outperforms PointZero without downstream track supervision on Glass (74.9% vs. 64.6%).

9.7.5 Ablations

We study the impact of the training data fraction and of model size.

Training data fraction. Table 9.6 reports performance on the articulated split of the real-world set as the pre-training dataset is reduced, with training epochs scaled inversely to hold the optimization budget comparable. Performance degrades gracefully: reducing the data to 10% increases MDE from 2.2 to 3.1 cm, and further reduction to 5% yields only a small additional drop. This suggests the pre-training objective extracts useful dynamics priors even from a fraction of the dataset, while the full dataset remains clearly beneficial.

Table 9.6: Data-scaling ablation on the articulated split of the real-world dataset: performance degrades gracefully as the pre-training dataset is reduced. Error values are in centimeters; MSE is in cm^2. Training epochs are scaled inversely with the data fraction (250 epochs at 10%, 500 at 5%). **Bold** marks the best result for each metric.

Data Fraction	MDE↓	MSE↓	CD↓	EMD↓
100%	**2.2**	**12.1**	**1.4**	**2.0**
10%	3.1	49.6	2.0	2.9
5%	3.3	64.3	2.0	3.1

Model size. We additionally train a Small variant from scratch (384 token dimensions, 6 heads, 6 layers) with otherwise identical inputs and training data. Table 9.7 reports performance in the same setting: shrinking the model degrades all metrics substantially, suggesting that further scaling the Base model could yield additional gains.

Table 9.7: Model-size ablation on the articulated split of the real-world dataset. Small uses 384 token dimensions, 6 heads, and 6 layers (45M parameters) versus the Base configuration (768 dimensions, 12 heads, 12 layers). Error values are in centimeters; MSE is in cm^2. **Bold** marks the best result for each metric.

Size	MDE↓	MSE↓	CD↓	EMD↓
Base	**2.2**	**12.1**	**1.4**	**2.0**
Small	3.1	37.4	1.8	3.0

9.7.6 Additional Synthetic Data Point Dynamics Completion Results

In Table 9.8, we expand Table 9.1 with additional evaluations of PointZero-JiT and PointZero-FM. For each sample, we generate ten predictions using different random noise initializations. The "-oracle-10" rows select the lowest error separately for each scene and metric; "-first" uses the first prediction, and "-mean-10" averages scalar errors over all ten predictions. For "-medoid-10", each corresponding point's final position is selected from its ten sampled positions to minimize the summed Euclidean distance to the other nine, without using ground truth. Different points may come from different samples. The "-pooled-scalar-median-10" rows take each metric's median over scene–prediction pairs within each dataset, then average dataset medians. These distribution summaries are not directly comparable to mean-error rows.

PointZero-JiT-first outperforms PointZero-Regression on 11 of 12 metrics; the exception is rigid-object MSE (48.627 versus 44.576 in $10^{-2}\,m^2$). Both per-metric best-of-10 oracles

Table 9.8: Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in $10^{-2}\,\mathrm{m}^2$. First-sample, mean, pointwise medoid, per-metric oracle, and pooled scalar median statistics use ten seeded predictions and are defined in Section 9.7.6. Pooled medians summarize a different statistic from mean errors; no cross-statistic ranking is applied.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND [393]	17.38	6.15	7.45	16.47	11.15	6.46	7.38	11.11	155.72	759.93	93.30	154.44
ParticleFormer [120]	17.35	6.15	7.42	16.44	10.86	6.46	7.11	10.83	155.04	760.43	92.76	153.77
PGND [392]	9.01	1.36	4.10	7.86	8.70	2.13	5.40	8.29	67.58	123.79	33.56	64.02
PTv3 [357]	9.20	1.59	3.91	8.28	8.25	2.31	5.25	8.00	61.97	114.77	29.87	58.57
PointZero-Regression	4.08	0.32	1.74	3.63	3.85	0.66	2.57	3.68	32.14	44.58	16.43	29.78
PointZero-FM-first	3.60	0.25	1.56	3.17	2.39	0.35	1.77	2.33	28.97	38.79	15.16	27.20
PointZero-FM-mean-10	3.59	0.25	1.55	3.18	2.40	0.36	1.77	2.34	27.26	34.25	14.17	25.41
PointZero-FM-pooled-scalar-median-10	3.21	0.14	1.30	2.80	1.75	0.05	1.47	1.74	11.65	1.85	7.03	11.27
PointZero-FM-medoid-10	3.26	0.21	1.40	2.87	2.23	0.33	1.65	2.18	24.46	28.57	12.59	22.74
PointZero-FM-oracle-10	2.80	0.15	1.24	2.41	1.96	0.25	1.48	1.91	19.91	18.80	10.39	18.43
PointZero-JiT-first	3.65	0.26	1.59	3.24	2.36	0.36	1.70	2.29	30.26	48.63	16.03	28.67
PointZero-JiT-mean-10	3.62	0.26	1.56	3.22	2.36	0.37	1.70	2.30	28.06	39.25	14.65	26.35
PointZero-JiT-pooled-scalar-median-10	3.27	0.15	1.31	2.87	1.65	0.06	1.37	1.64	11.23	1.86	6.96	11.01
PointZero-JiT-medoid-10	3.32	0.22	1.42	2.92	2.23	0.33	1.61	2.17	24.93	33.08	12.81	23.32
PointZero-JiT-oracle-10	2.85	0.16	1.26	2.47	1.95	0.24	1.44	1.89	19.46	17.89	10.30	17.99

outperform regression on all 12 metrics, showing that sampling can produce candidates with lower reconstruction error.

9.7.7 Real-World Dynamics Prediction Quantitative Results

In Table 9.9, we report point track completion error metrics for PointZero and baselines on our real-world dataset. All methods are trained on the same synthetic dataset and evaluated zero-shot on the unseen real-world objects. Both PointZero-FM and PointZero-JiT outperform all baselines on 11 of 12 metrics; PTv3 achieves the lowest rigid-object MSE ($11.604\,\mathrm{cm}^2$, versus $12.376\,\mathrm{cm}^2$ for PointZero-JiT).

9.7.8 PGND Zero-shot Evaluation

We train PointZero and several 3D dynamics architecture baselines on point track completion on the PointZero synthetic dataset. Then, we evaluate the zero-shot performance on the PGND benchmark. To enable zero-shot predictions without fine-tuning the models to condition on robot pose, we extract a single object-point trajectory per PGND sample, and condition on this track.

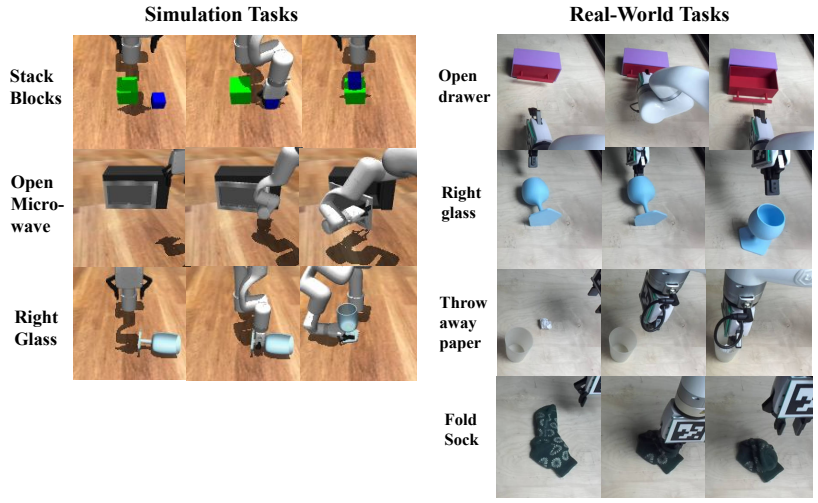


Figure 9.5: Visualizations of the simulation and real-world tasks used for imitation learning evaluation.

Table 9.9: Dynamics prediction errors on the real-world dataset (centimeters; MSE in cm^2). All baselines were trained on the proposed simulation dataset. "-first" rows use a single sample; PointZero-FM and PointZero-JiT report all metrics of the sample with the lowest MDE out of 10 (best-of-10). **Bold** marks the best result in each column, and underlining indicates the second-best result.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND	6.179	81.114	4.039	6.069	3.051	24.148	2.089	2.912	4.379	34.418	2.085	4.307
ParticleFormer	6.156	80.928	4.022	6.044	3.026	24.009	2.078	2.887	4.367	34.332	2.081	4.296
PGND	3.757	27.054	2.244	3.646	2.828	20.137	1.702	2.666	2.650	13.210	1.212	2.546
PTv3	4.705	35.109	2.975	4.606	4.430	43.665	2.645	4.284	2.487	**11.604**	1.108	2.371
PointZero-FM-first	3.739	28.765	2.293	3.642	2.046	10.152	1.168	1.883	2.711	15.697	1.216	2.542
PointZero-FM	2.923	**17.385**	1.852	**2.827**	**1.720**	**8.096**	**1.031**	**1.580**	**2.322**	12.484	**1.078**	**2.182**
PointZero-Regression	3.643	25.258	2.416	3.584	2.192	11.227	1.322	2.060	2.654	14.647	1.133	2.558
PointZero-JiT-first	3.706	32.545	2.294	3.636	2.317	12.608	1.338	2.166	2.815	15.969	1.276	2.656
PointZero-JiT	**2.915**	19.162	**1.837**	2.846	1.914	9.670	1.156	1.767	2.398	12.376	1.103	2.238

Results are shown in Table 9.10. PointZero-FM-ZS achieves an average MDE of approximately 3.8 cm across the six scenes, compared with 5.2 cm for the strongest baseline, PTv3-ZS, a reduction of approximately 26%.

Table 9.10: Evaluation of different 3D dynamics architectures on zero-shot predictions on the PGND benchmark [392]. All methods were trained on synthetic data from the PointZero dataset, and evaluated on unseen PGND scenes. Errors are in centimeters. **Bold** marks the best result in each column.

Method	Box			Paperbag			Cloth			Rope			Sloth			Bread		
	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓
PGND-ZS	3.1	2.0	3.0	2.9	1.7	2.7	5.9	3.0	5.6	8.6	5.1	8.5	8.3	3.1	7.5	2.9	1.3	2.8
GBND-ZS	2.5	1.6	2.4	3.7	2.2	3.6	8.9	4.5	8.7	13.0	8.8	12.9	11.1	4.8	10.3	3.3	1.6	3.3
ParticleFormer-ZS	2.5	1.6	2.4	3.7	2.1	3.6	8.9	4.5	8.7	13.0	8.8	12.9	11.1	4.8	10.3	3.3	1.6	3.3
PTv3-ZS	3.4	2.0	3.3	3.9	2.2	3.7	7.2	3.6	6.9	7.7	4.9	7.6	6.3	2.5	5.2	2.6	1.2	2.5
PointZero-FM-ZS	**2.2**	**1.4**	**2.1**	**2.7**	**1.6**	**2.5**	5.1	2.5	4.9	**5.4**	**3.5**	**5.3**	**5.6**	**2.3**	**4.9**	**1.9**	**0.9**	**1.8**
PointZero-Regression-ZS	2.5	1.6	2.3	2.8	1.7	2.7	4.9	2.5	4.7	5.9	3.6	5.8	6.3	2.5	5.5	2.4	1.1	2.4
PointZero-JiT-ZS	2.3	1.5	**2.1**	**2.7**	**1.6**	**2.5**	**4.7**	**2.4**	**4.4**	5.8	3.8	5.8	6.1	2.6	5.5	2.1	1.0	2.0

9.7.9 Imitation Learning Experimental Setup

Following 3PoinTr [125], we evaluate on three simulation tasks: (1) stacking blocks, (2) opening a microwave, and (3) righting a fallen glass. We also evaluate on four real-world tasks: (1) opening a drawer, (2) righting a fallen glass, (3) picking up a crumpled piece of paper and placing it in the trash, and (4) folding a sock in half. The position and orientation of the objects are varied across trials, and the same initial configurations are used across all methods. Figure 9.5 shows images of all the tasks.

Part III

Spatiotemporal World-Action Models

Chapter 10

3PoinTr: 3D Point Tracks for Learning Manipulation from Unconstrained Human Videos

ORIGINALLY APPEARED AS

Adam Hung, Bardienus Pieter Duisterhof, and Jeffrey Ichnowski. *3PoinTr: 3D Point Tracks for Learning Manipulation from Unconstrained Human Videos [125].* arXiv:2603.08485, 2026.

Chapter in Context

Recent methods have shown that predicting the future in RGB images can be a powerful objective for robot learning. In 3PoinTr, I explore the alternative representation of 3D point tracks. I condition a closed-loop policy on a single 3D point track 'plan' per episode. This, and other design decisions, turn out to be crucial to the success of 3PoinTr. This tests whether *focused* motion plans remain *sufficient* for closed-loop control, linking two SOLF goals.

Learning manipulation policies from human videos could greatly reduce the need for expensive robot demonstrations, but existing approaches typically require restrictive assumptions such as choreographed human motions, predefined keypoints, manual annotations, or known grasp locations. We propose 3PoinTr, a method for pre-training sample-efficient robot policies from *unconstrained human videos* by predicting dense 3D point tracks. In the unconstrained human demonstration videos, humans are free to follow whatever trajectories and manipulation strategies they see fit, rather than choreographing their motions to mimic a robot. 3PoinTr uses a lightweight visibility-aware transformer to learn how scene points should move from human videos, and then trains a closed-loop multitask robot policy to flexibly extract action-relevant priors from those predicted point tracks. With only 20 action-labeled robot demonstrations, 3PoinTr achieves a **25.0** percentage point higher average success rate than the strongest behavior cloning and video-pre-training baselines on real-world tasks, and a **29.6** percentage point higher average success rate in simulation. Targeted ablations support the key design choices and confirm the benefit of learning from actionless

videos. We further show that 3PoinTr's point track prediction transformer outperforms a strong baseline by preserving supervision over partially occluded points. **Project page: https://adamhung60.github.io/3PoinTr/**.

10.1 Introduction

Autonomous robots could take on dangerous, tedious, and high-value tasks, but such systems must be able to generalize across diverse tasks, objects, and settings. A typical approach is to train robots from expert teleoperated robot demonstrations [128, 226], but these demonstrations are expensive to collect. Human videos are cheap to record and abundantly available on the internet with large diversity. Such videos encode the rich physical and behavioral knowledge that robots must acquire to perform tasks, and thus a central question is how to extract these priors from human videos for robot learning.

Efficiently extracting these priors is challenging due to the embodiment gap: humans have very different kinematics from parallel-jaw gripper robot setups. As a result, videos of humans performing tasks often exhibit manipulation strategies and motions that are infeasible, inefficient, or even dangerous for robots. Furthermore, while 3D methods better capture the scene geometry and align with the robot's 3D action space, existing 3D methods rely on carefully curated human demonstrations to limit the embodiment gap. Overcoming these challenges would help unlock learning from human videos at scale, with the potential to dramatically lower the cost of teaching robots new manipulation tasks.

We propose **3PoinTr** (Figure 10.1), a method for learning manipulation policies from *unconstrained* human videos. Unconstrained means that the human can act freely and naturally rather than mimicking target robot kinematics. 3PoinTr takes in a point cloud of the scene, and answers the question: *how will the 3D scene evolve when completing this task?* 3PoinTr predicts these 3D point tracks using a lightweight transformer, and trains a policy conditioned on the tracks. We show that several key design choices make this a robust recipe even with a large embodiment gap. As a result, when trained with 20 action-labeled robot demonstrations per task, 3PoinTr outperforms the baselines with a **25.0** percentage point higher average success rate on our real-world tasks and a **29.6** percentage point higher average success rate in simulation.

3PoinTr uses a Perceiver-IO [131] architecture to compress the dense point tracks into a compact set of point-track tokens that serve as a static *open-loop plan* for the task. A *closed-loop* policy then re-queries on the current observation at each control step, providing reactivity while remaining anchored to the open-loop plan. Instead of imposing biases such as human-to-robot keypoint mappings or object-specific features, *learned queries autonomously discover task-relevant features*, enabling sample-efficient policy learning.

Prior work has explored representations based on point tracks (also known as *flow* and *scene flow*) [20, 49, 61, 78, 106, 121, 159, 248, 350, 367, 382, 399], and recent point-tracking foundation models [139] have enabled autonomous extraction of point tracks from videos [121]. Many of these works rely on carefully collected videos where the human demonstrator's arm must closely mimic robot kinematics, often involving forming a claw

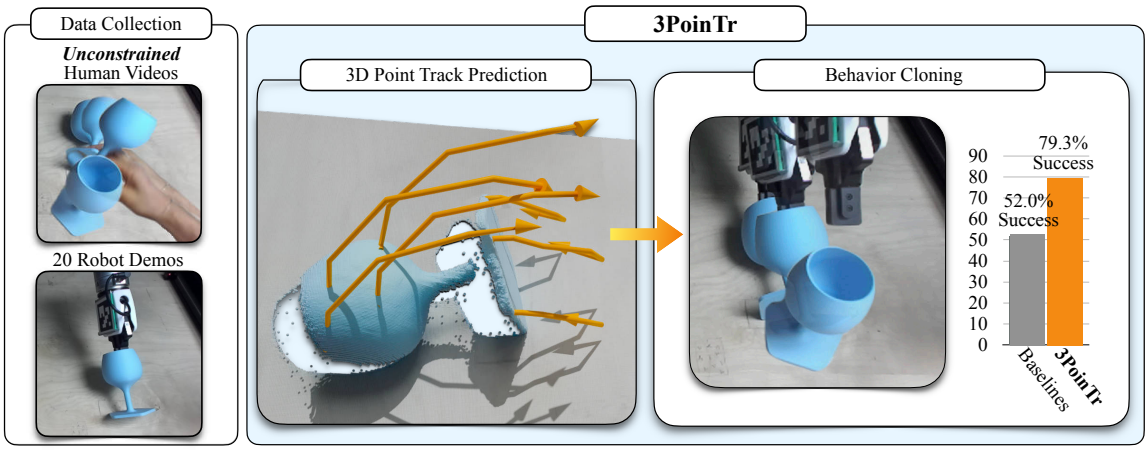


Figure 10.1: **3PoinTr** learns manipulation from *unconstrained* human videos: videos where the human demonstrator can act freely rather than mimicking target robot kinematics. 3PoinTr first predicts dense 3D point tracks — *how the scene should move to complete the task* — and then conditions a closed-loop multitask policy on these tracks. 3PoinTr outperforms strong behavior cloning and learning-from-video baselines across simulated and real-world evaluations.

shape with their hands to imitate a parallel-jaw robot gripper. Additionally, most methods remain 2D, even though 3D state representations have been shown to improve sample efficiency [386, 404]. The few methods that use 3D point tracks [106, 382] cannot learn from unconstrained videos, and often require additional biases such as manual annotations [382], predefined keypoints [106], or goal-image conditioning [20]. Instead of constraining the human demonstrator, 3PoinTr predicts 3D point tracks of the scene with the embodiment points removed and autonomously *extracts action-relevant priors* from these tracks, leaving the human free to use any grasp, hand pose, or trajectory natural to them.

This chapter contributes:

- The first method for learning closed-loop robot policies from *unconstrained human videos* using dense 3D point tracks as an intermediate representation.
- A state-of-the-art track-conditioned *policy architecture* that compresses predicted point tracks into compact point track tokens and injects them through both global conditioning and per-block residual conditioning in the action head, enabling sample-efficient policy learning. We ablate the crucial design choices.
- A simple *visibility-aware transformer* for predicting dense 3D point tracks, which outperforms a strong baseline [382] under two 3D displacement-based metrics on our real-world data.

10.2 Related Work

3PoinTr builds on prior work on behavior cloning and learning from videos.

Behavior cloning. Behavior cloning succeeds across a wide variety of real-world tasks [45, 396], but requires substantial teleoperated data to achieve robustness. Structured representations such as point clouds [386, 404] or keypoints [314] can improve sample efficiency, but still must learn to map observations to actions from limited robot demonstrations. Conversely, 3PoinTr improves sample efficiency by first predicting 3D point tracks from ample video data. These 3D point tracks encode rich priors about task specification and scene dynamics.

Learning from videos. Actionless videos are easy to collect and encode useful task structure. Prior work uses them for visual pre-training [218], reward learning [385], affordance extraction [10, 37], and conditioning robot policies on predicted or latent video dynamics [175, 227]. However, these methods are sensitive to visual perturbations, and they typically learn human-centric features from human videos which do not necessarily transfer well to learning robot policies. 3PoinTr instead learns a 3D point track representation that is aligned with 3D robot actions, agnostic to appearance changes, and flexibly learns from off-embodiment demonstrations.

Point track representations for robot learning. Point tracks provide a natural bridge between actionless videos and robot policies, encoding scene motion in an interpretable, spatially grounded form. Prior work on using point tracks for learning manipulation from videos differs along three axes: (1) *what is assumed of the human video*, (2) *how the predicted tracks drive the policy*, and (3) *2D vs. 3D point tracks*.

On the first axis, some methods retarget predicted human hand keypoints directly to the robot end-effector [106, 248]; this requires choreographed videos in which the demonstrator precisely follows the target robot motion under a fixed human-to-robot keypoint mapping. Other methods predict dense tracks of all points [49, 350], which implicitly assumes that human motion closely aligns with robot motion; otherwise, embodiment point tracks induce distribution mismatch between human and robot motion. Im2Flow2Act [367] and General Flow [382] circumvent this assumption by predicting object-only tracks, but Im2Flow2Act requires a large robot play dataset in a digital twin, and General Flow relies on manual annotations and a predefined grasp. 3PoinTr makes no such assumptions about the video data: it predicts dense 3D point tracks over all non-embodiment points, and our proposed architecture autonomously discovers which aspects of the predicted motion are action-relevant.

On the second axis, retargeting and heuristic-execution methods [106, 248, 382] predict open-loop point tracks and execute them directly; this approach lacks reactivity to execution-time error or perturbation. Track-conditioned policies [49, 350] and model-based planning methods [61, 121, 159, 399] that re-predict tracks at every control step regain closed-loop

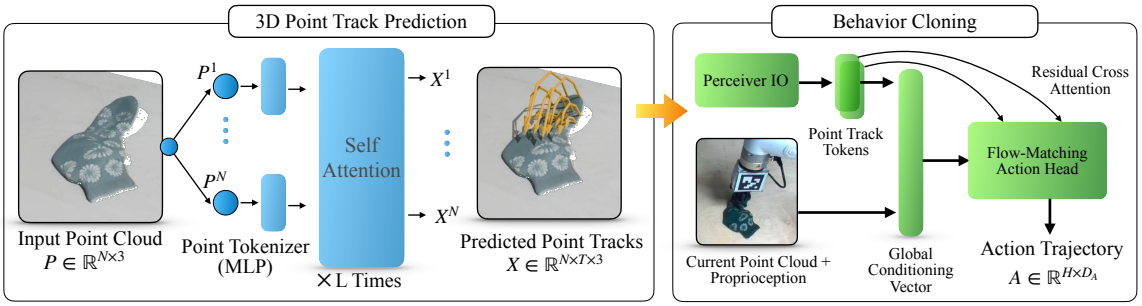


Figure 10.2: Diagram of the 3PoinTr network architecture. Given an initial point cloud, a transformer predicts dense 3D point tracks describing how objects should evolve during the task. A Perceiver-IO point track encoder compresses these tracks into a small set of learned point track tokens. The policy conditions on these point track tokens together with the current point cloud and robot state, enabling closed-loop action prediction. Beyond global conditioning, residual track-token cross-attention in every U-Net block gives the action head a direct path to the predicted object motion, encouraging a simpler mapping from point tracks to robot actions.

reactivity, but their track predictions suffer from human-to-robot distribution shift, and any temporarily occluded points are excluded from the conditioning. 3PoinTr instead predicts point tracks open-loop and uses the predictions to condition a closed-loop policy, maintaining reactivity while avoiding distribution shift and occlusion issues. The closest prior work is Im2Flow2Act [367], which similarly pairs an open-loop track predictor with a closed-loop policy. However, in addition to the simulation requirements discussed above, it also requires execution-time object segmentation and point tracking, reintroducing potential occlusion issues.

On the third axis, most point-track-based methods are 2D, while a few extend to 3D [106, 382] and report improved sample efficiency. But as discussed previously, General Flow [382], Point Policy [106], and similar methods cannot learn from unconstrained human videos.

10.3 Method

3PoinTr (Figure 10.2) consists of two stages: (1) learning a model that predicts dense future 3D point trajectories from a single scene observation, and (2) learning a closed-loop policy conditioned on these predicted point tracks.

10.3.1 Problem Formulation

Stage 1: 3D point track prediction. First, given an initial point cloud $P_0 \in \mathbb{R}^{N\times 3}$ of N visible scene points expressed in the camera coordinate frame with all embodiment points removed, the goal is to predict 3D point tracks: the future 3D positions of each initial point over a fixed horizon of T timesteps. Formally, we learn $f_\theta : P_0 \mapsto X$, where $X = \{X_t^i \mid i \in [0, N), t \in [0, T)\} \in \mathbb{R}^{N\times T\times 3}$, with each $X^i \in \mathbb{R}^{T\times 3}$ corresponding to a prediction for the i-th point in P_0 over T time steps.

Stage 2: Track-conditioned policy learning. Given the predicted 3D point tracks X together with a current observation $o_t = (P_t, q_t)$, where $P_t \in \mathbb{R}^{N\times 3}$ is the visible scene point cloud at timestep t (with embodiment points removed) and q_t is the robot configuration, we train a closed-loop behavior-cloning policy that predicts a short chunk of H actions. Formally, we learn $\pi_\phi : (X, o_t) \mapsto A$, where $A \in \mathbb{R}^{H\times D_A}$ is a chunk of H actions of D_A dimensions each. The point tracks X are produced once from the initial observation P_0 and reused throughout the episode, while o_t provides the closed-loop signal: after executing the first $H_{\text{exec}} \leq H$ predicted actions the policy is queried again on the updated o_t.

10.3.2 Network architecture and training

3D point track prediction. We embed visible non-embodiment points with an MLP and pass the resulting tokens through a transformer with standard self-attention. A linear prediction head then outputs each point's 3D trajectory over horizon T, yielding point tracks $X \in \mathbb{R}^{N\times T\times 3}$.

During training, we subsample a fixed number of random input points from the initial point cloud, and apply per-sample unit-sphere normalization. We assume access to visibility values $m_t^i \in \{0, 1\}$, indicating whether point i is visible at timestep t. We mask losses by m_t^i, and use an L1 loss on 3D position error for all visible point-timestep pairs.

Track-conditioned policy learning. During policy training, we freeze the 3D point-track predictor. Given the first frame of a task, we generate point-track predictions, and then extract compact yet informative point-track tokens with a *Perceiver-IO* architecture [131]: each predicted point trajectory X^i is first embedded into a 3D point-track token $F^i \in \mathbb{R}^{D_F}$, giving $F \in \mathbb{R}^{N\times D_F}$, and a small set of M learnable query tokens $Q \in \mathbb{R}^{M\times D_F}$ attends to F via cross-attention.

Because the policy is closed-loop, we condition it on the current point cloud P_t and current robot configuration q_t in addition to the open-loop track tokens. We use a PointNet-style encoder following DP3 [386] to encode P_t, and a small MLP to encode q_t. The point-track, point-cloud, and configuration features are concatenated and projected into a global conditioning vector that drives a flow-matching action head built on the conditional 1D U-Net architecture of Chi et al. [45].

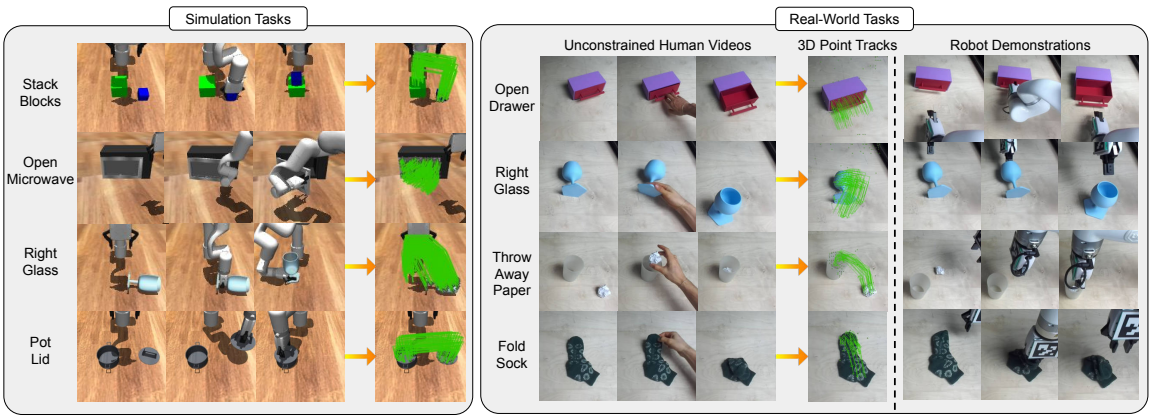


Figure 10.3: Simulation and real-world tasks used for evaluation. Simulation tasks use procedurally generated robot trajectories for both video pre-training and policy learning; real-world tasks use unconstrained human videos for video pre-training and robot teleoperation data for policy learning. Green tracks show 2D projections of the 3D point-track prediction targets (only tracks for moving points are visualized).

To bias the action head to utilize the predicted object motion rather than relying on only the raw point-cloud observation, we add *residual point-track-token cross-attention* adapters throughout the U-Net in addition to the standard FiLM-style [234] global conditioning vector. At each residual block in the down path, middle blocks, and up path, the point track tokens are linearly projected to the current U-Net channel width, and then the U-Net features query the point track tokens. The resulting cross-attention output is added back to the U-Net features. The cross-attention output is zero-initialized, so the U-Net starts with the same function as the globally conditioned model. The action head generates action chunks composed of end-effector position, orientation, and gripper position. Actions are expressed in the camera coordinate frame. We train one policy for all simulation tasks and one policy for all real-world tasks. Section 10.7 has additional architectural details.

10.4 Experiments

We evaluate 3PoinTr in simulation and on a real robot, for both 3D point track prediction and track-conditioned policy learning.

Simulation Setup In simulation, we collect demonstrations using a UFactory xArm 7 with a UFactory parallel-jaw gripper. We procedurally generate demonstrations by executing end-effector waypoints derived heuristically from randomly generated initial configurations. We evaluate on four simulation tasks (Figure 10.3), varying the position and orientation of objects across trials (Figure 10.5).

Real-World Setup For real-world robot demonstrations and evaluation, we use an Xbox controller to teleoperate the same UFactory xArm 7 and parallel-jaw gripper as in simulation. For unconstrained video demonstration collection, a human performs the manipulation tasks. We calibrate a ZED Mini camera to the scene using eye-to-hand calibration, and keep the camera static.

We evaluate on four real-world tasks (Figure 10.3), varying the position and orientation of objects across trials. Although human and robot demonstrations share similar initial and goal states, the robot gripper and human hand do not need to have the same orientations and trajectories throughout the task. Instead of trying to mimic the intended downstream robot motion, the human demonstrator performs the tasks in a manner they see fit. For example, for the *Right Glass* task, the human grasps the glass by the stem, which leads to singular configurations if the robot attempts to mimic—instead, the robot inserts its gripper into the glass and lifts the rim such that the glass pivots about its base. Although the general strategies for the other tasks are more similar between human and robot, the angles of approach, grasp points, and trajectories differ between the human and robot embodiments. Figure 10.6 (Section 10.9) shows examples of these embodiment-induced trajectory differences.

10.4.1 Implementation Details

Data collection In simulation, we collect videos of robot demonstrations, while in real-world experiments, we collect unconstrained human demonstration videos. We extract 3D point trajectories for a dense grid of points initialized at the first image frame of each demonstration. The real-world data additionally includes per-frame visibility masks. Section 10.10 describes the full process. We use coarse trajectories of $\sim$10–16 timesteps as the training targets for the 3D point-track predictor, and finer trajectories of $\sim$30–50 timesteps as the targets for training the closed-loop policy. In real-world experiments, we train point track prediction networks solely on human videos, as we find that this provides consistent conditioning for the policies.

Point track prediction network We use a 3-layer MLP with output dimension 256 and GELU activations to tokenize input points. We use two decoder blocks with 4 attention heads with head dimension 64. In between the blocks we use a feed-forward network with expansion ratio 4 and GELU activation. After the decoder, a linear head predicts per-point 3D trajectories over the full prediction horizon. At both training and test times, we randomly subsample 2048 points from the initial point cloud. We add Gaussian noise with $\sigma = 0.01$ to input point clouds as data augmentation.

Track-conditioned policy network The 2048 per-point trajectory tokens are projected to 128-D, and 4 learned Perceiver-IO queries cross-attend to them to produce 4 point track tokens. These point track tokens are then used for both global conditioning and residual cross-attention in the U-Net. The global conditioning vector is formed by concatenating three features: (i) the flattened point-track tokens, (ii) a 64-D PointNet feature of the current point cloud, and (iii) a 32-D feature of the current robot configuration, followed by a projection to 128-D.

The flow-matching action head predicts an action chunk $A \in \mathbb{R}^{H \times D_A}$ with $D_A = 10$, where H is the prediction horizon. Each action contains a 3-D end-effector position, a 6-D continuous orientation representation [402], and a 1-D gripper command.

10.4.2 Baselines

3D Point Track Prediction Baseline We compare to General Flow [382], a state-of-the-art 3D flow prediction method. We retrain the General Flow network from scratch on our dataset, which excludes all embodiment points.

Policy Learning Baselines We compare to two behavior cloning baselines that do not use any pre-training, and two point-track-based methods that use track-prediction networks pre-trained on actionless videos:

- *Diffusion Policy (DP)* [45]: behavior cloning with a diffusion action head conditioned on RGB images and robot poses.
- *3D Diffusion Policy (DP3)* [386]: Diffusion Policy conditioned on 3D point clouds instead of images.
- *ATM* [350]: predicts future 2D point tracks and uses them for policy learning. Following ATM, we train its track predictor on both robot and human videos for real-world tasks.
- *AMPLIFY* [49]: learns video dynamics in a compressed latent space, then learns inverse dynamics from limited action-labeled demonstrations.

We evaluate all baselines in simulation. We select the best performing baseline from each class (DP3 for behavior cloning, and ATM for flow-based video pre-training) for evaluation on real-world tasks.

Ablations We also ablate key design decisions on the simulation task set, using 20 action-labeled robot demonstrations and 100 actionless videos:

- *No Perceiver-IO*: replaces the Perceiver-IO encoder with mean pooling over per-point tokens.
- *No U-Net Xattn*: removes residual point-track-token cross-attention.
- *2D Point Tracks*: predicts and conditions on 2D instead of 3D point tracks, and uses the 2D image encoder from DP instead of the 3D point cloud encoder from DP3.
- *No Extra Videos*: trains the point-track predictor only on the 20 robot demonstrations, without any additional video data.

Table 10.1: Average Displacement Error (ADE) and ADE of the 5 % of points that move the most (5% ADE) for 3PoinTr and General Flow [382] on real-world tasks. Units are millimeters.

Task	**General Flow**		**3PoinTr**	
	ADE↓	5% ADE↓	ADE↓	5% ADE↓
Open Drawer	2.54	16.37	**2.18**	**13.19**
Right Glass	3.34	30.38	**2.29**	**18.55**
Throw Away Paper	2.40	22.87	**1.47**	**8.17**
Fold Sock	1.56	10.09	**1.14**	**4.68**

10.4.3 Results: 3D Point Track Prediction

We evaluate 3PoinTr's 3D point-track predictions on all simulation and real-world tasks using 3D Average Displacement Error (ADE [382]) in millimeters. We report total ADE over all points and ADE over the 5 % of points that move the most (5 % ADE). We propose the 5 % ADE metric to reflect task-critical motions, as total ADE is dominated by static background points. Results are computed over 100 unseen simulation samples and 20 unseen real-world samples. Although each method predicts full trajectories for every input point, errors are computed only for visible ground-truth point-timestep pairs (i, t).

Table 10.1 shows 3PoinTr outperforming General Flow in both metrics on every real-world task, with an average error reduction of **28 %** and **44 %**. The primary advantage of 3PoinTr is that it trains on data General Flow ignores. Real-world points are often temporarily occluded; General Flow removes any trajectory with invisible point-timestep pairs during preprocessing, whereas 3PoinTr retains all trajectories and masks losses for individual invisible point-timestep pairs. This provides additional supervision over task-critical object points that are temporarily occluded during manipulation. For example, in the *Throw Away Paper* task, every paper point is invisible by the final frame, so General Flow receives no supervision over paper motion. As a result, 3PoinTr generates more informative point tracks from realistic manipulation videos. We include point track prediction results for simulation tasks (where supervision is equal) in Section 10.11; we find that 3PoinTr achieves competitive results with a significantly simpler architecture.

10.4.4 Results: Sample-Efficient Policy Learning

Next, we evaluate the performance of 3PoinTr's policies, which condition on 3D point track predictions. We display success rates across simulated and real tasks in Tables 10.2 and 10.3. We evaluate policies trained on all real-world tasks with 20 robot demonstrations and 50 human videos per task, and separately on all simulation tasks with 20, 50, and 100 action-labeled demonstrations and 100 actionless videos per task. 3PoinTr achieves the highest success rate on every task with 20 robot demonstrations, and the highest average

Table 10.2: Simulation success rates (%) evaluated over 200 rollouts per task. Results are reported for policies trained with 20, 50, and 100 action-labeled demonstrations and 100 actionless videos per task.

Task	**AMPLIFY** [49]			**DP** [45]			**ATM** [350]			**DP3** [386]			**3PoinTr**		
# Demonstrations	20	50	100	20	50	100	20	50	100	20	50	100	20	50	100
Block Stack	0.5	3.5	2.0	17.0	30.5	60.5	11.0	47.5	67.0	25.5	41.5	57.0	**66.0**	**84.5**	**94.0**
Right Glass	2.0	6.0	9.0	16.5	27.5	76.5	8.0	25.0	32.5	21.0	43.0	70.0	**51.0**	**77.0**	**97.0**
Pot Lid	3.0	4.0	7.0	42.5	51.0	58.0	27.5	51.0	54.5	51.5	60.0	77.5	**67.5**	**91.0**	**93.0**
Open Microwave	7.0	2.5	8.0	17.5	35.0	81.5	70.0	**93.5**	**99.0**	43.0	48.0	56.0	**75.0**	78.0	89.0

Table 10.3: Real-world success rates for policies trained with 20 robot demonstrations and 50 human videos per task. 3PoinTr achieves the highest success rate on all four tasks, with a 25.0 percentage point higher average success rate than the best baseline, DP3.

Task	**ATM [350]**	**DP3 [386]**	**3PoinTr**
Open Drawer	6/20	14/20	**20/20**
Right Glass	3/20	18/20	**20/20**
Throw Away Paper	0/20	9/20	**18/20**
Fold Sock	7/20	14/20	**17/20**

success rate with 50 and 100 demonstrations.

Outperforming the behavior cloning baselines suggests that 3PoinTr is able to extract useful priors from the 3D point-track predictions. Outperforming ATM and AMPLIFY suggests that encoding embodiment-specific points is not necessary for learning from videos. In fact, because the robot points occupy a large portion of the scene, one explanation is that ATM and AMPLIFY become reliant on robot point track predictions, and end up encoding less information about the underlying scene and task. Removing these biases becomes additionally advantageous in real-world experiments: compared to 3PoinTr, ATM performance suffers from the distribution shift between human videos and robot demonstrations. ATM's per-step track re-prediction is also vulnerable to occlusions: temporarily occluded points are unrepresented in track predictions. For example, in the *Throw Away Paper* task, where ATM obtains a 0 % success rate, the gripper occludes nearly the entire object after grasping. 3PoinTr predicts tracks once from the initial scene, so points that become temporarily occluded during manipulation remain present in the conditioning. DP3 also struggles with this task, likely because the paper is small and close to the table, making it difficult to distinguish in the raw point cloud. In contrast, point-track predictions clearly highlight the paper's motion and location, and 3PoinTr achieves double DP3's success rate.

Overall, 3PoinTr's conditioning representation and policy learning framework achieve substantially stronger performance across our evaluated tasks while making fewer assumptions about visual and kinematic alignment between actionless videos and robot demonstrations. By avoiding reliance on embodiment point tracks, 3PoinTr learns a task representation

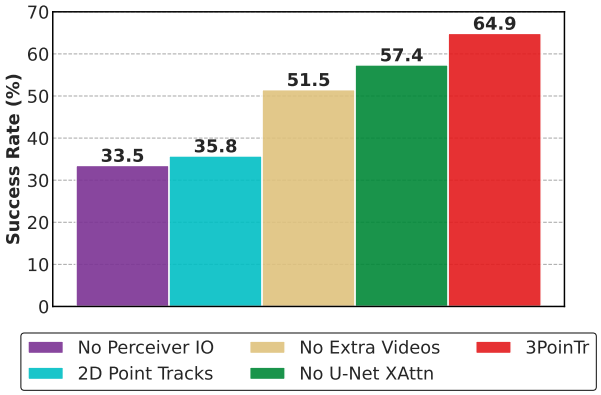


Figure 10.4: Ablation average success rates over simulation tasks with 20 robot demonstrations.

that transfers robustly from human videos to robot execution.

10.4.5 Results: Ablations

3PoinTr achieves the highest average success rate across ablations, as shown in Figure 10.4. Full results are shown in Section 10.12. We find the Perceiver-IO encoder and the use of 3D point tracks to be particularly influential. *No Extra Videos* does surprisingly well, showing that learning to predict 3D point tracks is a useful inductive bias in addition to enabling learning from actionless videos; however, 3PoinTr still achieves the highest average success rate, showing that 3PoinTr is able to utilize actionless data to further improve performance. We also find that removing residual point-track-token cross-attention in the U-Net (*No U-Net Xattn*) reduces performance across all tasks.

10.5 Conclusion

We introduced 3PoinTr, a framework for learning manipulation policies from unconstrained human videos by pre-training dense 3D point track priors with embodiment points removed and transferring them to behavior cloning with a small number of action-labeled robot demonstrations. Across real-world tasks, 3PoinTr produces higher-quality 3D point tracks than a strong baseline, and these high-quality tracks translate into robust policies that outperform several robot learning baselines. We show that 3PoinTr's 3D point track representation is an effective task specification: it encodes the scene dynamics and goal state in metric space, remains robust to appearance variation, and avoids brittle assumptions about matching human and robot kinematics. By avoiding cross-embodiment retargeting, 3PoinTr can learn from unconstrained human videos that encode useful manipulation priors without providing one-to-one motion guidance for the robot.

By contributing an approach to pre-train rich 3D point track priors from unconstrained videos, 3PoinTr takes a step towards future work in making effective use of internet-scale, in-the-wild human interaction data for learning robot policies.

10.6 Limitations

While 3PoinTr enables learning from unconstrained videos, where the human does not need to mimic the robot, our current experiments still rely on videos produced under relatively consistent laboratory conditions to achieve tractable sample efficiency. Scaling both pre-training and behavior cloning data remains an important direction for harnessing large-scale internet video data. We further discuss limitations and future work in Section 10.13.

10.7 Implementation Details

Perceiver-IO point-track encoder. The frozen point track prediction model outputs $X \in \mathbb{R}^{N\times T\times 3}$. Following [131], we compress X with a small set of M learnable queries that cross-attend to the per-point tokens. Each predicted trajectory $X^i \in \mathbb{R}^{T\times 3}$ is embedded by a shared MLP into a track token $F^i \in \mathbb{R}^{D_F}$, giving $F \in \mathbb{R}^{N\times D_F}$. A small stack of decoder blocks then updates the latents through self-attention, cross-attention to F, and a feed-forward network, producing point track tokens $Z \in \mathbb{R}^{M\times D_F}$. We use Z in two complementary ways, both described below.

Global FiLM conditioning vector. The current point cloud P_t is encoded by a DP3-style PointNet [386] into $f_{\mathrm{pc}} \in \mathbb{R}^{D_{\mathrm{pc}}}$, and the current robot configuration q_t (end-effector position, 6D continuous rotation [402], and gripper value) is encoded by an MLP into $f_{\mathrm{conf}} \in \mathbb{R}^{D_{\mathrm{conf}}}$. These are concatenated with $f_{\mathrm{track}} \in \mathbb{R}^{D_{\mathrm{track}}}$, the point track tokens Z flattened and linearly projected, and projected by a final shallow MLP ψ to the global conditioning vector

$$c = \psi(f_{\mathrm{track}}, f_{\mathrm{pc}}, f_{\mathrm{conf}}) \in \mathbb{R}^{D_c}. \tag{10.1}$$

Flow-matching action head. The action head uses conditional flow-matching and outputs a chunk of H future actions $A \in \mathbb{R}^{H\times D_A}$, with $D_A = 10$ per step (end-effector position, 6D rotation, and gripper command), expressed in the camera frame used for data collection and track prediction. Its inputs are a noisy action chunk $A_\tau \in \mathbb{R}^{H\times D_A}$, a flow time $\tau \in [0, 1]$, the global conditioning vector c, and the point track tokens Z. The backbone is the conditional 1D U-Net of Chi et al. [45]: τ is encoded with a sinusoidal embedding, concatenated with c, and used to modulate every residual block via feature-wise linear modulation (FiLM).

Residual point-track-token cross-attention adapters. In addition to the global FiLM conditioning, we insert a residual cross-attention adapter into every residual block of the U-Net, each receiving the same point track tokens Z. The U-Net activations at each block (one feature vector per horizon position) serve as queries, while the point track tokens Z serve as keys and values. Each horizon position can therefore attend to the subset of motion tokens most relevant to its own action step, and the result is added back into the block as a residual update. Each adapter's output projection is zero-initialized, so at the start of training the policy reduces to the standard FiLM-only U-Net.

Normalization and Loss Weighting. Input point clouds and action targets are jointly normalized per sample. We center each sample by the mean of its initial point cloud and scale by the 90th-percentile distance of points from that center. We apply the same transform to the current point cloud, end-effector position, and positional action targets. Orientation targets remain in their raw 6-D representation, and gripper values are divided by a fixed scale factor.

For multitask training, we use task-balanced sampling and per-task loss reweighting. The sampler draws examples uniformly across tasks. After one warmup epoch, we compute the mean unweighted training loss for each task and weight each subsequent sample by the inverse of its task's baseline loss. This prevents tasks with larger raw losses or more samples from dominating the shared policy update.

10.8 Visualization of Task Spatial Variation

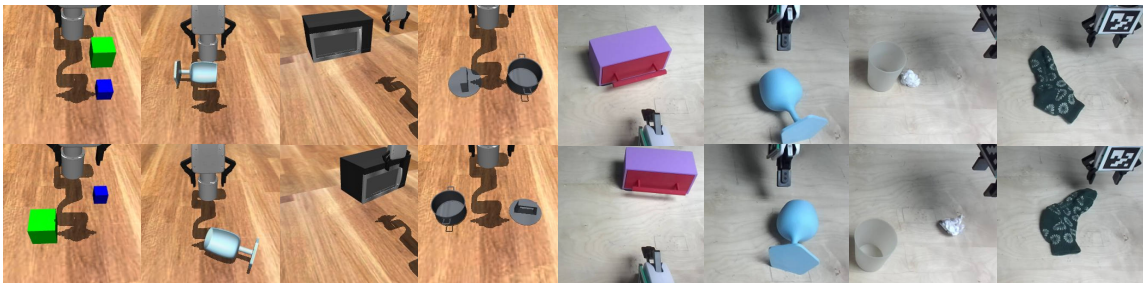


Figure 10.5: Visualization of the spatial variation used in our experiments.

10.9 Distribution Shift Between Human Videos and Robot Demonstrations

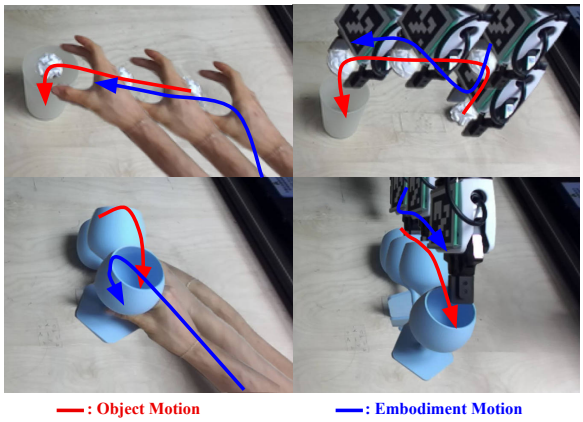


Figure 10.6: Human vs. robot trajectories: 3PoinTr learns object motion from unconstrained videos (left) and maps it to robot actions (right); embodiment and object motions need not match.

We visualize examples of the distribution shift between human and robot demonstrations in Figure 10.6. Even though object and embodiment motion differ between the human videos and the robot demonstrations, 3PoinTr is able to effectively extract useful priors from unconstrained human videos.

10.10 Data Collection Details

Our data collection procedure for training our point track prediction model and policy is as follows.

In *simulation*, we have access to ground-truth labels for 3D point track prediction training. We obtain these labels by tracking the poses of all rigid bodies throughout each trial, and applying the corresponding transformations to each point in the initial frame. Robot points are excluded.

For *real-world* videos, several steps are required to extract 3D point tracks. First, we obtain 2D point trajectories and visibility masks using a 2D point tracking model [140]. Then, we use a stereo-to-depth model [349] to obtain depth maps, and lift the 2D tracks to 3D using depth. We also segment embodiment points with SAM3 [34] in video mode using the text prompt "human arm," and mark all such points as invisible.

For both simulation and real-world datasets, we then have a set of 3D point trajectories excluding embodiment points, where the real-world data additionally includes per-frame visibility masks. We then reparameterize each trajectory by normalized time and resample it at uniform intervals, resulting in fixed-length discretized 3D point trajectories. We use

Table 10.4: Average Displacement Error (ADE) and ADE of the 5 % of points that move the most (5% ADE) for 3PoinTr and General Flow [382] on simulation and real-world tasks. Units are millimeters.

Simulation Tasks					Real-World Tasks				
Task	**General Flow**		**3PoinTr**		**Task**	**General Flow**		**3PoinTr**	
	ADE↓	5% ADE↓	ADE↓	5% ADE↓		ADE↓	5% ADE↓	ADE↓	5% ADE↓
Block Stack	**0.47**	**2.56**	0.85	5.98	Open Drawer	2.54	16.37	**2.18**	**13.19**
Open Microwave	**0.68**	**6.04**	1.79	6.79	Right Glass	3.34	30.38	**2.29**	**18.55**
Right Glass	1.96	28.82	**1.66**	**15.86**	Throw Away Paper	2.40	22.87	**1.47**	**8.17**
Pot Lid	1.18	6.89	**0.77**	**4.83**	Fold Sock	1.56	10.09	**1.14**	**4.68**

Table 10.5: Ablation simulation success rates (%) evaluated over 200 rollouts per task. Results are reported for policies trained with 20 demonstrations.

Task	**2D Point Tracks**	**No Perceiver-IO**	**No U-Net Xattn**	**No Extra Videos**	**3PoinTr**
Block Stack	35.0	24.5	56.0	**69.0**	<u>66.0</u>
Right Glass	26.5	25.0	<u>41.0</u>	19.5	**51.0**
Pot Lid	44.0	55.5	<u>62.5</u>	57.5	**67.5**
Open Microwave	37.5	29.0	<u>70.0</u>	60.0	**75.0**

coarse tracks of length ∼10–16 as the training targets for the 3D point-track predictor, and finer tracks of length ∼30–50 as the targets sampled during behavior-cloning training.

10.11 Additional Point Track Prediction Results

In simulation, all points are visible at every timestep and the two methods share the exact same training data. As shown in Table 10.4, 3PoinTr and General Flow each perform better on two of the four simulation tasks, showing that 3PoinTr's single-transformer decoder architecture is competitive with General Flow's more complicated architecture, which uses a PointNeXt [241] encoder-decoder to condition a conditional VAE.

10.12 Additional Ablation Results

Table 10.5 reports per-task simulation success rates for the ablations, using 20 demonstrations and 200 evaluation rollouts per task.

10.13 Future Work

In addition to extending to training on internet-scale, in-the-wild videos, scaling data should enable policies that generalize across camera viewpoints, environments, and other task variations, which we do not address in this chapter.

For our experiments, we find that the 3D point tracks can fully represent the task specification, so we forgo conditioning policies on additional modalities such as colored images and text prompts. However, we acknowledge that these features may be necessary for certain combinations of tasks.

3PoinTr currently subsamples a fixed number of state-action pairs for behavior cloning training spaced uniformly throughout time. Future work could explore adaptive selection strategies that allocate temporal resolution to periods of dynamic contacts or motions. 3PoinTr also relies on accurate 2D point tracking at data collection time, which can fail under certain conditions such as transparent objects, occluded objects, reflective objects, and shadows.

Chapter 11

Modality-Autoregressive World-Action Models

ORIGINALLY APPEARED AS

Adam Hung, Bardienus P. Duisterhof, Deva Ramanan, and Jeffrey Ichnowski. *Modality-Autoregressive World-Action Models [126].* arXiv:2609.17524, 2026.

Chapter in Context

Point clouds and point tracks are rich representations, but insufficient for certain tasks and can be hard to learn. What if a combination of representations is best? With ModAR, we contribute modality-autoregressive generation for world-action models. We conduct controlled experiments to study the impact of various modalities and WAM formulations. In SOLF, these comparisons test *sufficiency* and *focus* across modalities. Predicting one modality before the next improves *learnability* by letting later predictions build on earlier ones.

World-action models (WAMs) jointly model future observations and actions, typically predicting the future as RGB images. Other visual modalities such as depth, pre-trained visual features, and point tracks can more efficiently capture geometric, semantic, and motion features. However, how best to combine these modalities within WAMs remains an open question. We introduce ModAR (Figure 11.1), the first WAM to autoregressively denoise multiple future modalities before predicting actions. This allows each prediction to condition on previously generated modalities. We train from scratch to systematically study how training-data mixtures, predicted modalities, and WAM formulations affect performance. In our evaluations, WAMs benefit from predicting point tracks, DINO features, and depth maps, while additionally predicting future RGB does not provide a consistent benefit. We also find that ModAR's sequential generation outperforms existing WAM formulations, with the highest average success rate at all evaluated data scales. We also fine-tune the video-model-initialized WAM Flex-π on the same data; ModAR achieves a slightly higher observed average success rate (75% vs. 72%) while using approximately

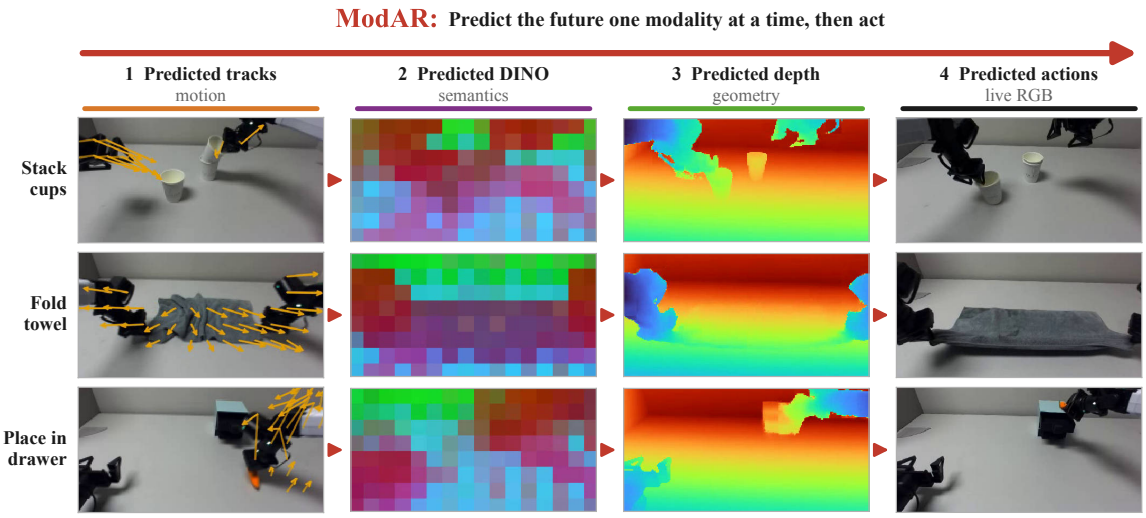


Figure 11.1: We present ModAR, a world-action model (WAM) that predicts future observations as a sequence of multiple modalities. ModAR autoregressively denoises one modality at a time before finally denoising robot actions. We train all models from scratch for controlled comparisons across predicted modalities and training data. We find that additional modalities improve performance, and our multimodal formulation outperforms prior WAM formulations. We further validate ModAR on real-world bimanual manipulation using robot demonstrations and human videos.

$20\times$ fewer training FLOPs and no pre-training. On three real-world bimanual tasks, ModAR outperforms baselines and improves with human videos.

11.1 Introduction

Future-observation prediction is a powerful objective for learning rich representations of the world's dynamics and semantics. World-action models (WAMs) harness this objective for robotics by jointly modeling future observations and corresponding robot actions.

While robot action prediction requires *action-labeled* robot data, future-observation prediction can learn from broader *actionless* data sources or initialize from pre-trained video generation models. Supervision on these broader data sources can benefit action prediction both as a training-time auxiliary objective, even when future generation is omitted at deployment [383], and by enabling policies to condition action generation on predicted visual futures [67, 163]. Recent predictive world models have explored other representations of the future that emphasize physical, spatial, or semantic structure. These include depth [168], point tracks [102, 125], and pre-trained visual features such as DINO [328, 400]. Here, we use *modality* to denote a representation of the future, including both sensory signals and derived features. These modalities each contain unique inductive biases that capture

manipulation-relevant features. In this chapter, we ask: *How should WAMs combine multiple modalities?*

Our contributions are as follows: **(1)** We introduce ModAR, a WAM that autoregressively denoises multiple future-observation modalities before generating actions. **(2)** We contribute controlled experiments on how WAM formulation, target modalities, and actionless data scale affect policy performance. We find that modality-autoregressive generation performs best across all evaluated data scales and benefits most from additional actionless data. In our experiments, predicting point tracks, DINO features, and depth provides additive gains, while additionally predicting future RGB provides no consistent benefit. We also fine-tune the 6B-parameter video-pre-trained WAM Flex-π [369] on our data. Our 30.1M-parameter ModAR achieves a slightly higher observed average success rate (75% vs. 72%) despite using approximately $20\times$ fewer training FLOPs and no pre-training. **(3)** We validate the resulting design on real-world bimanual manipulation using a mix of robot demonstrations and actionless human demonstrations.

11.2 Related work

11.2.1 World-action models

World-action models (WAMs) jointly model robot actions and future observations, enabling large-scale training on both action-labeled robot demonstrations and actionless demonstrations. Their designs vary along two primary axes: how predicted futures inform action generation and which representations of the future they predict.

WAM formulations differ in how predicted futures inform action generation (Figure 11.2). Joint-generation models like DreamZero [377] co-denoise visual futures and actions simultaneously with cross-stream attention (Figure 11.2(b)). Unified World Models [403] and Flex-π [369] do the same, but sample noise levels independently for different streams during training. Fast-WAM [383] instead prevents attention between future and action targets (Figure 11.2(c)), using future-observation prediction as a training-time auxiliary objective and omitting it at deployment. UniPi [67] and VERA [163] instead fully denoise visual futures and then infer actions from the resulting frames, rather than co-denoising both streams. Building on this futures-then-actions ordering, ModAR autoregressively denoises multiple future modalities one at a time before denoising robot actions. Our controlled experiments compare each formulation (Figure 11.2) for multimodal WAMs and we find ModAR performs best.

WAMs also differ in how they represent predicted futures. Most WAMs predict future RGB as image latents from frozen video VAEs [148, 161, 227, 377, 383, 403]. While this representation provides a convenient interface to pre-trained video generators, reconstructing visual appearance does not explicitly prioritize the geometric, physical, and semantic structures most relevant to manipulation tasks. Recent work has therefore explored predicting alternative modalities either in place of RGB [158, 400] or alongside it [102, 168, 328]. Point

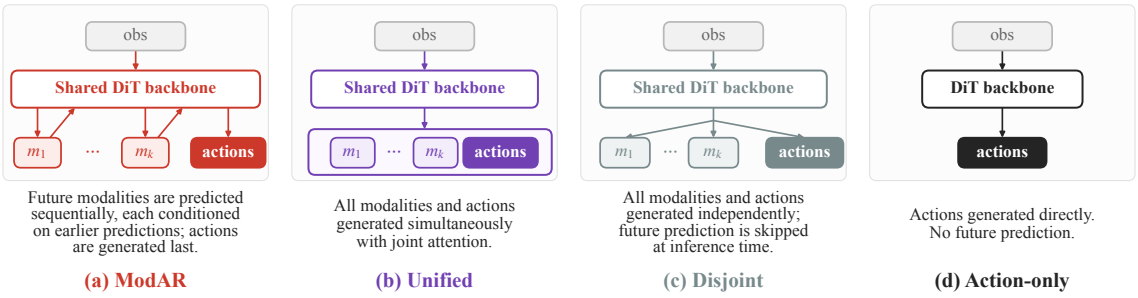


Figure 11.2: **WAM formulations.** We draw block diagrams of the evaluated WAM formulations and the Action-only baseline, which differ only in how they couple future prediction with action prediction. ModAR (a) autoregressively denoises future modalities before denoising actions. Unified (b) represents joint-generation WAMs such as DreamZero and Cosmos Policy [148, 377], which denoise future-observation and action streams together. Disjoint (c) represents Fast-WAM [383], which predicts future observations and actions independently and omits future-observation prediction at inference time. Action-only (d) denoises actions directly without predicting future observations. Independent-noise (not drawn) is identical to Unified (b) except during training: rather than applying one shared noise level to all predicted streams, it samples each stream's flow timestep independently.

tracks encode scene motion and correspondence independently of appearance, providing direct supervision for how task-relevant scene elements move through time [20, 125, 350]. DINO features are robust to appearance variation while encoding object semantics and scene structure [225]. Depth makes scene geometry explicit and provides direct supervision for the spatial reasoning required for predicting 3D actions [168]. Concurrent work Flex-π finds that jointly predicting multiple future modalities (RGB, 3D pointmaps, and DINO features) can improve performance over predicting future RGB [369].

Most existing WAMs build on pre-trained video-generation models [148, 369, 377]. This initialization is highly effective, but makes it difficult to isolate the effects of WAM formulation, target representations, and actionless-data scale. We instead train all models from scratch and systematically study these factors in a controlled setting.

11.2.2 Multimodal generation

Predicting multiple representations of the same underlying signal can provide complementary supervision for representation learning. By co-training a shared backbone across multiple objectives, each objective can enrich and regularize the representations used by the others. Prior work finds that such co-training can improve per-task performance relative to single-task training, and recent systems have scaled this idea to many vision, language,

and action tasks with strong results [9, 198, 212, 222]. Following this idea, we jointly predict multiple representations of the future so that action prediction can benefit from their inductive biases.

Beyond the choice of targets, their generation order can influence how much they help one another. Latent Forcing [8] generates image latents before pixels, allowing the latents to serve as a semantic "scratchpad" for generating fine-grained appearance. Similarly, Modality Forcing [74] adapts a pre-trained image generation model to jointly generate depth and finds that image models contain valuable priors which improve depth accuracy. These results suggest that *generating easier-to-predict modalities earlier can expose structure that simplifies subsequent predictions.* We extend this principle to multimodal WAMs by generating modalities autoregressively, starting with more structured modalities like point tracks and DINO features followed by increasingly detailed depth and RGB predictions, and finally actions.

11.3 Method

11.3.1 Problem definition

Given demonstrations spanning multiple manipulation tasks, our objective is to jointly model future multimodal observations and robot actions conditioned on the current observation, robot configuration, and task label. We consider the set of predicted future modalities $\mathcal{M} = \{\mathrm{rgb}, \mathrm{depth}, \mathrm{dino}, \mathrm{tracks}\}$. At decision time t, let $\mathbf{o}_t$ denote the multimodal observation. The complete conditioning information is $\mathbf{c}_t = (\mathbf{o}_t,\ \mathbf{q}_t,\ g)$, where $\mathbf{q}_t$ is the robot's proprioceptive configuration (joint positions and gripper opening) and g is the learned embedding associated with a discrete task label. For each $m \in \mathcal{M}$, let $\mathbf{Y}_t^m = (\mathbf{y}_{t+\Delta}^m, \ldots, \mathbf{y}_{t+J\Delta}^m)$ denote J future targets in modality m, where Δ is the dynamics stride and $H = J\Delta$ is the prediction horizon. We predict actions at every control step: $\mathbf{A}_t = (\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1})$. Thus, the final prediction target corresponds to the next decision point $t + H$, while $\mathbf{A}_t$ contains the H actions executed between t and $t + H$.

11.3.2 ModAR: Modality-Autoregressive World Modeling

Given the targets above, ModAR denoises one future modality at a time and uses each completed prediction as context for generating the next. We generate actions last, conditioning the policy on every generated modality. Formally, let $(m_1, \ldots, m_K)$ be an ordering of $\mathcal{M}$ (or a subset of $\mathcal{M}$), and write $\mathbf{Y}_t = (\mathbf{Y}_t^{m_1}, \ldots, \mathbf{Y}_t^{m_K})$ for the corresponding future targets. We model their joint distribution with actions as

$$p_\theta(\mathbf{Y}_t, \mathbf{A}_t \mid \mathbf{c}_t) = p_\theta(\mathbf{A}_t \mid \mathbf{c}_t, \mathbf{Y}_t) \prod_{k=1}^{K} p_\theta(\mathbf{Y}_t^{m_k} \mid \mathbf{c}_t, \mathbf{Y}_t^{<k}), \tag{11.1}$$

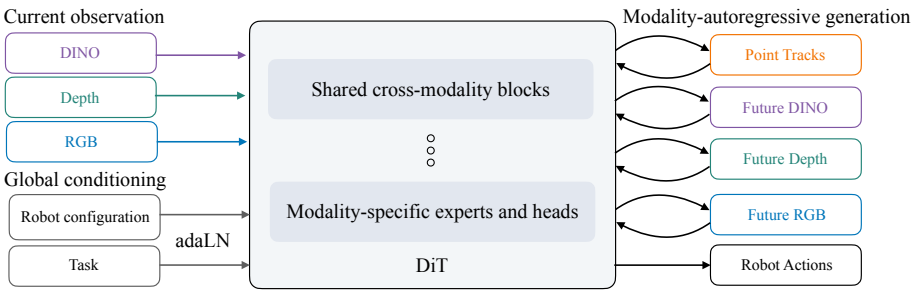


Figure 11.3: **ModAR architecture.** We embed the current observation from each modality and pass the resulting tokens into a shared diffusion transformer. The DiT combines cross-modal blocks with modality-specific experts and output heads. Robot configuration and a learned task embedding provide global adaLN conditioning to each DiT layer. ModAR sequentially denoises future modalities, conditioning each on earlier predictions, and denoises actions last.

where $\mathbf{Y}_t^{<k} = (\mathbf{Y}_t^{m_1}, \ldots, \mathbf{Y}_t^{m_{k-1}})$ denotes the modalities preceding m_k in the generation order. The final action-prediction step acts as an inverse dynamics model (IDM), mapping the generated future observations to the actions that induce the predicted transitions. For actionless examples, we omit action prediction and supervise only the available future targets.

Modality tokenization. We patchify RGB and depth maps. We extract DINO tokens from the spatial patch tokens of a frozen DINOv2 encoder [225]. For point tracks, we initialize a 2D grid of queries at patch centers and track them with CoTracker3 [140]; we represent each point-time pair with a token that encodes its displacement from its initial position and its visibility. We linearly project each modality into the common token dimension. We add a learned modality embedding and apply axial rotary position embeddings (RoPE) [282] over time and the two spatial dimensions for visual tokens, and over time for action tokens.

Shared world-action backbone. As shown in Figure 11.3, we process the resulting tokens from all modalities with a diffusion transformer (DiT) [230]. The DiT first applies several shared transformer blocks that attend across all causally available modality tokens. It then applies a small modality-specific expert stack, whose attention is restricted to that modality's stream, followed by a linear output head. The robot configuration, task embedding, and flow timesteps for each modality condition the transformer through adaLN. Thus, cross-modal information is fused in the shared blocks before within-modality specialization in the expert blocks.

Block-causal modality generation. Equation (11.1) factorizes the joint multimodal distribution into a sequence of modality-conditional distributions. Our autoregressive sequence consists of modality blocks: we generate all tokens of one modality jointly before moving to the next modality. When generating block m_k, the model attends to the current observation and to the completed blocks $m_1, \ldots, m_{k-1}$; blocks later in the sequence are not available. At inference, we therefore denoise one block, re-embed the completed prediction as context, and then denoise the next block. We use the order tracks $\rightarrow$ DINO $\rightarrow$ depth $\rightarrow$ RGB. Intuitively, this orders the targets from compact, structured representations that are easier to predict toward increasingly high-dimensional and detailed representations.

During training, we supervise every stage in one pass by creating a clean context copy and a noisy prediction copy of each target modality. A block-causal mask lets the prediction copy for m_k attend only to the observation and context copies of $m_1, \ldots, m_{k-1}$, preventing target leakage.

Injecting context noise. Modality-autoregressive generation is susceptible to cascading error, as imperfections in earlier generations become inputs to all later predictions. Following Latent Forcing [8], we mitigate this by adding noise to context blocks during training. Specifically, when predicting m_k, we independently sample $\boldsymbol{\epsilon}_j^{\text{ctx}} \sim \mathcal{N}(0, I)$ and $\tau_j^{\text{ctx}} \sim \mathcal{U}(1-\beta, 1)$ for every context block m_j in every training example, and replace its clean target with $\widetilde{\mathbf{Y}}_{t,\text{ctx}}^{m_j} = \tau_j^{\text{ctx}} \mathbf{Y}_t^{m_j} + (1-\tau_j^{\text{ctx}})\boldsymbol{\epsilon}_j^{\text{ctx}}$. We apply this noise only during training, not during inference.

Training. We train every output stream with a JiT-style x-prediction objective [164]. Empirically, we find that replacing x-prediction with velocity (v) prediction is often unstable and can cause training to diverge. Velocity prediction can struggle in high-dimensional spaces; predicting the clean sample is well-conditioned when the data lie on a low-dimensional manifold, as with images, depth, and other visual representations [74, 164]. We use the same parameterization for robot action generation [39, 375]. For each supervised modality m, we independently sample $\boldsymbol{\epsilon}^m \sim \mathcal{N}(0, I)$ and a flow timestep τ_m, and form a linear flow-matching interpolant [184]

$$\widetilde{\mathbf{Y}}_t^m = \tau_m \mathbf{Y}_t^m + (1-\tau_m)\boldsymbol{\epsilon}^m. \tag{11.2}$$

The corresponding output head directly predicts the clean target $\widehat{\mathbf{Y}}_t^m$. We minimize the loss function:

$$\mathcal{L}_m(\theta) = \mathbb{E}\left[\frac{\|\widehat{\mathbf{Y}}_t^m - \mathbf{Y}_t^m\|_2^2}{\max(1-\tau_m, \delta_m)^2}\right], \tag{11.3}$$

where δ_m stabilizes the loss near the clean endpoint. We apply the same objective to the action chunk, replacing $(\mathbf{Y}_t^m, \widehat{\mathbf{Y}}_t^m, \tau_m, \delta_m)$ with $(\mathbf{A}_t, \widehat{\mathbf{A}}_t, \tau_{\text{act}}, \delta_{\text{act}})$. The total loss is

$$\mathcal{L}(\theta) = \sum_{k=1}^{K} \mathcal{L}_{m_k} + \mathcal{L}_{\text{act}}, \tag{11.4}$$

We include the action term only for examples with action labels.

Inference. We generate one modality at a time in the order $m_1, \ldots, m_K$. We initialize each stream from isotropic Gaussian noise and integrate it from $\tau = 0$ to 1 using the velocity implied by its clean prediction, $\mathbf{v}_\theta = (\widehat{\mathbf{Y}} - \widetilde{\mathbf{Y}})/(1-\tau)$; the same equation applies to the action stream with $\mathbf{Y}$ replaced by $\mathbf{A}$. The resulting clean tokens become context for generating the next modality, and we condition the final action chunk on the complete generated future. We cache keys and values for the current observation and all previously generated modalities to avoid recomputing them at every denoising step.

11.4 Experiments

In simulation, we evaluate WAM formulations (Figure 11.2), actionless-data scaling, and predicted modality sets. We also compare with a large video-model-initialized WAM, Flex-π [369]. We then evaluate ModAR on real-world bimanual manipulation and learning from human demonstrations.

11.4.1 Experimental setup

Simulation. We evaluate on six representative RoboTwin [40] tasks: "dump bin", "pick bottles", "place bread", "put bottles", "stack bowls", and "turn switch." Each task uses $D \in \{50, 250, 1250\}$ total training demonstrations. We retain 50 action-labeled demonstrations and use the remaining $D - 50$ as actionless demonstrations. For each method and data scale, we train one multitask model for all six tasks. We evaluate all models on 50 held-out initial conditions per task.

Real world. We evaluate on three challenging tabletop tasks—stacking cups, folding a crumpled towel, and placing an object in a drawer and closing the drawer—using bimanual YAM arms (Figure 11.1). For each task, we collect 100 teleoperated robot demonstrations, 200 in-domain actionless human demonstrations, and 1,000 EgoDex demonstrations [117]. We use the corresponding EgoDex categories "stack/unstack cups," "basic fold," and "insert/remove drawer" for cup stacking, towel folding, and drawer placement, respectively. For each method and data mixture, we train one multitask model for all three real-world tasks. We evaluate each reported method and data mixture with 30 rollouts per task, with varied initial object poses.

Baselines. We compare representative WAM formulations (Figure 11.2) within one controlled implementation. All methods share the same backbone, action-labeled data, and optimization budget.

Unified (Figure 11.2(b)) jointly denoises all future-observation and action streams using one shared flow timestep, representing joint-generation WAMs such as DreamZero and Cosmos Policy [148, 377].

Disjoint (Figure 11.2(c)) predicts each target independently, without attention between future-observation and action targets, as in Fast-WAM [383].

Independent-noise samples a separate flow timestep for every stream during training, as in Unified World Models and Flex-π [369, 403]; at inference, it uses the same simultaneous denoising process as Unified (Figure 11.2(b)).

Action-only (Figure 11.2(d)) predicts actions directly from observations without future-observation prediction, representing a typical flow-matching behavior cloning policy. Because it has no future-observation prediction objective, Action-only trains only on action-labeled data.

11.4.2 Simulation results

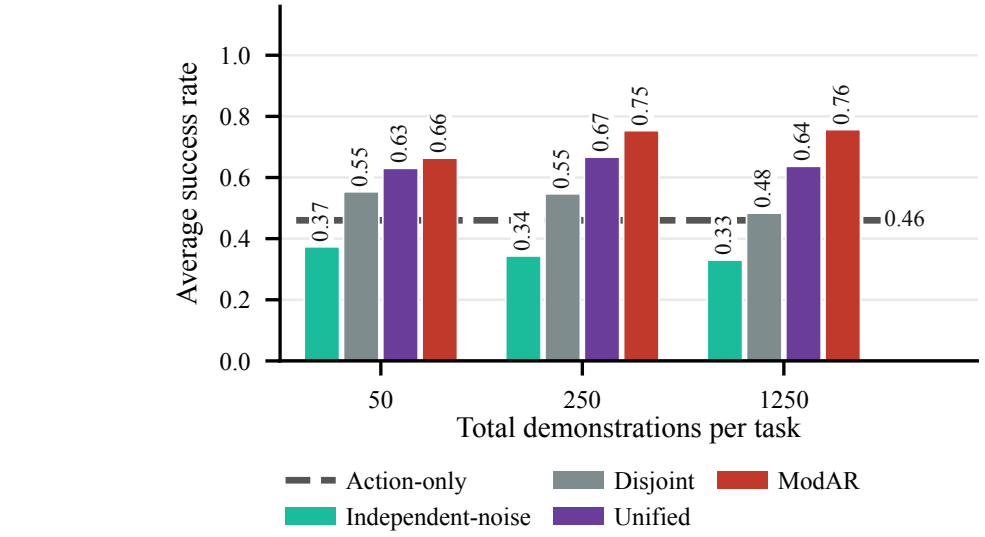
Figure 11.4 compares WAM formulations and predicted-modality subsets as the amount of actionless data varies.

Formulation comparison. Figure 11.4(a) compares formulations trained to predict all four modalities; Table 11.1 gives the per-task breakdown. ModAR achieves the highest average success rate at every data scale. We hypothesize that early modalities act as "scratchpads" for later ones: generating coarser or easier targets first provides structured context for more detailed targets [8].

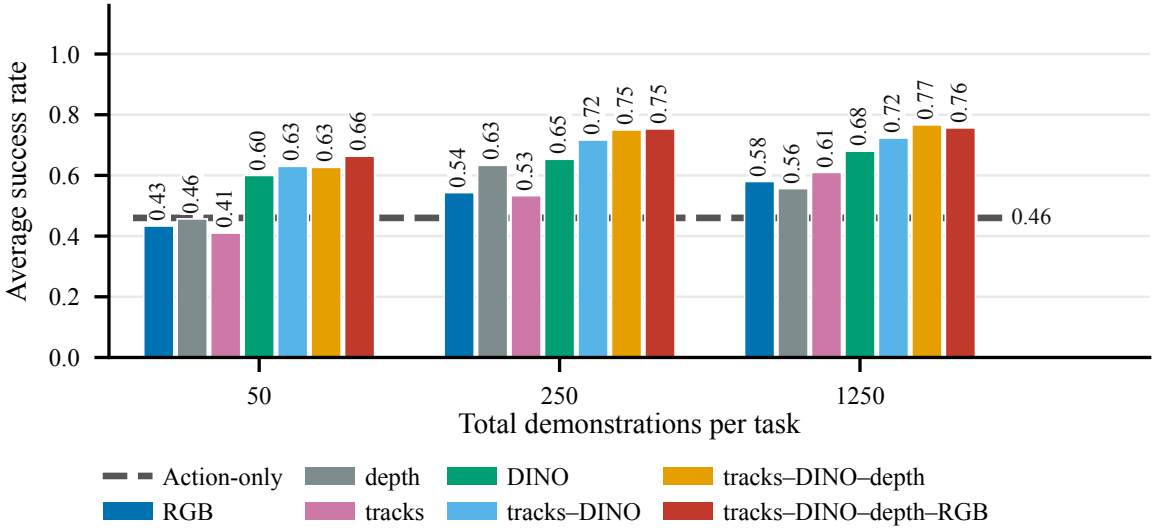
Independent-noise generation performs poorly in our from-scratch experiments. Because independently sampled noise levels rarely match the synchronized test-time denoising schedule, we hypothesize that the model receives insufficient training signal near its inference regime.

Scaling with actionless data. ModAR benefits most from additional actionless data: adding 1,200 actionless demonstrations improves the average success rate from 66% to 76% (10 percentage points), compared with a mere 1% improvement for Unified. Disjoint generation outperforms Action-only prediction on average, but its performance decreases as we add actionless data. One possible explanation is that with more actionless data, the shared representation becomes increasingly shaped by the future-observation prediction objective, causing negative transfer to the action prediction objective.

Modality comparison. Figure 11.4(b) compares ModAR variants that predict each target modality individually against variants that predict progressively larger modality sets, following the generation order point tracks, DINO, depth, and RGB; Table 11.2 gives the per-task breakdown. Across data scales, performance generally holds or increases with each added modality, showing that ModAR can effectively combine the benefits of predicting multiple modalities. However, the best modality set naturally varies across tasks because different features define each task, and some modalities represent those features better than others; as a result, additional modalities do not always help. In particular, additionally predicting future RGB on top of the first three modalities provides no consistent gain. We hypothesize



(a) Formulation comparison (Figure 11.2)



(b) Predicted modalities

Figure 11.4: **RoboTwin simulation results.** Success rates over six tasks with 50 trials per task. We train all models with 50 action-labeled demonstrations per task, then add actionless demonstrations to increase the total demonstration count from 50 to 250 to 1,250. (a) ModAR achieves the highest average success rate among the formulations at each data scale. (b) We find it beneficial to predict future tracks, DINO, and depth, while additionally predicting future RGB provides no consistent gain. RGB (dark blue) matches the typical WAM setting of predicting only future RGB and actions. Independent-noise is identical to Unified except during training: rather than applying one shared noise level to all predicted streams, it samples each stream's flow timestep independently.

Table 11.1: Per-task success rates by WAM formulation at total-demonstration scale D: action-only, independent-noise, disjoint, unified, and ModAR.

Task	D	Action-only	Independent-noise	Disjoint	Unified	ModAR
dump bin	50	0.82	0.64	0.90	0.88	**0.92**
	250	—	0.68	0.92	0.92	**0.94**
	1250	—	0.64	0.78	0.94	**0.98**
pick bottles	50	0.30	0.40	0.44	**0.58**	0.48
	250	—	0.38	0.42	**0.60**	**0.60**
	1250	—	0.38	0.38	**0.62**	0.58
place bread	50	0.24	0.20	0.46	0.44	**0.52**
	250	—	0.14	0.48	0.50	**0.74**
	1250	—	0.28	0.22	0.44	**0.72**
put bottles	50	0.32	0.20	0.36	0.52	**0.64**
	250	—	0.20	0.36	0.62	**0.82**
	1250	—	0.18	0.42	0.56	**0.82**
stack bowls	50	0.64	0.36	0.58	**0.76**	**0.76**
	250	—	0.18	0.62	**0.76**	0.68
	1250	—	0.04	0.62	0.68	**0.80**
turn switch	50	0.44	0.44	0.58	0.60	**0.66**
	250	—	0.48	0.48	0.60	**0.74**
	1250	—	0.46	0.48	0.58	**0.64**
overall	50	0.46	0.37	0.55	0.63	**0.66**
	250	—	0.34	0.55	0.67	**0.75**
	1250	—	0.33	0.48	0.64	**0.76**

that predicting future RGB introduces high-variance appearance details while adding little information beyond the more structured targets in our setting.

Separate inverse-dynamics model. Recall from Section 11.3 that ModAR's final action-prediction step acts as an inverse-dynamics model (IDM), mapping its fully generated future-observation predictions to actions. Unified instead predicts actions while its predicted futures are still being jointly denoised. Thus, ModAR could outperform Unified simply because its action predictor conditions on more informative predicted futures. To test this, we evaluate both models using the same separately trained IDM in place of their native action predictors. We train this IDM to map ground-truth future observations to actions, and apply context noise to its inputs during training. We evaluate the best-performing ModAR and Unified checkpoints on 50 held-out initial conditions per task using this separate IDM. At each prediction step, each model generates its future-observation predictions and actions normally. We then discard its native action prediction, pass its predicted futures to the separate IDM, and execute the resulting actions (Figure 11.5). ModAR still outperforms

Table 11.2: Per-task success rates by predicted-modality subset at total-demonstration scale D. Modalities comprise RGB, depth, point tracks, and DINO features.

Task	D	Action-only	RGB	Depth	Tracks	DINO	Tracks + DINO	Tracks + DINO + Depth	Tracks + DINO + Depth + RGB
dump bin	50	0.82	0.80	0.90	0.90	**0.96**	0.92	0.92	0.92
	250	—	0.94	0.92	0.92	0.94	0.94	**0.96**	0.94
	1250	—	0.98	0.88	0.90	0.96	**1.00**	0.94	0.98
pick bottles	50	0.30	0.18	0.44	0.12	0.44	**0.58**	0.48	0.48
	250	—	0.32	0.48	0.48	**0.62**	**0.62**	0.58	0.60
	1250	—	0.30	0.40	0.42	0.62	**0.70**	0.66	0.58
place bread	50	0.24	0.26	0.38	0.10	0.44	0.52	**0.54**	0.52
	250	—	0.34	0.64	0.26	0.56	0.64	**0.76**	0.74
	1250	—	0.36	0.60	0.38	0.54	0.64	**0.72**	**0.72**
put bottles	50	0.32	0.20	0.12	0.30	**0.64**	0.48	0.48	**0.64**
	250	—	0.38	0.48	0.48	0.60	0.66	0.74	**0.82**
	1250	—	0.54	0.42	0.74	0.64	0.66	**0.90**	0.82
stack bowls	50	0.64	0.58	0.54	0.74	0.66	0.70	0.72	**0.76**
	250	—	0.72	0.74	0.64	0.66	**0.86**	0.78	0.68
	1250	—	0.66	0.64	0.66	**0.84**	0.74	0.80	0.80
turn switch	50	0.44	0.58	0.36	0.30	0.46	0.58	0.62	**0.66**
	250	—	0.56	0.54	0.42	0.54	0.58	0.68	**0.74**
	1250	—	**0.64**	0.40	0.56	0.48	0.60	0.58	**0.64**
overall	50	0.46	0.43	0.46	0.41	0.60	0.63	0.63	**0.66**
	250	—	0.54	0.63	0.53	0.65	0.72	**0.75**	**0.75**
	1250	—	0.58	0.56	0.61	0.68	0.72	**0.77**	0.76

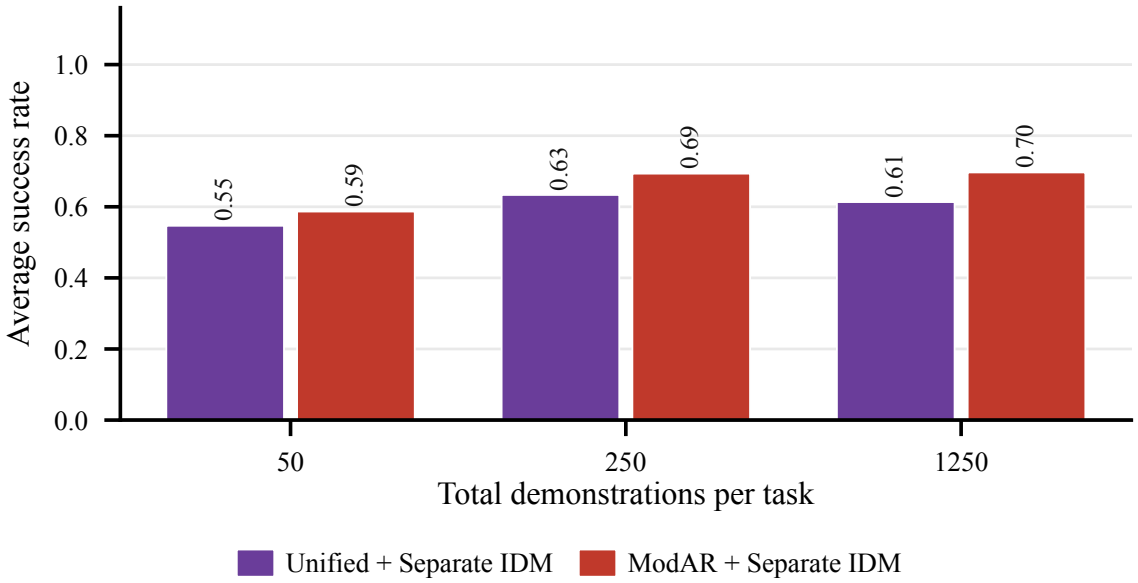


Figure 11.5: **Evaluating predicted futures with a separate IDM.** Recall from Section 11.3 that ModAR's final action-prediction step acts as an inverse-dynamics model (IDM), mapping its future-observation predictions to actions. In contrast, Unified denoises futures and actions jointly, so its action prediction conditions on partially noisy predicted futures. To remove this asymmetry, we discard both models' native action predictions and pass their predicted futures to the same separately trained IDM. ModAR still outperforms Unified, showing that its predicted futures are themselves more useful for predicting actions.

Unified with the separate IDM, suggesting that its predicted futures are themselves more useful for predicting actions.

Sampling-step comparison. ModAR generates each modality autoregressively, and performs 8 Euler integration steps per modality—40 steps across four future-observation streams and actions—whereas Unified, Independent-noise, and Disjoint use only 8 steps in total. To test whether ModAR's gains arise simply from its larger sampling budget, we re-evaluate each formulation at $D = 250$ with 40 Euler steps, matching ModAR's total number of sampling steps. Additional steps do not close the gap: Unified decreases from 67% to 59%, Disjoint from 55% to 54%, and Independent-noise increases from 34% to 37%, compared to ModAR's 75%.

Comparison to a video-model-initialized WAM. In addition to our controlled from-scratch experiments, we compare against concurrent work Flex-π [369] at $D = 250$. We adapt the official implementation to our single head-camera setting and to an action horizon of $H = 16$, with visual targets at $t + \{4, 8, 12, 16\}$. Following their recipe, we initialize the video backbone from Wan2.2-TI2V-5B [297, 320] and interpolate those weights to a smaller hidden size for the action expert, then full-fine-tune all trainable Flex-π components while keeping the VAE, text encoder, and DINOv3 encoder [274] frozen. On actionless demonstrations we retain every visual objective (RGB latents, pointmaps, and DINO features) and mask the action loss.

We train with global batch size 288 and evaluate on the same 50 held-out initial conditions per task used in our other experiments. At inference we jointly denoise all visual streams and actions, i.e., Flex-π's full multi-stream generation mode. At its final checkpoint after 30,000 steps, Flex-π attains an average success rate of **72%**. Compared to our trained-from-scratch tracks–DINO–depth ModAR variant, which achieves 75% average success, Flex-π has approximately $200\times$ as many parameters (6B vs 30.1M) and uses approximately $20\times$ as many training FLOPs. This comparison shows that, on these in-distribution tasks, a compact WAM trained from scratch can perform as well as or better than a much larger video-model-initialized WAM using substantially less training compute. We note that this is a system-level comparison rather than a controlled architectural comparison, since the models differ in scale, pre-training, target modalities, and training recipe.

Inference latency. On a single NVIDIA GeForce RTX 5090 GPU, end-to-end inference for ModAR takes 147.9 ms (6.76 Hz) when generating all four future-observation modalities and actions.

11.4.3 Ablations

Figure 11.6 ablates the ModAR design at the 250-demonstration scale, where the full model achieves a 75% average success rate.

Context noise. First, we remove the context noise added to previously generated modalities during training (*No context noise*). The success rate falls to 63%, showing that context noise during training is critical for preventing errors from compounding across successive modalities.

Modality order. We also reverse the future-modality order to RGB $\rightarrow$ depth $\rightarrow$ DINO $\rightarrow$ tracks (actions are still generated last). This lowers the average success rate to 65%, supporting our choice to generate compact, structured representations before increasingly detailed ones. We leave a full systematic comparison of modality orderings to future work.

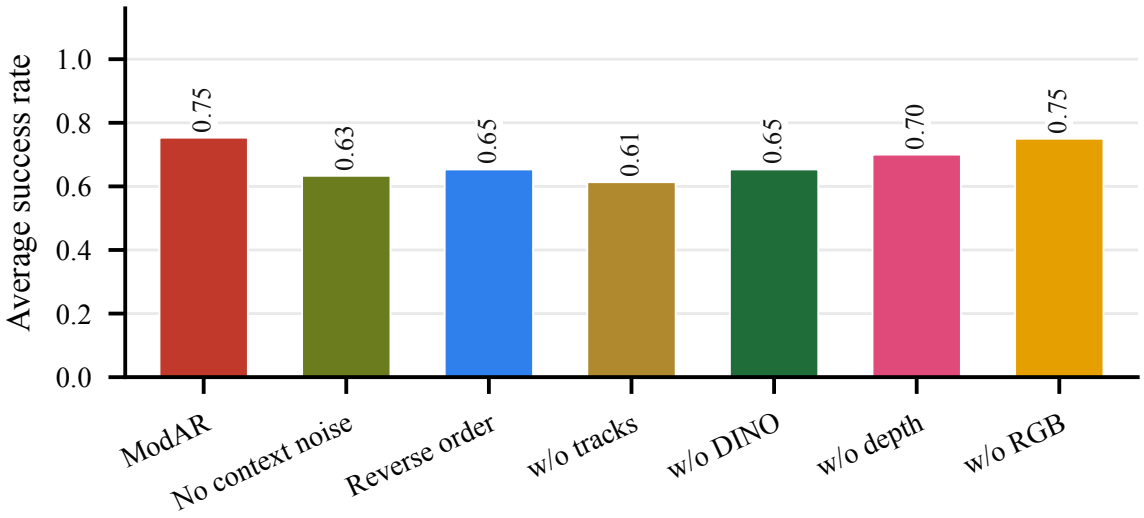


Figure 11.6: **ModAR ablations in RoboTwin.** Average success rate over six tasks, with 250 total demonstrations per task and 50 action-labeled demonstrations. We ablate context noise (*No context noise*), generation order (*Reverse order*), and each individual future-observation modality. Results suggest that context noise is critical for robust autoregressive generation. Generating modalities in reverse order also reduces performance, supporting our hypothesis that more structured modalities act as "scratchpads" for more detailed modalities. Removing future RGB prediction does not reduce performance, suggesting that RGB contributes the least of the four future-observation modalities.

Ablating modalities. Finally, we measure the contribution of each future-observation modality by removing one modality at a time from the full model. Removing tracks (*w/o tracks*), DINO (*w/o DINO*), or depth (*w/o depth*) lowers the average success rate to 61%, 65%, and 70%, respectively, whereas removing future RGB prediction leaves it unchanged. Thus, predicting RGB contributes the least of the four modalities to success rate in this setting.

11.4.4 Real-world experiments

Because adding RGB provided no consistent benefit in simulation while increasing training and generation costs, we train all real-world WAMs to observe and predict only tracks, DINO, and depth.

Formulation comparison. Results in Figure 11.7(a) mirror the simulation results: across 30 trials per task, ModAR achieves the highest success rate on all three tasks and an overall success rate of 83.3%, compared with 66.7% for Unified and 52.2% for Action-only.

Effect of adding human demonstration data. We compare ModAR trained on only the 100 teleoperated robot demonstrations per task against adding 200 in-domain human demonstrations and then also adding 1,000 out-of-domain EgoDex demonstrations. These additions improve the average success rate from 70.0% to 81.1% and 83.3%, respectively (Figure 11.7(b)), indicating that ModAR can benefit from actionless data collected across embodiments and domains.

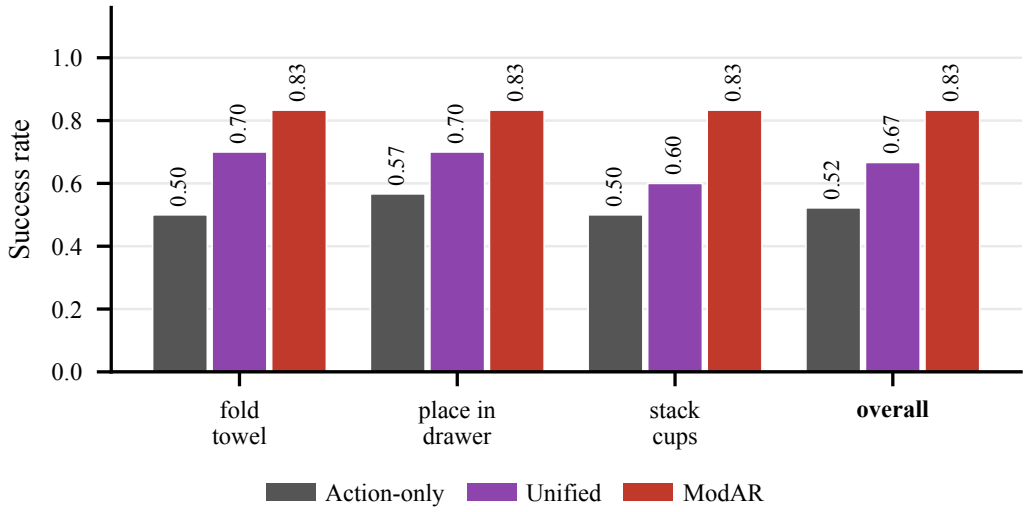
11.4.5 Implementation details

Inputs and outputs. Models condition on a single-camera 168×224 observation and predict future observations with horizon $H = 16$ and dynamics stride $\Delta = 8$, yielding $J = 2$ sparse targets at $t + 8$ and $t + 16$, together with a dense H-step action chunk; policies replan every H steps. Actions and robot configurations are 14-dimensional absolute dual-arm joint configurations. RGB, depth, and track queries use a 12×16 grid of 14×14 patches. We obtain point tracks by tracking the patch-center queries with CoTracker3 [140], and extract DINO targets with a frozen DINOv2 ViT-S/14 encoder [225].

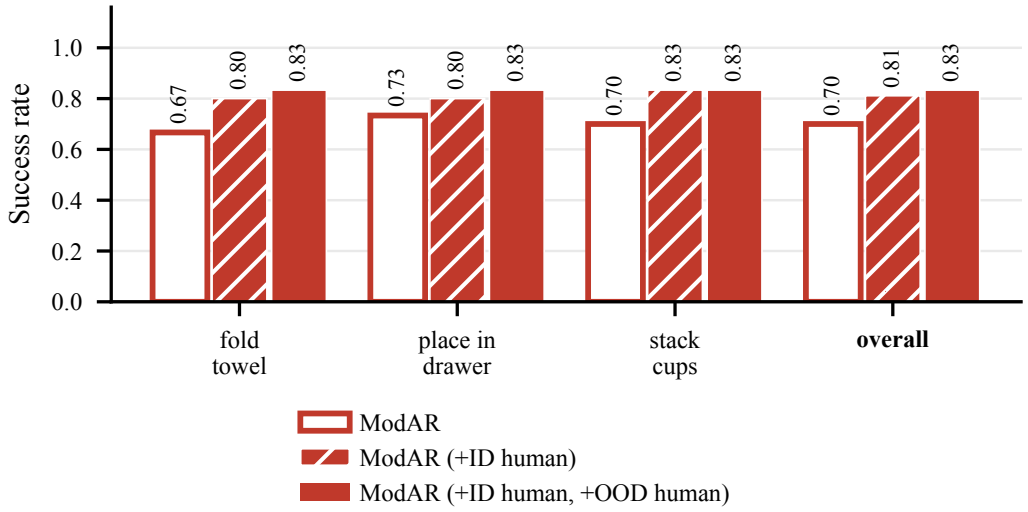
Architecture. The shared DiT contains six width-384 transformer blocks with six attention heads (ViT-S), followed by two width-384 expert blocks for DINO, depth, and RGB, one width-128 block for tracks, and two width-128 blocks for actions.

Optimization. We use AdamW with learning rate 10^{-4}, betas $(0.9, 0.95)$, weight decay 0.1, gradient clipping at 1.0, and global batch size 48. Training uses bfloat16, an EMA with decay 0.999, and a 48,000-sample linear warmup followed by a constant learning rate. We train all methods for 1.2M optimizer steps. At each optimizer step, we sample equal-sized batches from the action-labeled and actionless pools and weight their losses equally. We evaluate checkpoints every 100,000 optimizer steps and report the best-checkpoint success rate on 50 held-out initial conditions per task.

Flow and sampling. For ModAR, we set the maximum context-noise level to $\beta = 0.5$, set $\delta_m = \delta_{\text{act}} = 0.05$, use unit loss weights, and sample logit-normal flow timesteps with $(\mu, \sigma) = (-2, 1), (0, 1), (-1, 1.6), (-1, 1), (-1, 1)$ for DINO, tracks, depth, RGB, and actions, respectively. Unified uses one shared flow timestep with $(\mu, \sigma) = (-1, 1)$; Independent-noise samples each stream's modality-specific distribution independently. We integrate each generated stream with eight Euler steps.



(a) Formulation



(b) Actionless human data

Figure 11.7: **Real-world results.** Success rates on real-world tasks, with 30 trials per model per task. (a) All methods use 100 robot demonstrations per task; ModAR and Unified additionally use 200 in-domain actionless human demonstrations and 1,000 out-of-domain EgoDex demonstrations. ModAR achieves the highest success rate on all three tasks. (b) We train separate ModAR models with just 100 robot demonstrations per task, then adding 200 in-domain actionless human demonstrations (used to supervise future-observation predictions, but not action prediction), and then also adding 1,000 out-of-domain human demonstrations from EgoDex (used similarly). The average success rate progressively improves from 70.0% to 81.1% to 83.3%.

Real-world data collection. We collect robot demonstrations by teleoperation with paired teacher arms using the RAIDEN toolkit [130]. A fixed third-person ZED stereo camera provides RGB observations and stereo depth for both robot and actionless human demonstrations.

11.5 Conclusion

We introduce ModAR, a world-action model that autoregressively denoises multiple future-observation modalities before predicting actions. In a controlled simulation study, this formulation achieves the highest average success rate among representative existing WAM formulations and improves the most as we add actionless data; on three real-world tasks, it achieves the highest success rate among baselines and also improves with human video demonstrations. On average, point tracks, DINO features, and depth provide complementary gains, whereas additionally predicting future RGB provides no consistent benefit in our experiments. These findings highlight modality-autoregressive prediction as a promising alternative to relying solely on future RGB prediction for world-action modeling.

11.6 Limitations

Our experiments cover a limited number of tasks and use discrete task labels rather than language instructions, so they do not establish broad generalization across tasks, objects, or scenes. Sequentially generating multiple modalities also increases inference latency relative to simultaneous or action-only generation. Future work should explore larger-scale experiments with broader task diversity, generalization evaluation, and training on heterogeneous internet-scale actionless data. Future work can also explore the best order for generating modalities, which may depend on the task or specific scenario.

Chapter 12

Conclusion

This thesis investigates how spatiotemporal representations can focus robot learning on task-relevant geometry and dynamics. Good old-fashioned engineering (GOFE) readily enables systems to perform specialized and repeatable tasks at scale. Extending GOFE to new or non-repeatable settings, however, often requires substantial additional effort, even with learning-based systems. To support efficient learning across tasks, I develop methods for reconstructing geometry and motion and learning generative spatiotemporal priors. Furthermore, I use these priors to generate robot actions in world-action models (WAMs) to improve closed-loop robot performance.

12.1 Summary of contributions

Contributions are split into three parts: (I) reconstructing geometry and motion from observations with fewer constraints, (II) learning generative priors focused on spatiotemporal intelligence, and (III) using predictions of scene changes to generate robot actions with WAMs. Together, these contributions advance spatiotemporal representations towards the SOLF goals introduced in Chapter 1: Sufficient, Obtainable, Learnable and Focused.

12.1.1 Spatiotemporal reconstruction

Part I investigates how to recover geometry and dynamics when calibration, sparse captures, and large deformations make reconstruction difficult. These limitations restrict the *obtainability* of spatiotemporal representations. TartanCalib (Chapter 2) improves wide-angle camera calibration by using intermediate camera estimates to refine target detection and now serves researchers around the world. MASt3R-SfM (Chapter 3) explores how a learned pairwise reconstruction model can support geometry and camera estimation from arbitrary image collections, including those with little overlap or motion. Using calibrated captures, Residual-NeRF (Chapter 4) uses neural rendering for object reconstruction when we know the background in advance. Specifically, it uses a background neural radiance field (NeRF) to reduce ambiguity in the reconstruction of transparent objects and speed up convergence.

Extending to dynamic scenes, DeformGS (Chapter 5) recovers dense 3D trajectories in highly deformable scenes through a canonical Gaussian representation, a learned deformation function, and physics-inspired regularization. Finally, Cloth-Splatting (Chapter 6) extends DeformGS by combining a learned cloth dynamics model with image-based state refinement, improving estimation accuracy and convergence time.

12.1.2 Generative spatiotemporal priors

Humans reason beyond present observations to anticipate how objects will respond to interaction and infer geometry hidden from view. These capabilities may also help robots plan actions and manipulate objects from partial observations. Part II proposes methods to *generate* unseen observations across time and space, providing geometric and dynamic priors for robot manipulation.

RaySt3R (Chapter 7) infers unobserved object geometry by predicting depth maps from novel viewpoints. Its predictions transfer from synthetic training to real objects and set a new performance record for object completion. Modality Forcing (Chapter 8) examines a different source of geometric knowledge: image generation pre-training. With depth supervision held fixed, additional image-only pre-training improves depth prediction. This establishes that the data used to acquire useful geometric knowledge do not need to all contain geometric labels.

PointZero (Chapter 9) extends the study of transferable priors to dynamics. Completing 3D point tracks from a single RGB-D image and sparse future motion tracks provides a rich pre-training objective without requiring robot action labels. The learned prior transfers from simulated interactions to the completion of real-object motion and improves downstream action-conditioned dynamics and imitation learning. These results suggest that we can learn to reason about 3D dynamics without robot action labels and transfer that knowledge to improve robot learning.

Together, these chapters contribute novel, simple generative formulations for learning focused spatiotemporal representations.

12.1.3 Spatiotemporal world-action models

In the final Part III, I tighten the connection with closed-loop robot action generation. I study how to represent and generate future states and how to use these in a closed-loop policy.

Related work shows that predicting future images improves robot action generation. In 3PoinTr (Chapter 10), I instead use 3D point tracks as scene-motion plans learned from human videos. I train a closed-loop policy on robot demonstrations, using those plans as input. This representation connects desired changes in the environment to actions executable by the robot, allowing human demonstrations that need not reproduce the robot's kinematics. Its results show that the representation and policy require co-design.

ModAR (Chapter 11) investigates how complementary future modalities can contribute to action generation. Sequential, modality-autoregressive prediction gives later modalities

and actions access to earlier predictions. In the controlled simulation study, this formulation achieves the highest average success rate among the evaluated formulations on each data scale tested. Modality comparison experiments show that point tracks, depth, and DINO features improve action generation, while adding future RGB prediction does not give a consistent improvement. The choice of prediction targets and the way the model connects their predictions therefore affect the usefulness of world modeling for manipulation.

12.2 Revisiting SOLF

The SOLF framework in the Introduction organizes the representation design around four goals: Sufficiency, Obtainability, Learnability, and Focus.

Sufficiency. Geometry and motion support many of the decisions robots take: from estimating graspable surfaces to describing desired object movement. ModAR shows the value of complementary visual features for when geometry is not enough. Overall, I show that spatiotemporal representations are sufficient for many tasks and I contribute a performant recipe to add complementary visual features.

Obtainability. The reconstruction chapters make spatial and temporal supervision more obtainable under challenging conditions. Modality Forcing and PointZero reduce the dependence on particular kinds of labels during pre-training. Reliable supervision from arbitrary real-world video remains an open challenge, especially when camera motion, occlusion, and ambiguous depth complicate the recovery of 3D trajectories.

Learnability. Novel-view depth prediction, point-track completion, and modality-autoregressive generation provide learnable objectives through which models can learn useful structure for downstream manipulation. In ModAR, I further show that autoregressive prediction in a specified order is more learnable and improves action generation.

Focus. This thesis demonstrates the effectiveness of focused spatiotemporal representations in robot manipulation. Point tracks, depth and DINO are rich representations that are focused on relevant features for the diverse tasks studied. ModAR's modality study provides insight into the value of additional modalities and the trade-offs discussed in SOLF.

These findings support the goal illustrated in Figure 1.1: to retain a broadly useful physical structure while concentrating on what helps action generation. They also make clear that we must consider the four goals together. Discarding information can improve focus while impairing sufficiency, and a useful representation may still be costly or unreliable to obtain. The thesis establishes empirical advances in this trade-off.

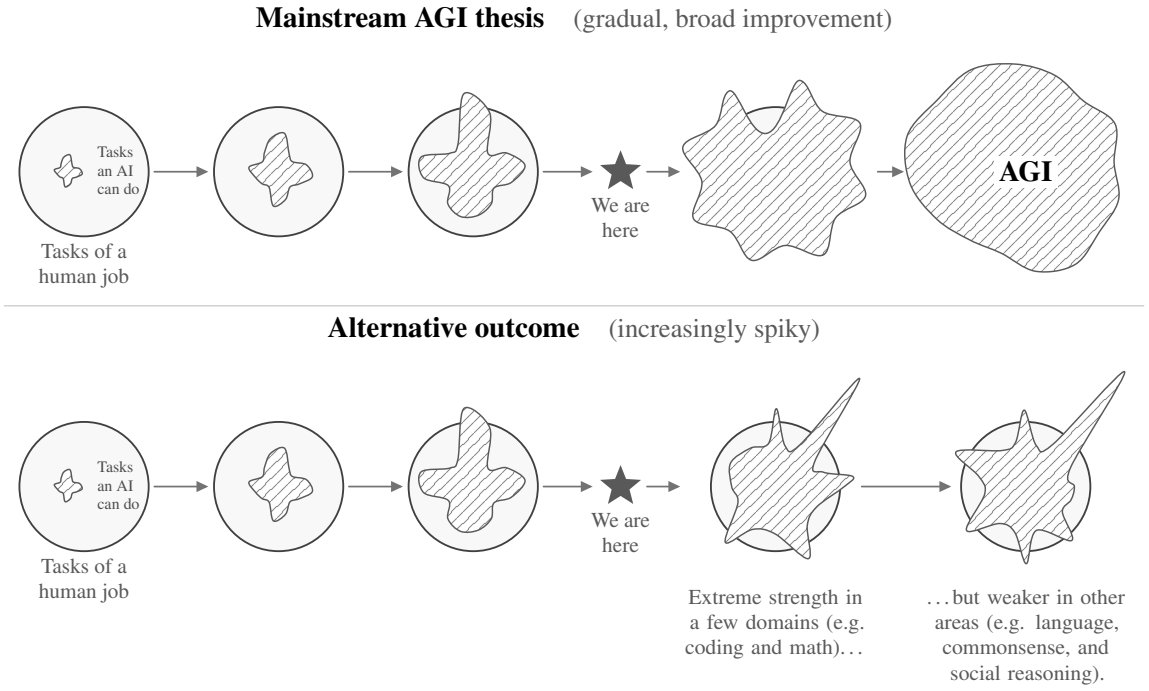


Figure 12.1: **Two trajectories of AI progress.** Circles represent the capabilities required for human jobs; hatched regions represent tasks an AI system can perform. The upper row illustrates broad improvement toward artificial general intelligence (AGI). The lower row illustrates increasingly uneven capabilities, with exceptional performance in a few domains and persistent gaps elsewhere. Adapted from an illustration by Adam Hunt.

12.3 Future work

Artificial intelligence has undergone a profound transformation in recent years. Models can generate images and 3D assets, and write code at expert levels. The breadth of these capabilities has expanded the ambitions of our field. Intelligent systems that can learn and act in the world are a natural extension of these newly acquired capabilities.

However, these achievements raise important questions about who will control future capabilities and how AI will serve society. Figure 12.1 portrays two possible trajectories in AI development. A popular thesis is that increasing scale will ultimately help models perform at the level of humans for *any* task. So far, we have *not seen any evidence* of this level of generalization. Instead, we are observing spectacular performance in narrow domains through tremendous scale.

Developing these specialized models requires substantial data, computation, and engineering. Organizations with vast resources will retain control over emerging capabilities, and their commercial priorities need not reflect the full range of tasks that matter to people. I believe that more people should have the ability to adapt intelligent systems, in order to provide the tools for AI to serve the people and use fewer precious resources.

This provides an important motivation for academic research: developing methods that allow more people to build useful capabilities with fewer resources. In robotics, efficient world models offer a concrete direction. World models can learn reusable knowledge of geometry and motion, focus predictions on the physical structure relevant to an interaction, and connect those predictions to actions. This thesis contributes methods toward this goal.

Concrete directions for future work. Future work may explore pushing spatiotemporal representations to be more general and efficient than those contributed in this thesis. Multimodal generation as in ModAR can predict many properties of a scene, but much of the predicted content is irrelevant to robotics. For example, a robot inserting a key into a lock needs precise information near the contact region, while coarser information may be adequate elsewhere. Predicting all regions, or even each sample, is unnecessary.

A promising direction is hierarchical world modeling. Predictions should focus on likely contacts, occluded task geometry, and the motion needed to complete the interaction. Extending this idea across modalities will allow models to select which future quantities to generate for a particular decision, instead of always predicting a fixed set.

The central difficulty is in learning what a model can omit. Relevance can change during an interaction, and a region that initially appears to be unimportant can become an obstacle or a source of failure. A useful hierarchy must preserve enough context to revise its focus.

Another promising direction is to use world models for autonomous data collection at test time. We can train a world-action model's observation-prediction component on interactions without robot action labels or task success. An open question is which interactions a robot should collect to improve action generation.

World models could also inform safer exploration by anticipating undesirable outcomes or identifying when a robot should not trust a prediction. Evaluating uncertainty, failure detection, and recovery during adaptation would help reduce unrecoverable interactions. The goal is sustained improvement in reliability and throughput with limited human intervention.

12.4 Closing perspective

This thesis develops spatiotemporal representations that focus on learning information relevant to robot manipulation while remaining general and scalable. I develop methods to reconstruct and generate geometry and dynamics, and connect predictions to robot actions.

The broader opportunity is to let robots acquire knowledge from diverse visual experiences and specialize it through limited task-specific interaction. SOLF provides a framework to pursue that opportunity while keeping representation design tied to its practical purpose: information that is sufficient to act, obtainable at useful scale, learnable with available resources, and focused on the demands of the task. Progress in focused prediction, online adaptation, and the role of generation can bring this research closer to manipulation systems that are capable and economical to deploy.

Bibliography

[1] Aditya Agarwal, Gaurav Singh, Bipasha Sen, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Scenecomplete: Open-world 3d scene completion in complex real world environments for robot manipulation, 2024. URL https://arxiv.org/abs/2410.23643. xvii, xxiv, 90, 92, 97, 98, 99, 100

[2] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. Bundle adjustment in the large. In *ECCV*, 2010. 28

[3] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 2023. 28

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016. 29

[5] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 46

[6] Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis. 134

[7] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. In *IEEE/RSJ IROS*, pages 1–8, 2022. 74, 75

[8] Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. URL https://arxiv.org/abs/2602.11401, 2026. 109, 115, 116, 165, 167, 169

[9] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal Multi-task Masked Autoencoders. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 348–367, 2022. doi: 10.1007/978-3-031-19836-6_20. 165

[10] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from Human Videos as a Versatile Representation for Robotics, April 2023. URL http://arxiv.org/abs/2304.08488. arXiv:2304.08488 [cs]. 146

[11] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *CVPR*, 2018. 26

[12] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. MAGSAC++, a fast, reliable and accurate robust estimator. In *CVPR*, 2020. 26

[13] Jonathan T. Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013. doi: 10.1109/CVPR.2013.10. 97

[14] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5460–5469, 2022. doi: 10.1109/CVPR52688.2022.00539. 26, 35, 109

[15] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *IJCV*, 101:6–21, 2010. URL `https://api.semanticscholar.org/CorpusID:1284146`. 61

[16] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006. 28

[17] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995. 61

[18] Johannes Beck and Christoph Stiller. Generalized b-spline camera model. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 2137–2142, 2018. doi: 10.1109/IVS.2018.8500466. xxiii, 12, 21

[19] Filippo Bergamasco, Luca Cosmo, Andrea Gasparetto, Andrea Albarelli, and Andrea Torsello. Parameter-free lens distortion calibration of central cameras. In *ICCV*, pages 3867–3875, 10 2017. doi: 10.1109/ICCV.2017.415. 12

[20] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision (ECCV)*, 2024. 58, 61, 144, 145, 164

[21] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. URL `https://arxiv.org/abs/2302.12288`, 2023. 113

[22] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. NOPE-NeRF: Optimising neural radiance field with no pose prior. In *CVPR*, 2023. 36

[23] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. URL `https://arxiv.org/abs/2410.02073`, 2025. 112, 113

[24] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6302–6314, 2022. 91, 97, 99

[25] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data, 2020. 61

[26] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. *Advances in Neural Information Processing Systems*, 34:1403–1414, 2021. 61, 91

[27] Eric Brachmann, Jamie Wynn, Shuai Chen, Tommaso Cavallari, Áron Monszpart, Daniyar Turmukhambetov, and Victor Adrian Prisacariu. Scene Coordinate Reconstruction: Posing of Image Collections via Incremental Learning of a Relocalizer. In *ECCV*, 2024. 26, 28, 36, 38

[28] Gary Bradski. The opencv library. *Dr.Dobb's Journal*, 25(11):120–125, 11 2000. URL `https://www.proquest.com/trade-journals/opencv-library/docview/202684726/se-2`. Copyright - Copyright Miller Freeman Inc. Nov 2000; Last updated - 2021-09-14; CODEN - DDJOEB. 12, 13

[29] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`, 2020. 104

[30] Kwon Byung-Ki, Qi Dai, Lee Hyoseok, Chong Luo, and Tae-Hyun Oh. Jointdit: Enhancing rgb-depth joint modeling with diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25261–25271, October 2025, 2025. xviii, 105, 107, 109, 112, 113, 114, 115

[31] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 2021. 26

[32] Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. In *CVPR*, pages 130–141, 2023. doi: 10.1109/CVPR52729.2023.00021. 60, 63, 64

[33] Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 3d shape regression for real-time facial animation. *ACM Transactions on Graphics (TOG)*, 32:1–10, 2013. URL `https://api.semanticscholar.org/CorpusID:2818777`. 61

[34] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719. 157

[35] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 123

[36] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2025, 2025. 109, 122

[37] Hanzhi Chen, Boyang Sun, Anran Zhang, Marc Pollefeys, and Stefan Leutenegger. VidBot: Learning Generalizable 3D Actions from In-the-Wild 2D Human Videos for Zero-Shot Robotic Manipulation, March 2025. URL http://arxiv.org/abs/2503.07135. arXiv:2503.07135 [cs]. 146

[38] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. ASpanFormer: Detector-free image matching with adaptive span transformer. In *ECCV*, 2022. 28

[39] Jintao Chen, Peidong Jia, Qingpo Wuwu, Jiaming Liu, Mengfei Du, Chun-Kai Fan, Xiaowei Chi, Hao Chen, Chengyu Bai, Zezhong Qian, Hao Wang, Jiajun Cao, Weishi Mi, Xiaozhu Ju, Jian Tang, and Shanghang Zhang. MV-WAM: Manifold-Aware World Action Model with Value Augmentation, June 2026. arXiv:2606.21088 [cs.RO]. 167

[40] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, June 2025. arXiv:2506.18088 [cs.RO]. 168

[41] Xiaotong Chen, Huijie Zhang, Zeren Yu, Anthony Opipari, and Odest Chadwicke Jenkins. Clearpose: Large-scale transparent object dataset and benchmark. In *European Conference on Computer Vision*, 2022. 44, 47, 50

[42] Yue Chen, Xingyu Chen, Xuan Wang, Qi Zhang, Yu Guo, Ying Shan, and Fei Wang. Local-to-global registration for bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8264–8273, 2023. 46

[43] Ho Kei Cheng and Alexander G Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, pages 640–658. Springer, 2022. 68

[44] Cheng Chi and Shuran Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. In *IEEE/CVF ICCV*, pages 3324–3333, 2021. 75

[45] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. 132, 135, 146, 148, 151, 153, 155

[46] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. 46

[47] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 97, 99

[48] Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. URL `https://arxiv.org/abs/2304.09151`, 2023. 112

[49] Jeremy A. Collins, Loránd Cheng, Kunal Aneja, Albert Wilcox, Benjamin Joffe, and Animesh Garg. AMPLIFY: Actionless Motion Priors for Robot Learning from Videos, June 2025. URL `http://arxiv.org/abs/2506.14198`. arXiv:2506.14198 [cs]. 144, 146, 151, 153

[50] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017, 2017. 111, 112, 113

[51] Murtaza Dalal, Min Liu, Walter Talbott, Chen Chen, Deepak Pathak, Jian Zhang, and Ruslan Salakhutdinov. Local policies enable zero-shot long-horizon manipulation. *International Conference of Robotics and Automation*, 2025. 97

[52] Radek Danecek, Endri Dibra, A. Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment : 3d garment shape estimation from a single image. *Computer Graphics Forum (Proc. Eurographics)*, 36(2), 2017. 75

[53] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023. 126

[54] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *CVPR*, 2022. 96

[55] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934. 80

[56] K. Deng, A. Liu, J. Zhu, and D. Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12872–12881, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.01254. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01254`. 46

[57] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020. 58, 61

[58] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR*, 2018. 28, 34, 37, 40

[59] Frederic Devernay and Olivier D. Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13:14–24, 2001. 10, 12

[60] Shivin Devgon, Jeffrey Ichnowski, Ashwin Balakrishna, Harry Zhang, and Ken Goldberg. Orienting novel 3d objects using self-supervised learning of rotation transforms. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1453–1460. IEEE, 2020. 58, 61

[61] Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. Dream2Flow: Bridging Video Generation and Open-World Manipulation with 3D Object Flow, December 2025. URL `http://arxiv.org/abs/2512.24766`. arXiv:2512.24766 [cs]. 144, 146

[62] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022. 58, 61, 76, 81

[63] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv preprint arXiv:2306.08637*, 2023. 61

[64] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 91

[65] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, 2022. 96

[66] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14304–14314, 2021. doi: 10.1109/ICCV48922.2021.01406. 60

[67] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation, November 2023. URL `http://arxiv.org/abs/2302.00111`. arXiv:2302.00111 [cs]. 162, 163

[68] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion, 2024. URL `https://arxiv.org/abs/2409.19152`. 25, 31, 40, 91, 122

[69] Bardienus P Duisterhof, Yaoyu Hu, Si Heng Teng, Michael Kaess, and Sebastian Scherer. Tartancalib: Iterative wide-angle lens calibration using adaptive subpixel refinement of apriltags, 2022. 9

[70] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes, 2023. 76, 81, 82

[71] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. Deformgs: Scene flow in highly deformable scenes for deformable object manipulation, 2024. URL `https://arxiv.org/abs/2312.00583`. 46, 57, 76

[72] Bardienus P. Duisterhof, Yuemin Mao, Si Heng Teng, and Jeffrey Ichnowski. Residual-nerf: Learning residual nerfs for transparent object manipulation, 2024. URL `https://arxiv.org/abs/2405.06181`. 43

[73] Bardienus P Duisterhof, Jan Oberst, Bowen Wen, Stan Birchfield, Deva Ramanan, and Jeffrey Ichnowski. Rayst3r: Predicting novel depth maps for zero-shot object completion. *NeurIPS*, 2025. xvii, 89, 98, 100, 122

[74] Bardienus Pieter Duisterhof, Deva Ramanan, Jeffrey Ichnowski, Justin Johnson, and Keunhong Park. Modality Forcing for Scalable Spatial Generation, June 2026. arXiv:2606.13676 [cs.CV]. 103, 165, 167

[75] Bardienus Pieter Duisterhof, Kaifeng Zhang, Adam Hung, Bowen Wen, Stan Birchfield, Yunzhu Li, Deva Ramanan, and Jeffrey Ichnowski. PointZero: 3D point track completion for learning transferable 3D dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2026. 119

[76] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *CVPR*, 2023. 28, 40

[77] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *CVPR*, 2024. 28

[78] Ben Eisner, Harry Zhang, and David Held. FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects, May 2024. URL `http://arxiv.org/abs/2205.04382`. arXiv:2205.04382 [cs]. 144

[79] Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Vaishaal Shankar, Alexander Toshev, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. 94

[80] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. In *ICPR*, 1994. 31

[81] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aar6170. 76

[82] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. URL `https://arxiv.org/abs/2403.03206`, 2024. 106, 112

[83] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 91

[84] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. doi: 10.1145/3550469.3555383. 46, 63, 64

[85] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 26

[86] Wolfgang Förstner and Eberhard Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS intercommission conference on fast processing of photogrammetric data*, 1987. 15

[87] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *CVPR*, pages 12479–12488, 2023. doi: 10.1109/CVPR52729.2023.01201. 60, 64

[88] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. URL `https://arxiv.org/abs/2403.12013`, 2024. 113

[89] Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T Barron, Kyle Genova, Nithish Kannen, Sherry Ben, Yandong Li, Mandy Guo, Suhas Yogin, Yiming Gu, Huizhong Chen, Oliver Wang, Saining Xie, Howard Zhou, Kaiming He, Thomas Funkhouser, Jean-Baptiste Alayrac, and Radu Soricut. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026, 2026. 105

[90] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5692–5701, 2021. doi: 10.1109/ICCV48922.2021.00566. 60

[91] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular Dynamic View Synthesis: A Reality Check. In *NeurIPS*, pages 33768–33780, 2022. 76

[92] Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. Mesh-based gaussian splatting for real-time large-scale deformation. *arXiv preprint arXiv:2402.04796*, 2024. 76

[93] Ruiqi Gao∗, Aleksander Holynski∗, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole∗. Cat3d: Create anything in 3d with multi-view diffusion models. *Advances in Neural Information Processing Systems*, 2024. 92

[94] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14326–14335, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.01408. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.01408`. 46

[95] Irene Garcia-Camacho, Martina Lippi, Michael C Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, Carme Torras, Alessandro Marino, Guillem Alenya, et al. Benchmarking bimanual cloth manipulation. *RA-L*, 5(2):1111–1118, 2020. 83

[96] Irene Garcia-Camacho, Alberta Longhini, Michael Welle, Guillem Alenya, Danica Kragic, and Julia Borras. Standardization of cloth objects and its relevance in robotic manipulation. In *IEEE ICRA*, pages 8298–8304. IEEE, 2024. 75

[97] Marcos García, César Mendoza, Luis Pastor, and Angel Rodríguez. Optimized linear fem for modeling deformable objects. *Computer Animation and Virtual Worlds*, 17 (3-4):393–402, 2006. doi: https://doi.org/10.1002/cav.142. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.142. 122

[98] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern recognition*, 51:481–491, 2016. 18

[99] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, September 2013. ISSN 0278-3649. doi: 10.1177/0278364913491297. URL https://doi.org/10.1177/0278364913491297, 2013. 112, 113

[100] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3749–3761, June 2022. 134

[101] M.D. Grossberg and S.K. Nayar. A general imaging model and a method for finding its parameters. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 108–115 vol.2, 2001. doi: 10.1109/ICCV.2001.937611. 12

[102] Jiarui Guan, Wenshuai Zhao, Yue Pei, Ziliang Chen, Arno Solin, and Juho Kannala. Point Tracking Improves World Action Models, May 2026. arXiv:2605.23856 [cs.RO]. 162, 163

[103] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward Flow for Novel View Synthesis of Dynamic Scenes. In *ICCV*, pages 16022–16033, October 2023. 61

[104] Hyowon Ha, Michal Perdoch, Hatem Alismail, In So Kweon, and Yaser Sheikh. Deltille grids for geometric camera calibration. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5354–5362, 2017. doi: 10.1109/ICCV.2017.571. 12, 13, 18

[105] Annika Hagemann, Moritz Knorr, and Christoph Stiller. Deep geometry-aware camera self-calibration from video. In *ICCV*, 2023. 26

[106] Siddhant Haldar and Lerrel Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *arXiv preprint arXiv:2502.20391*, 2025. 144, 145, 146, 147

[107] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, pages 59–75. Springer, 2022. 61

[108] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 26

[109] Jing He, Haodong Li, Wei Yin, Yixun Liang, Leheng Li, Kaiqiang Zhou, Hongbo Zhang, Bingbing Liu, and Ying-Cong Chen. Lotus: Diffusion-based visual foundation model for high-quality dense prediction. URL `https://arxiv.org/abs/2409.18124`, 2025. 113

[110] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *CVPR*, 2022. doi: 10.1109/CVPR52688.2022.01553. 92, 123

[111] Xingyi He, Jiaming Sun, Yifan Wang, Sida Peng, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Detector-free structure from motion. In *CVPR*, 2024. 26, 28, 36, 38

[112] J. Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, 2000. doi: 10.1109/34.879788. 13

[113] Julius Hietala, David Blanco-Mulero, Gokhan Alcan, and Ville Kyrki. Learning visual feedback control for dynamic cloth folding. In *IEEE/RSJ IROS*, pages 1455–1462, 2022. 75

[114] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 92

[115] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 92

[116] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for physical sequential fabric manipulation. *Autonomous Robots*, pages 1–25, 2022. 75

[117] Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning Dexterous Manipulation from Large-Scale Egocentric Video, August 2025. URL `http://arxiv.org/abs/2505.11709`. arXiv:2505.11709 [cs]. 168

[118] Ji Hou, Angela Dai, and Matthias Nießner. Revealnet: Seeing behind objects in rgb-d scans. In *CVPR*, 2020. 91

[119] Huang Huang, Letian Fu, Michael Danielczuk, Chung Min Kim, Zachary Tam, Jeffrey Ichnowski, Anelia Angelova, Brian Ichter, and Ken Goldberg. Mechanical search on shelves with efficient stacking and destacking of objects. In *Robotics Research*, pages 205–221, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-25555-7. 90

[120] Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. *arXiv preprint arXiv:2506.23126*, 2025. 120, 122, 128, 129, 137

[121] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation, 2026. URL https://arxiv.org/abs/2601.03782. 121, 123, 144, 146

[122] Zixuan Huang, Xingyu Lin, and David Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. *RSS*, 2022. 74

[123] Zixuan Huang, Xingyu Lin, and David Held. Self-supervised cloth reconstruction via action-conditioned cloth tracking. In *IEEE ICRA*, pages 7111–7118, 2023. 74, 75, 79

[124] Martin Humenberger, Yohann Cabon, Nicolas Guerin, Julien Morat, Vincent Leroy, Jérôme Revaud, Philippe Rerole, Noé Pion, Cesar de Souza, and Gabriela Csurka. Robust image retrieval-based visual localization using Kapture. *arXiv preprint arXiv:2007.13867*, 2020. 29, 40

[125] Adam Hung, Bardienus P. Duisterhof, and Jeffrey Ichnowski. 3pointr: From human videos to robot policies with 3d point-track plans. *arXiv preprint arXiv:2603.08485*, 2026. xviii, 120, 123, 131, 132, 139, 143, 162, 164

[126] Adam Hung, Bardienus P. Duisterhof, Deva Ramanan, and Jeffrey Ichnowski. Modality-autoregressive world-action models, 2026. URL https://arxiv.org/abs/2609.17524. 123, 161

[127] Jeffrey Ichnowski∗, Yahav Avigal∗, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *Conference on Robot Learning (CoRL)*, 2020. xv, 44, 46, 47, 49, 50, 52, 53

[128] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759. 144

[129] Shun Iwase, Katherine Liu, Vitor Guizilini, Adrien Gaidon, Kris Kitani, Rares Ambrus, and Sergey Zakharov. Zero-shot multi-object scene completion, 2024. xvii, xxiv, xxv, 90, 91, 96, 97, 98, 99, 100, 101

[130] Shun Iwase, Patrick Miller, Jonathan Yao, Krishna Murthy Jatavallabhula, and Sergey Zakharov. RAIDEN: A Toolkit for Policy Learning with YAM Bimanual Robot Arms, 2026. 178

[131] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver io: A general architecture for structured inputs & outputs, 2022. URL https://arxiv.org/abs/2107.14795. xviii, 124, 125, 144, 148, 155

[132] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5846–5854, October 2021. 46

[133] Boyi Jiang, Juyong Zhang, Yang Hong, Jinhao Luo, Ligang Liu, and Hujun Bao. BCNet: Learning body and cloth shape from a single image. In *ECCV*, pages 18–35, 2020. 75

[134] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos, 2025. URL https://arxiv.org/abs/2503.17973. 122

[135] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence transformer for matching across images. In *ICCV*, 2021. 28

[136] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*, 2024. 76

[137] Jan Kallwies, Bianca Forkel, and Hans-Joachim Wuensche. Determining and improving the localization accuracy of apriltag detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8288–8294. IEEE, 2020. 18

[138] J. Kannala and S.S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006. doi: 10.1109/TPAMI.2006.153. 10, 12

[139] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023. 58, 76, 144

[140] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proc. arXiv:2410.11831*, 2024. 121, 130, 157, 166, 176

[141] Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4d: Unified feed-forward metric 4d reconstruction. *arXiv preprint arXiv:2512.10935*, 2025. 121

[142] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. *ICCVW*, 2019. xxiv, 96, 98, 99

[143] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, 2024. 106, 109, 113, 114

[144] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*. IEEE, 2026. 122

[145] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`. 26, 58, 60, 62, 63, 65, 74, 76

[146] Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 353–367. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/kerr23a.html. 44, 46, 54

[147] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *European Conference on Computer Vision (ECCV)*, 2022. 58, 60

[148] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026. xx, 163, 164, 168

[149] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. doi: 10.48550/arXiv.1412.6980. 33, 35

[150] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *IEEE/CVF CVPR*, pages 3992–4003, 2023. 68, 104

[151] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 2017. 35

[152] Akshay Krishnan, Xinchen Yan, Vincent Casser, and Abhijit Kundu. Orchid: Image latent diffusion for joint appearance and geometry generation. URL https://arxiv.org/abs/2501.13087, 2025. 107

[153] Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. Flexible layouts for fiducial tags. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1898–1903, 2019. doi: 10.1109/IROS40897.2019.8967787. 10

[154] Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025, 2025. 106, 110, 112

[155] Lubor Ladicky, Olivier Saurer, SoHyeon Jeong, Fabio Maninchedda, and Marc Pollefeys. From point clouds to mesh using regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3893–3902, 2017. 91

[156] Karel Lebeda, Jirı Matas, and Ondrej Chum. Fixing the locally optimized RANSAC. In *BMVC*, 2012. 26

[157] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MAST3R. In *ECCV*, 2024. 26, 27, 29, 30, 37, 39, 113

[158] Baoyu Li, Xinchen Yin, Mengying Lin, Yixin Zhang, and Danfei Xu. Egowam: World action models beyond pixels with in-the-wild egocentric human data. *arXiv preprint arXiv:2607.08436*, 2026. 163

[159] Hongyu Li, Lingfeng Sun, Yafei Hu, Duy Ta, Jennifer Barry, George Konidaris, and Jiahui Fu. Novaflow: Zero-shot manipulation via actionable flow from generated videos, 2025. URL https://arxiv.org/abs/2510.08568. 144, 146

[160] Jie Li, Kai Han, Peng Wang, Yu Liu, and Xia Yuan. Anisotropic convolutional networks for 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3351–3359, 2020. 91, 97, 99

[161] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. In *Proc. Robot.: Sci. Syst. (RSS)*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.016. 163

[162] Rui Li, Biao Zhang, Zhenyu Li, Federico Tombari, and Peter Wonka. Lari: Layered ray intersections for single-view 3d geometric reasoning. In *arXiv preprint arXiv:2504.18424*, 2025. xvii, xxiv, 92, 98, 99, 100

[163] Sizhe Lester Li, Evan Kim, Xingjian Bai, Tong Zhao, Tao Pang, Max Simchowitz, and Vincent Sitzmann. Turning Video Models into Generalist Robot Policies, May 2026. arXiv:2605.27817 [cs.RO]. 162, 163

[164] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. URL https://arxiv.org/abs/2511.13720, 2026. 108, 126, 128, 167

[165] Xirui Li, Charles Herrmann, Kelvin CK Chan, Yinxiao Li, Deqing Sun, and Ming-Hsuan Yang. A simple approach to unifying diffusion-based conditional generation. In *The Thirteenth International Conference on Learning Representations*, 2025, 2025. 107, 109, 113, 115

[166] 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. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tVkrbkz42vc. 122

[167] Yiming Li, Zhiding Yu, Christopher Choy, Chaowei Xiao, Jose M Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *CVPR*, 2023. 97, 99

[168] Ying Li, Xiaobao Wei, Jiajun Cao, Hao Wang, Xiaowei Chi, Chengyu Bai, Qianpu Sun, Jiajun Li, Xiaojie Zhang, Peidong Jia, Jian Tang, Sirui Han, and Shanghang Zhang. WAM4D: Fast 4D World Action Model via Spatial Register Tokens, June 2026. arXiv:2606.14048 [cs.CV]. 162, 163, 164

[169] Yinxiao Li, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *IEEE/RSJ IROS*, pages 1046–1052. IEEE, 2014. 75

[170] Yinxiao Li, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter K Allen. Folding deformable objects using predictive simulation and trajectory optimization. In *IEEE/RSJ IROS*, pages 6000–6006, 2015. 75

[171] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 122

[172] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6494–6504, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00643. URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00643. 60

[173] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural Dynamic Image-Based Rendering. In *IEEE/CVF CVPR*, pages 4273–4284, 2023. 76

[174] Zhizhong Li and Derek Hoiem. Learning without forgetting. URL https://arxiv.org/abs/1606.09282, 2017. 110

[175] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies, 2025. URL https://arxiv.org/abs/2508.00795. 146

[176] Litian Liang, Liuyu Bian, Caiwei Xiao, Jialin Zhang, Linghao Chen, Isabella Liu, Fanbo Xiang, Zhiao Huang, and Hao Su. Robo360: A 3d omnispective multi-material robotic manipulation dataset, 2023. xv, xvi, 59, 68, 71

[177] Amy Lin, Jason Y. Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose++: Recovering 6D poses from sparse-view observations. In *3DV*, 2024. 37

[178] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 46

[179] Jiehong Lin, Lihua Liu, Dekun Lu, and Kui Jia. Sam-6d: Segment anything model meets zero-shot 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27906–27916, 2024. 58, 61

[180] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *CoRL*, pages 432–448. PMLR, 2021. 80

[181] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *CoRL*, pages 256–266. PMLR, 2022. 74

[182] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *ICCV*, 2021. 28, 34, 37

[183] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *ICCV*, 2023. 40

[184] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. URL https://arxiv.org/abs/2210.02747, 2023. 108, 126, 167

[185] Thomas Lips, Victor-Louis De Gusseme, et al. Learning keypoints for robotic cloth manipulation using synthetic data. *arXiv preprint arXiv:2401.01734*, 2024. 80, 134

[186] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 60, 61

[187] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, pages 15651–15663, 2020. 46

[188] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *CVPR*, pages 9298–9309, 2023. 92

[189] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Trans. Graph.*, 32(6), November 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508406. URL https://doi.org/10.1145/2508363.2508406. 122

[190] Y. Lochman, K. Liepieshov, J. Chen, M. Perdoch, C. Zach, and J. Pritts. Babelcalib: A universal approach to calibrating central cameras. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15233–15242, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.01497. URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.01497. 10, 12

[191] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459863. URL https://doi.org/10.1145/3450626.3459863. 46

[192] Alberta Longhini, Marco Moletta, Alfredo Reichlin, Michael C Welle, David Held, Zackory Erickson, and Danica Kragic. Edo-net: Learning elastic properties of deformable objects from graph dynamics. *arXiv preprint arXiv:2209.08996*, pages 3875–3881, 2022. 74

[193] Alberta Longhini, Yufei Wang, Irene Garcia-Camacho, David Blanco-Mulero, Marco Moletta, Michael Welle, Guillem Alenyà, Hang Yin, Zackory Erickson, David Held, et al. Unfolding the literature: A review of robotic cloth manipulation. *arXiv preprint arXiv:2407.01361*, 2024. 74

[194] Alberta Longhini, Michael C Welle, Zackory Erickson, and Danica Kragic. AdaFold: Adapting folding trajectories of cloths via feedback-loop manipulation. *arXiv preprint arXiv:2403.06210*, 2024. 74, 75

[195] Alberta Longhini, Marcel Büsching, Bardienus Pieter Duisterhof, Jens Lundell, Jeffrey Ichnowski, Mårten Björkman, and Danica Kragic. Cloth-splatting: 3d cloth state estimation from rgb supervision. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2845–2865. PMLR, 06–09 Nov 2025. URL `https://proceedings.mlr.press/v270/longhini25a.html`. 73, 79, 81

[196] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`. 97

[197] G. Lowe. SIFT-the scale invariant feature transform. *IJCV*, 2004. 28, 34

[198] Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-IO 2: Scaling Autoregressive Multimodal Models with Vision, Language, Audio, and Action. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 26439–26455, 2024. doi: 10.1109/CVPR52733.2024.02497. 165

[199] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, pages 800–809, 2024. 46, 58, 59, 60, 61, 65, 67, 68, 69, 70, 76, 81, 82

[200] Bin Luo and Edwin R. Hancock. Procrustes alignment with the EM algorithm. In *ICCAIP*, 1999. 35

[201] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for visual manipulation of deformable objects. In *IEEE ICRA*, pages 8266–8273, 2022. 74

[202] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. 80, 122, 134, 135

[203] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, abs/1703.09312, 2017. URL http://arxiv.org/abs/1703.09312. 47

[204] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019. 90

[205] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *CoRL*, pages 734–743. PMLR, 2018. 75

[206] Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12923–12932, 2023. 91

[207] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470, 2019. 91

[208] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 2019. 35

[209] 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. In *ECCV*, 2020. 26, 44, 46, 48, 50, 52, 60, 76

[210] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR*, 2022. 46

[211] Itamar Mishani, Hayden Feddock, and Maxim Likhachev. Constant-time motion planning with anytime refinement for manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, page 10337–10343. IEEE, May 2024. doi: 10.1109/icra57147.2024.10611675. URL http://dx.doi.org/10.1109/ICRA57147.2024.10611675. 90

[212] David Mizrahi, Roman Bachmann, Oğuzhan Fatih Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4M: Massively Multimodal Masked Modeling. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 36, pages 58363–58408, 2023. 165

[213] Kai Mo, Yuhong Deng, Chongkun Xia, and Xueqian Wang. Learning language-conditioned deformable object manipulation with graph dynamics. *arXiv preprint arXiv:2303.01310*, 2023. 75

[214] Andrew S Morgan, Bowen Wen, Junchi Liang, Abdeslam Boularias, Aaron M Dollar, and Kostas Bekris. Vision-driven compliant manipulation for reliable, high-precision assembly tasks. *RSS*, 2021. 61

[215] Muhammad Husnain Mubarik, Ramakrishna Kanungo, Tobias Zirr, and Rakesh Kumar. Hardware acceleration of neural graphics, 2023. 46

[216] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4): 102:1–102:15, July 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127. 46, 48, 51, 64

[217] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015. 26

[218] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 892–909. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/nair23a.html. 123, 146

[219] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. ISSN 1467-8659. doi: 10.1111/cgf.14340. URL https://doi.org/10.1111/cgf.14340. 46

[220] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015. doi: 10.1109/CVPR.2015.7298631. 61

[221] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 46

[222] NVIDIA et al. Cosmos 3: Omnimodal World Models for Physical AI, June 2026. arXiv:2606.02800 [cs.CV]. 165

[223] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference*, 2011. URL https://api.semanticscholar.org/CorpusID:8677556. 61

[224] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011. doi: 10.1109/ICRA.2011.5979561. 10, 12, 13, 18

[225] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. xvii, xxiv, 58, 91, 93, 94, 97, 99, 101, 109, 125, 164, 166, 176

[226] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, and Abraham Lee. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, May 2025. URL http://arxiv.org/abs/2310.08864. arXiv:2310.08864 [cs]. 144

[227] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-Action Models for Generalizable Robot Control Beyond VLAs, December 2025. URL http://arxiv.org/abs/2512.15692. arXiv:2512.15692 [cs]. 146, 163

[228] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019. 91

[229] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, pages 5845–5854, 2021. 60, 76

[230] William Peebles and Saining Xie. Scalable diffusion models with transformers. URL https://arxiv.org/abs/2212.09748, 2023. 106, 112, 166

[231] Rui Peng, Rongjie Wang, Zhenyu Wang, Yawen Lai, and Ronggang Wang. Rethinking depth estimation for multi-view stereo: A unified representation. In *CVPR*, 2022. 26

[232] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 91, 97, 99

[233] M. V. Peppa, J. P. Mills, K. D. Fieber, I. Haynes, S. Turner, A. Turner, M. Douglas, and P. G. Bryan. Archaeological feature detection from archive aerial photography with a SfM-MVS and image enhancement pipeline. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2018. 26

[234] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017. URL http://arxiv.org/abs/1709.07871. arXiv:1709.07871 [cs.CV]. 149

[235] Vladimír Petrík and Ville Kyrki. Feedback-based fabric strip folding. In *IEEE/RSJ IROS*, pages 773–778, 2019. 75

[236] Cody J. Phillips, Matthieu Lecce, and Kostas Daniilidis. Seeing glassware: from edge detection to pose estimation and shape recovery. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016. doi: 10.15607/RSS.2016.XII.021. 44, 47

[237] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, 2024. 113

[238] Poly Haven. Poly haven: Free high-quality assets, 2026. 134

[239] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10313–10322, 2021. 60, 63, 76

[240] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 128

[241] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies, October 2022. URL `http://arxiv.org/abs/2206.04670`. arXiv:2206.04670 [cs]. 158

[242] Shengyi Qian, Kaichun Mo, Valts Blukis, David F Fouhey, Dieter Fox, and Ankit Goyal. 3d-mvp: 3d multiview pretraining for manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22530–22539, 2025. 123

[243] Srikumar Ramalingam and Peter Sturm. A unifying model for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1309–1319, 2017. doi: 10.1109/TPAMI.2016.2592904. 12

[244] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. xvii, 93, 94

[245] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2016. doi: 10.1109/ICRA.2016.7487628. xiii, 11, 12, 13

[246] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *CoRR*, abs/2103.13744, 2021. URL `https://arxiv.org/abs/2103.13744`. 46

[247] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, pages 10881–10891, 2021. doi: 10.1109/ICCV48922.2021.01072. xxiii, 35, 37

[248] Juntao Ren, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg. Motion Tracks: A Unified Representation for Human-Robot Transfer in Few-Shot Imitation Learning, October 2025. URL `http://arxiv.org/abs/2501.06994`. arXiv:2501.06994 [cs]. 144, 146

[249] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024. 97

[250] Jerome Revaud, Jon Almazán, Rafael S. Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019. 29

[251] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2D2: Reliable and repeatable detector and descriptor. In *NeurIPS*, 2019. 28, 40

[252] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 92, 106, 126

[253] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76: 38–47, 2018. 18

[254] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 28

[255] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022. 92

[256] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J Black. Scanimate: Weakly supervised learning of skinned clothed avatar networks. In *IEEE/CVF CVPR*, pages 2886–2897, 2021. 75

[257] Gautam Salhotra, I-Chun Arthur Liu, Marcus Dominguez-Kuhne, and Gaurav S Sukhatme. Learning deformable object manipulation from expert demonstrations. *RA-L*, 7(4):8775–8782, 2022. 74, 75

[258] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *ICML*, pages 8459–8468. PMLR, 2020. 74, 77, 80, 122

[259] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 29

[260] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 28, 34, 37

[261] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018. 39

[262] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701, 2006. doi: 10.1109/IROS.2006.282372. 10

[263] Davide Scaramuzza, Agostino Martinelli, and Roland Y. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45, 2006. 10

[264] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. Dart: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, 39:239–258, 2015. URL `https://api.semanticscholar.org/CorpusID:254251877`. 61

[265] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. doi: 10.1109/CVPR.2016.445. 26, 28, 34, 35, 36, 38, 44, 51

[266] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, pages 501–518, 2016. doi: 10.1007/978-3-319-46487-9_31. 26

[267] T. Schops, V. Larsson, M. Pollefeys, and T. Sattler. Why having 10,000 parameters in your camera model is better than twelve. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2532–2541, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.00261. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00261`. xxiii, 10, 12, 13, 14, 16, 19, 21

[268] Thomas Schops, Johannes L. Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. xxv, 26, 35, 111, 112, 113

[269] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, et al. Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor. In *IEEE/RSJ IROS*, pages 9651–9658, 2020. 75

[270] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos, 2022. URL https://arxiv.org/abs/2212.04498. 123

[271] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks, 2022. 58

[272] Xiaoyu Shi, Zhaoyang Huang, Weikang Bian, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. VideoFlow: Exploiting temporal cues for multi-frame optical flow estimation. In *IEEE/CVF ICCV*, pages 12435–12446, 2023. 76

[273] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 746–760, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33715-4, 2012. xxv, 111, 112, 113

[274] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Trans. Mach. Learn. Res.*, 2026. URL https://openreview.net/forum?id=2NlGyqNjns. 174

[275] Cameron Smith, David Charatan, Ayush Tewari, and Vincent Sitzmann. FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent. In *ECCV*, 2024. xxiii, 26, 28, 35, 36, 37, 38

[276] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, 2006. 28

[277] Arun Somani, Thomas S. Huang, and Steven D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 698–700, 1987. xxiv, 98, 99

[278] Liangchen Song, Xuan Gong, Benjamin Planche, Meng Zheng, David Doermann, Junsong Yuan, Terrence Chen, and Ziyan Wu. Pref: Predictability regularized neural motion fields. In *ECCV*, 2022. 61

[279] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer, 2007. 79

[280] Gabriela Ben Melech Stan, Diana Wofk, Scottie Fox, Alex Redden, Will Saxton, Jean Yu, Estelle Aflalo, Shao-Yen Tseng, Fabio Nonato, Matthias Muller, and Vasudev Lal. Ldm3d: Latent diffusion model for 3d. URL https://arxiv.org/abs/2305.10853, 2023. 107

[281] Peter Sturm and Srikumar Ramalingam. A generic concept for camera calibration. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 1–13, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24671-8. 12

[282] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. In *ICLR*, 2020. 112, 166

[283] Zhaoqi Su, Tao Yu, Yangang Wang, and Yebin Liu. Deepcloth: Neural garment representation for shape and style editing. *IEEE TPAMI*, 45(2):1581–1593, 2022. 75

[284] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 46

[285] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87 (1):236–252, 1995. ISSN 0010-4655. doi: https://doi.org/10.1016/0010-4655(94)00170-7. URL https://www.sciencedirect.com/science/article/pii/0010465594001707. Particle Simulation Methods. 122

[286] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 46

[287] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 28, 40

[288] Priya Sundaresan, Rika Antonova, and Jeannette Bohgl. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *2022 IEEE/RSJ IROS*, pages 10828–10835. IEEE, 2022. 74

[289] Richard Sutton. The bitter lesson. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019. 104

[290] Marwan Taher, Ignacio Alzugaray, and Andrew J Davison. Fit-ngp: Fitting object models to neural graphics primitives. *arXiv preprint arXiv:2401.02357*, 2024. 58, 61

[291] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 39

[292] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis, 2022. 46

[293] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 46, 51, 54

[294] Zhenggang Tang, Balakumar Sundaralingam, Jonathan Tremblay, Bowen Wen, Ye Yuan, Stephen Tyree, Charles Loop, Alexander Schwing, and Stan Birchfield. Rgb-only reconstruction of tabletop scenes for collision-free manipulator control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1778–1785. IEEE, 2023. 90

[295] SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images. 2025. URL `https://arxiv.org/abs/2511.16624`, 2025. 104

[296] Z-Image Team. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025, 2025. 106

[297] Team Wan. Wan2.2-TI2V-5B. Model release, July 2025. URL `https://huggingface.co/Wan-AI/Wan2.2-TI2V-5B`. 174

[298] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*, pages 402–419. Springer, 2020. 61, 67, 69, 70, 76, 81, 82

[299] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *NeurIPS*, 2021. 36, 37

[300] Zachary Teed and Jia Deng. RAFT-3D: Scene flow using rigid-motion embeddings. In *IEEE/CVF CVPR*, pages 8375–8384, 2021. 76

[301] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 2002. 26

[302] Tongxuan Tian, Haoyang Li, Bo Ai, Xiaodi Yuan, Zhiao Huang, and Hao Su. Diffusion dynamics models with generative state estimation for cloth manipulation. *Conference on Robot Learning (CoRL)*, 2025. 122

[303] Christian Toepfer and Tobias Ehlgen. A unifying omnidirectional camera model and its applications. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–5, 2007. doi: 10.1109/ICCV.2007.4409207. xxiii, 21

[304] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *ICCV*, 2013. 29, 31

[305] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *ECCV*, 2020. 31

[306] Philip H. S. Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 2000. 26

[307] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In *Vision Algorithms: Theory and Practice*. Springer Berlin Heidelberg, 2000. 26, 34

[308] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4190–4200, June 2023. 46

[309] Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith, and Stan Birchfield. 6-dof pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark. In *International Conference on Intelligent Robots and Systems (IROS)*, 2022. xxiv, 96, 99

[310] Steffen Urban, Jens Leitloff, and Stefan Hinz. Improved wide-angle, fisheye and omnidirectional camera calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:72–79, 2015. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2015.06.005. URL https://www.sciencedirect.com/science/article/pii/S0924271615001616. 10, 12

[311] V. Usenko, N. Demmel, and D. Cremers. The double sphere camera model. In *Proc. of the Int. Conference on 3D Vision (3DV)*, September 2018. xxiii, 10, 12, 13, 21

[312] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 91

[313] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *CoRR*, abs/1908.00463, 2019. URL http://arxiv.org/abs/1908.00463, 2019. 111, 112, 113

[314] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023. 61, 146

[315] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV*, volume 2, pages 722–729. IEEE, 1999. 61

[316] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 46

[317] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *IJCV*, 115:1–28, 2015. URL https://api.semanticscholar.org/CorpusID:3358790. 61

[318] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2024. 92

[319] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. GaMeS: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024. 76

[320] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. URL `https://arxiv.org/abs/2503.20314`, 2025. 112, 174

[321] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis, 2021. 61

[322] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. PatchmatchNet: Learned multi-view patchmatch stereo. In *CVPR*, 2021. 26

[323] Jianyuan Wang, Christian Rupprecht, and David Novotny. PoseDiffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *ICCV*, 2023. 35, 37

[324] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Visual Geometry Grounded Deep Structure From Motion. In *CVPR*, 2024. 26, 28, 35, 36, 38

[325] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 94, 96, 122

[326] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-ω. URL `https://arxiv.org/abs/2605.15195`, 2026. 116

[327] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198. IEEE, 2016. 18

[328] Mingxin Wang, Bin Hu, Bin Qian, Kaitao Jiang, Haoning Wu, Feng Yan, Bowen Jing, Ruiyang Hao, Enyi Wang, Kangning Niu, Yandan Yang, Mu Xu, Yan Wang, Houde Liu, and Tianlun Li. ST-WAM: Semantic-Temporal World Action Model for Robust Manipulation under Visual Distribution Shifts, July 2026. arXiv:2607.28993 [cs.RO]. 162, 163

[329] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion. In *CVPRW*, 2020. 97, 99

[330] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *IEEE/CVF ICCV*, pages 19738–19749, 2023. 61, 67, 69, 76

[331] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025. 94

[332] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. MatchFormer: Interleaving attention in transformers for feature matching. In *ACCV*, 2022. 28

[333] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5261–5271, 2025, 2025. 106

[334] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. URL `https://arxiv.org/abs/2507.02546`, 2025. xxv, 106, 112, 113, 114, 115

[335] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 26, 27, 28, 29, 32, 33, 37, 80, 85, 91, 94, 95, 96, 104, 122

[336] Wenbo Wang, Gen Li, Miguel Zamora, and Stelian Coros. Trtm: Template-based reconstruction and target-oriented manipulation of crumpled cloths. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12522–12528. IEEE, 2024. 85

[337] Yi Ru Wang, Yuchi Zhao, Haoping Xu, Saggi Eppel, Alan Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Mvtrans: Multi-view perception of transparent objects, 2023. 47

[338] Yufei Wang, David Held, and Zackory Erickson. Visual haptic reasoning: Estimating contact forces by observing deformable object interactions. *RA-L*, 7(4): 11426–11433, 2022. 77

[339] Yufei Wang, Zhanyi Sun, Zackory Erickson, and David Held. Learning to dress people with diverse poses and garments. *RSS*, 2023. 75

[340] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 79

[341] Ethan Weber, Norman Müller, Yash Kant, Vasu Agrawal, Michael Zollhöfer, Angjoo Kanazawa, and Christian Richardt. Fillerbuster: Multi-view scene completion for casual captures, 2025. arXiv:2502.05175. 90, 94

[342] Tong Wei, Yash Patel, Alexander Shekhovtsov, Jiri Matas, and Daniel Barath. Generalized differentiable RANSAC. In *ICCV*, 2023. 26

[343] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021. 46

[344] Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. Learning super-features for image retrieval. In *ICLR*, 2022. 39

[345] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020. 58, 61, 97

[346] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. Catgrasp: Learning category-level task-relevant grasping in clutter from simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6401–6408. IEEE, 2022. 90

[347] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *RSS*, 2022. 61

[348] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024. 58, 61, 96, 97, 122

[349] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5249–5260, 2025. 122, 130, 157

[350] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *RSS*, 2023. 58, 61, 123, 132, 144, 146, 151, 153, 164

[351] William F Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. *arXiv preprint arXiv:2406.19800*, 2024. 122

[352] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270, 1989. 109

[353] Chao-Yuan Wu, Justin Johnson, Jitendra Malik, Christoph Feichtenhofer, and Georgia Gkioxari. Multiview compressive coding for 3D reconstruction. In *CVPR*, 2023. 91, 97, 99

[354] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, pages 20310–20320, 2023. xxiv, 46, 59, 60, 63, 64, 65, 67, 69, 70, 76

[355] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023. URL `https://arxiv.org/abs/2312.13139`. 121

[356] Kailu Wu, Fangfu Liu, Zhihan Cai, Runjie Yan, Hanyang Wang, Yating Hu, Yueqi Duan, and Kaisheng Ma. Unique3d: High-quality and efficient 3d mesh generation from a single image, 2024. xvii, xxiv, 92, 98, 99, 100

[357] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024. 128, 129, 137

[358] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *RSS*, 2020. 75

[359] W. Xian, J. Huang, J. Kopf, and C. Kim. Space-time neural irradiance fields for free-viewpoint video. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9416–9426, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00930. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00930`. 60

[360] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 134

[361] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. xvii, xxiv, 92, 98, 99, 100

[362] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.019. xxiv, 58, 61, 96, 99, 101

[363] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. Segmenting transparent objects in the wild with transformer. In Prof. Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1194–1200. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/165. URL https://doi.org/10.24963/ijcai.2021/165. Main Track. 47

[364] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-integrated 3d gaussians for generative dynamics. In *IEEE/CVF CVPR*, pages 4389–4398, 2024. 76

[365] Chi Xu, Jiale Chen, Mengyang Yao, Jun Zhou, Lijun Zhang, and Yi Liu. 6dof pose estimation of transparent object from a single rgb-d image. *Sensors*, 20(23), 2020. ISSN 1424-8220. doi: 10.3390/s20236790. URL https://www.mdpi.com/1424-8220/20/23/6790. 44, 47

[366] Gangwei Xu, Haotong Lin, Hongcheng Luo, Xianqi Wang, Jingfeng Yao, Lianghui Zhu, Yuechuan Pu, Cheng Chi, Haiyang Sun, Bing Wang, et al. Pixel-perfect depth with semantics-prompted diffusion transformers. *arXiv preprint arXiv:2510.07316*, 2025, 2025. xviii, 106, 112, 113, 114

[367] Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the Cross-Domain Manipulation Interface, October 2024. URL http://arxiv.org/abs/2407.15208. arXiv:2407.15208 [cs]. 144, 146, 147

[368] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze. *RSS*, 2022. 74, 75

[369] Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026. 123, 163, 164, 168, 169, 174

[370] Xingguang Yan, Liqiang Lin, Niloy J. Mitra, Dani Lischinski, Danny Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 97, 99

[371] Ceyuan Yang, Zhijie Lin, Yang Zhao, Fei Xiao, Hao He, Qi Zhao, Chaorui Deng, Kunchang Li, Zihan Ding, Yuwei Guo, Fuyun Wang, Fangqi Zhu, Xiaonan Nie, Shenhan Zhu, Shanchuan Lin, Hongsheng Li, Weilin Huang, Guang Shi, and Haoqi Fan. Context unrolling in omni models. URL https://arxiv.org/abs/2604.21921, 2026. 105

[372] Jiayu Yang, Wei Mao, Jose M. Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, 2020. 26

[373] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024, 2024. 106

[374] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024, 2024. 106, 112, 113

[375] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. ABot-M0: VLA Foundation Model for Robotic Manipulation with Action Manifold Learning, April 2026. arXiv:2602.11236 [cs.CV]. 167

[376] Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. CAST: Component-aligned 3d scene reconstruction from an RGB image. In *arXiv:2502.12894*, 2025. 92

[377] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL `https://arxiv.org/abs/2602.15922`. xx, 123, 163, 164, 168

[378] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 46

[379] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics*, 38(6):1–14, November 2019. ISSN 1557-7368. doi: 10.1145/3355089.3356513. URL `http://dx.doi.org/10.1145/3355089.3356513`. 63

[380] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 46

[381] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024. 90, 92, 98

[382] Chengbo Yuan, Chuan Wen, Tong Zhang, and Yang Gao. General flow as foundation affordance for scalable robot learning. *arXiv preprint arXiv:2401.11439*, 2024. xxvi, 144, 145, 146, 147, 151, 152, 158

[383] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination?, March 2026. arXiv:2603.16666 [cs.CV]. xx, 162, 163, 164, 169

[384] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. STaR: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *IEEE/CVF CVPR*, pages 13144–13152, 2021. 76

[385] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. XIRL: Cross-embodiment Inverse Reinforcement Learning, December 2021. URL `http://arxiv.org/abs/2106.03911`. arXiv:2106.03911 [cs]. 146

[386] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems*, 2024. 132, 135, 145, 146, 148, 151, 153, 155

[387] Jason Y. Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. NeRS: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. In *Conference on Neural Information Processing Systems*, 2021. 46

[388] Jason Y. Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose: Predicting probabilistic relative rotation for single objects in the wild. In *ECCV*, 2022. 37

[389] Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *ICLR*, 2024. 37, 46

[390] Jingyang Zhang, Shiwei Li, Yuanxun Lu, Tian Fang, David McKinnon, Yanghai Tsin, Long Quan, and Yao Yao. Jointnet: Extending text-to-image diffusion for dense distribution modeling. URL `https://arxiv.org/abs/2310.06347`, 2023. 105, 107, 113

[391] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. 122, 131

[392] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-grid neural dynamics for learning deformable object models from rgb-d videos. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025. xviii, xxvi, 120, 121, 122, 128, 129, 130, 131, 137, 139

[393] Mingtong Zhang, Kaifeng Zhang, and Yunzhu Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. In *8th Annual Conference on Robot Learning*, 2024. 128, 129, 137

[394] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. *CVPR*, 2022. 46

[395] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. doi: 10.1109/34.888718. 12

[396] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, April 2023. URL http://arxiv.org/abs/2304.13705. arXiv:2304.13705 [cs]. 146

[397] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data, 2026. URL https://arxiv.org/abs/2602.16710. 121

[398] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *CVPR*, pages 19855–19865, 2023. 61, 68, 81

[399] Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3DFlowAction: Learning Cross-Embodiment Manipulation from 3D Flow World Model, June 2025. URL http://arxiv.org/abs/2506.06199. arXiv:2506.06199 [cs]. 144, 146

[400] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, 2025. 122, 162, 163

[401] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *SIGGRAPH*, 2018. xxiii, 35, 37

[402] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the Continuity of Rotation Representations in Neural Networks, June 2020. URL http://arxiv.org/abs/1812.07035. arXiv:1812.07035 [cs]. 151, 155

[403] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets. In *Proc. Robot.: Sci. Syst. (RSS)*, 2025. doi: 10.15607/RSS.2025.XXI.015. 163, 169

[404] Haoyi Zhu, Yating Wang, Di Huang, Weicai Ye, Wanli Ouyang, and Tong He. Point Cloud Matters: Rethinking the Impact of Different Observation Spaces on Robot Learning, October 2024. URL http://arxiv.org/abs/2402.02500. arXiv:2402.02500 [cs]. 145, 146

[405] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, et al. Challenges and outlook in robotic manipulation of deformable objects. *Robotics & Automation Magazine*, 29(3):67–77, 2022. 74

[406] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew William Fitzgibbon, Charles T. Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33: 1–12, 2014. URL `https://api.semanticscholar.org/CorpusID:9616070`. 61