

Learning to Interact with the 3D World

by

Shengyi Qian

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2024

Doctoral Committee:

Assistant Professor David Fouhey, Co-Chair
Professor Joyce Chai, Co-Chair
Associate Professor Yasutaka Furukawa, Simon Fraser University
Professor Odest Chadwicke Jenkins
Professor Stella Yu

Shengyi Qian

syqian@umich.edu

ORCID iD: 0000-0003-0262-2412

© Shengyi Qian 2024

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my PhD advisor, David Fouhey, for his unwavering support, guidance, and mentorship throughout my doctoral journey. When I first joined David’s lab as an undergraduate with limited knowledge of proposing ideas, designing networks, and writing papers, David patiently taught me all these things from scratch. I still fondly remember my first CVPR draft, filled with David’s edits, which served as a testament to his dedication to my growth. As I became more senior and independent, David continued to support me and provide resources to pursue my own ideas. Beyond academic guidance, David became a friend and confidant, exchanging gossip and providing excellent wine recommendations for my wedding. His emotional support made the challenging process of obtaining a PhD more manageable. I cannot imagine a better PhD advisor.

I am also immensely grateful to Joyce Chai, my PhD co-advisor after David’s move to NYU. Joyce set a remarkable example of what it means to be a professor, graciously supporting me whenever I needed assistance. She was always available for meetings and helped me navigate numerous logistical issues. Working with Joyce provided me with the opportunity to expand my research into the domain of NLP, and I feel incredibly fortunate to have worked with her.

I am indebted to Jia Deng, my undergraduate research advisor, as well as my mentors Weifeng Chen and Hei Law. Joining Jia’s lab as a junior undergraduate student at the University of Michigan was a pivotal moment in my academic journey. Hei, my first mentor, introduced me to training deep networks from scratch, while Weifeng, my second mentor, opened my eyes to the world of 3D computer vision. Weifeng’s continued mentorship after graduation and our collaboration at AWS have been invaluable.

I would like to extend my thanks to Andrew Owens and Justin Johnson, who played multiple important roles during my PhD. Andrew served as my prelim exam committee member, my first faculty collaborator, and taught me how to write an excellent CVPR paper. Justin gave me consistent guidance throughout my PhD. He taught me the importance of having a long-term vision when conducting research, and show me the difference between academic and industry research.

I am grateful to my committee members: Stella Yu, Chad Jenkins, and Yasutaka Fu-

rukawa. They provide helpful feedbacks to my dissertation, and I appreciate their flexibility to attend my dissertation defense. Especially, Yasu's career advice was invaluable.

I would like to pay tribute to my other internship mentors: Georgia Gkioxari, Alexander Kirillov, Zhuowen Tu, Kaichun Mo, Ankit Goyal, Valts Blukis, Min Bai, Xiong Zhou, and Dieter Fox. Georgia was my first internship mentor and provided an exceptional experience that motivated me to pursue a career as an industry researcher. Alex taught me most of my segmentation tricks, while Zhuowen Tu, one of the most knowledgeable researchers I have worked with, provided extensive guidance on Vision Language models. Ankit Goyal, Kaichun Mo, Valts Blukis, and Dieter Fox introduced me to the intricacies of real robotics research.

I am thankful for my collaborators: Linyi Jin, Ziyang Chen, Chris Rockwell, Siyi Chen, Jianing Yang, Xuweiyi Chen, and Yuhang Zhou. I also learned a great deal from Tiange Luo, Ang Cao, Mohamed El Banani, Richard Higgins, Nilesh Kulkarni, Karen Desai, Gengshan Yang, Yixuan Wang, Wei-chiu Ma, Cheng Chi, and David Fan. Tiange and Mo provided excellent feedback on my paper drafts, while Yixuan Wang and Cheng Chi patiently answered my robotics questions.

I would like to express my heartfelt thanks to my parents for their unwavering support. Finally, and most importantly, I must thank my wife, Jing Zhu, for her love, encouragement, and support throughout this incredible journey. Her presence and understanding have been the bedrock upon which I have built my academic success.

TABLE OF CONTENTS

Acknowledgments	ii
List of Figures	vii
List of Tables	xiii
List of Appendices	xvi
Abstract	xvii

Chapter



1 Introduction	1
2 Volumetric Reconstruction from Sparse Views	6
2.1 Introduction	7
2.2 Related Work	8
2.3 Approach	9
2.3.1 Object Branch	10
2.3.2 Camera Branch	11
2.3.3 Stitching Object and Camera Branches	12
2.3.4 Implementation Details	13
2.4 Experiments	14
2.4.1 Experimental Setup	15
2.4.2 Full Scene Evaluation	17
2.4.3 Inter-view Object Affinity Matrix	19
2.4.4 Stitching Stage	20
2.4.5 Failure Cases	22
2.4.6 Results on NYU Dataset	23
2.5 Conclusion	23
3 3D Semantic Segmentation from Novel Viewpoints	24
3.1 Introduction	25
3.2 Related Work	27







3.3	Approach	28
3.3.1	Semantic Segmentation Module	29
3.3.2	Semantic 3D Representation	29
3.3.3	Learning Objective	31
3.4	Experiments	32
3.4.1	Experiments on Hypersim	33
3.4.2	Generalization to Replica and iPhone Examples	37
3.5	Conclusion	39
4	Understanding 3D Object Articulation in Internet Videos	40
4.1	Introduction	41
4.2	Related Work	42
4.3	Data Collection	43
4.4	Approach	45
4.4.1	3D Articulation Detection Network	45
4.4.2	Temporal Optimization	48
4.5	Experiments	48
4.5.1	Experimental Setup	48
4.5.2	Results	52
4.5.3	Generalization Results	53
4.5.4	Limitations and Failure Modes	53
4.6	Conclusion	54
5	Understanding 3D Object Interaction from a Single Image	55
5.1	Introduction	56
5.2	Related Works	57
5.3	Overview	58
5.4	3D Object Interaction Dataset	59
5.5	Approach	61
5.5.1	Backbone	61
5.5.2	Prediction Heads	61
5.5.3	Implementation Details	62
5.6	Experiments	63
5.6.1	Experimental Setup	63
5.6.2	Results	65
5.6.3	Generalization Results	67
5.6.4	Limitations and Failure Modes	68
5.7	Conclusion	69
6	Grounding Affordance from Vision Language Models	72
6.1	Introduction	72
6.2	Related Work	75
6.3	Approach	76
6.3.1	Overview	76
6.3.2	Network Architecture	78



6.4	Experiments	79
6.4.1	Experimental Setup	79
6.4.2	Dataset	81
6.4.3	Results	84
6.4.4	Ablation	85
6.4.5	Pseudodepth as Inputs	85
6.4.6	Generalization to Internet Images	85
6.4.7	Failure Examples	86
6.5	Conclusion	87
7	3D Multiview Pretraining for Robotic Manipulation	88
7.1	Introduction	88
7.2	Related Work	90
7.3	Approach	91
7.3.1	Background on Robotic View Transformer (RVT).	92
7.3.2	3D Multi-View Pretraining (3D-MVP)	93
7.3.3	Finetuning on Downstream Manipulation Tasks	94
7.4	Experiments	95
7.4.1	Validating 3D Masked Autoencoding	95
7.4.2	Results on RLBench	96
7.4.3	Results on COLOSSEUM	97
7.4.4	Ablation Studies	99
7.5	Conclusion	101
8	Future Directions	103
8.1	Multimodal LLM Agent	103
8.2	Dynamic Digital Twins	104
9	Conclusion	105
	Appendices	107
	Bibliography	135

LIST OF FIGURES




2.1	Given two views from unknown cameras, we aim to extract a coherent 3D space in terms of a set of volumetric objects placed in the scene. We represent the scene with a factored representation [265] that splits the scene into per-object voxel grids with a scale and pose.	6
2.2	Our approach. We pass the two RGB image inputs into two branches that extract evidence, which is then fused together to stitch a final result. Our first network, object branch , is a detection network following [138] that produces a set of objects in terms of voxels and a transformation into the scene. We also predict an object embedding which we can use to form an affinity matrix between objects across images. Our second network, camera branch , is a siamese network that predicts a distribution over translations and rotations between the cameras. Finally our stitching stage examines the evidence from the networks and produces a final prediction.	9
2.3	Qualitative results on the SUNCG test set [250]. The final 3D predictions are shown in three different camera poses (1) the same camera as image 1; (2) the same camera as image 2; (3) a bird view to see all the objects in the whole scene. In the prediction, red/orange objects are from the left image, blue objects are from the right image, green/yellow objects are stitched.	15
2.4	Comparison between Associative3D and alternative approaches. Row 1: Associative3D fixes the incorrect top-1 relative camera pose in light of a single bed in the room. Row 2: NMS works when the relative camera pose is accurate. Row 3: Associative3D outperforms all alternative approaches in finding correspondence in object clutter.	17
2.5	Visualization of the stitching stage. The affinity matrix generates proposals of corresponding objects, and then the stitching stage removes outliers by inferring the most likely explanation of the scene.	20
2.6	Representative failure cases on the SUNCG test set [250]. Row 1: The input images are ambiguous. There can be two or three beds in the scene. Row 2: The single-view backbone does not produce a reasonable prediction. Row 3: This is challenging because all chairs are the same.	22
2.7	Qualitative results on NYUv2 dataset [245]. Sideview corresponds to the camera view slightly transformed from the image 2 camera position.	22

3.1	We propose ViewSeg which takes as input (a) a few images of a novel, previously unobserved scene, and recognizes the scene from novel viewpoints. The novel viewpoint, in the form of camera coordinates, queries (b) our predicted 3D representation to produce (c) semantic segmentations from the view <i>without access to the view’s RGB image</i> . ViewSeg trains on hundreds of scenes using multi-view 2D annotations and no 3D supervision.	24
3.2	RGB is a poor bottleneck representation for semantic synthesis. While one may be certain of the labels of the new view (e.g., walls, tables), synthesizing the particular wall or table pixels from few views is challenging: outputs are often blurry (top) or collapse when extrapolating (bottom).	26
3.3	Our model, ViewSeg, uses a few source RGB views (here 2) of a <i>novel scene</i> and predicts the semantic segmentation of that scene from a novel target viewpoint. Our approach embeds each source view into a latent semantic space with a 2D segmentation module; this space is used to predict radiance, density and distribution over semantic classes at each spatial 3D location via an MLP. The final semantic segmentation is created by volumetric rendering from the target viewpoint. We generalize to unseen scenes by training on hundreds of diverse scenes and thousands of source-target pairs, and no 3D semantic supervision.	28
3.4	Predictions on Hypersim. For each example, we show the 4 input RGB views of a novel scene, the semantic and depth predictions from Semantic NeRF, PixelNeRF++ and our ViewSeg. We also show the true RGB, semantic and depth map of the target view. Our model does not have access to ground truth observations from the target view at test time. Depth colormap (scaled per example): min  max.	33
3.5	We test on real images taken with an iPhone 12. Despite noisy camera poses estimated by COLMAP [234] following [189], ViewSeg obtains reasonable segmentation results.	37
3.6	Predictions on Replica. For each example, we show the 4 input views (left), ground truth RGB, semantic and depth from the novel view (middle) and ViewSeg’s semantic and depth predictions (right). Depth colormap (scaled per example): min  max.	37
3.7	3D semantic reconstructions of novel scenes.	38
3.8	“Place-in-scene” AR demo with ViewSeg.	39
4.1	Given an ordinary video, our system produces a 3D planar representation of the observed articulation. The 3D renderings illustrate how the microwave (in Pink) can be articulated in 3D space. We also show the predicted rotation axis using a Blue arrow.	40
4.2	Overview of our approach. (a) Given an ordinary video clip, we first apply our 3D Articulation Detection Network (3DADN) to detect 3D planes can be articulated for each frame. (b) We then apply temporal optimization to fit the articulation model. Final results are demonstrated in both 2D image and 3D rendering.	45

4.3	Predictions on Internet videos. For each example, we show the input (left), detected 2D planes and how they will be articulated using the predicted articulation axes and surface normals (middle). We also show 3D renderings to illustrate how these common objects are articulated in the 3D space (right). The predicted rotation axis is shown as the Blue arrow, and translation axis is the Pink arrow.	47
4.4	We compare our approach with four baselines. See detailed discussions in the text. We show translation in Pink and rotation in Blue , except D3D-HOI which uses a different detector.	49
4.5	Qualitative results on Charades dataset. Without finetuning on Charades data, our model obtains strong performance on detecting and characterizing 3D articulation.	53
4.6	Typical failure modes. (1) Ambiguity of articulation type; (2) Axis outside of the frame or ambiguity of articulation axis location due to symmetry; (3) Object has complex motions (a person moving an object while articulating it; the rotation axis is outside of the articulating surface).	54
5.1	Given a single image and a set of query points our approach predicts: (a) whether the object at the location can be moved  , its rigidity  and articulation class  , and location  ; (b) an affordance  and action  ; and (c) potential 3D interaction for articulated objects. This ability can assist intelligent agents to better manipulate objects or explore the 3D scene.	55
5.2	Example annotations of our 3DOI dataset. Our images come from Internet videos [211], egocentric videos [47] and renderings of 3D dataset [62]	59
5.3	Overview of our approach. The inputs of our network is a single image and a set of query points. For each query point, it predicts the potential 3D interaction, in terms of movable, location, rigidity, articulation, action and affordance.	60
5.4	Results on our 3DOI dataset. (Row 1, 2) Our approach can correctly recognize articulated objects, as well as its type (rotation or translation), axes, and affordance. (Row 3, 4) Our approach can recognize rigid and nonrigid objects in egocentric video. (Row 5) Our approach can recognize objects need to be moved by two hands, such as a TV. We note that the affordance of these objects have multiple solutions. Affordance is zoomed manually for better visualization.	63
5.5	Prediction of 3D potential interaction of articulated objects. In prediction 1, 2, and 3, we rotate the object along its rotation axis, or translate the object along its normal direction.	66
5.6	Comparison of 3DADN [211], SAPIEN [287] and our approach. 3DADN has a strong performance when humans are present. However, it has difficulty detecting objects without human activities. SAPIEN does not generalize well to real images. However, it is sometimes better than 3DADN when humans are not present.	67

5.7	Comparison of InteractionHotspots [196] and our approach. We find InteractionHotspots typically makes a cloud like probability map on our data. Our model is very confident about its prediction, while there can be multiple solutions. Prediction and GT are zoomed manually for better visualization. Affordance colormap: min  max.	68
5.8	Results on robotics data [6]. Without finetuning, our approach generalizes well to robotics data, which indicates its potential to help intelligent agents to better manipulate objects. Row 1 and 2 are articulated objects. Row 3 and Row 4 are deformable objects. Affordance is zoomed manually for better visualization. Affordance colormap: min  max.	70
5.9	Typical failure modes of our approach. Row 1: Our predicted rotation axis is on the wrong side when the objects look symmetric. Row 2: Our predicted mask is partial when the scissors are occluded. Row 3: Our model thinks the trash bin can be picked up by 1 hand, potentially since its material looks plastic.	71
6.1	Illustration for the affordance grounding task. The input is a single image and the corresponding action (e.g, “hold”). The output is a heatmap which highlights regions one can interact. We aim to enhance the generalization capability of affordance grounding to in-the-wild objects that are unseen during training, by developing a new approach, AffordanceLLM, that takes the advantage of the rich knowledge from large-scale vision language models [163] beyond the supervision from the training images.	73
6.2	State-of-the-art vision language models, such as LLaVA [163], has rich human-object-interaction knowledge, thanks to the large-scale text pretraining. Given a question about how to interact with an object, it typically gives a reasonable solution.	74
6.3	Overview of AffordanceLLM. The inputs of our model includes a single image and a text prompt related to interaction. We use OWL-ViT [191] as the image encoder to generate image features and project it into the same hidden dimension as the large language model. As well, we use a tokenizer to encode the text prompt. The text features and image features are concatenated together and feed into the LLM. The LLM is fine-tuned to predict a special token, which is used as a query to the mask decoder to generate the final affordance map. . . .	77
6.4	Qualitative results on the test set of the hard split. LOCATE-Sup fails to learn a reasonable affordance map due to limited training data. LOCATE [147] typically predicts an affordance map which covers the whole object. 3DOI [209] focuses on a small area of the object. Overall, our approach produces the best-quality affordance predictions.	80
6.5	Ablation of different text prompts and depth. Ours w/o depth is our approach without pseudodepth as additional inputs. Ours is our full approach. We find constructing the correct text prompt typically helps our model to focus on the correct area. We believe it is because the correct text prompt would activate the world knowledge related to affordance embedded in the VLM.	82

6.6	Generalization results on random Internet images. We show the most similar objects in the training set to demonstrate how different the objects are from the ones in the training set. (Row 1, 2): AffordanceLLM generalizes to novel objects from random Internet images, while LOCATE [147] fails. (Row 3, 4): AffordanceLLM generalizes to novel actions plus novel objects. LOCATE cannot infer novel actions thus we left it blank.	86
6.7	Failure examples. (Row 1:) AffordanceLLM sometimes fails due to multiple objects present in the scene and it fails to refer to the correct object. (Row 2:) AffordanceLLM thinks humans should hold the handle to cut something using the knife, while AGD20K annotators think “cut with” should refer to the blade.	87
7.1	Overview of 3D-MVP. (a) We first pretrain a Multiview 3D Transformer using masked autoencoder on multiview RGB-D images. (b) We then finetune the pretrained Multiview 3D Transformer on manipulation tasks. Since the MVT is pretrained, the learned manipulation policy generalizes better. For example, it is more robust to changes of texture, size and lighting.	92
7.2	MAE Reconstruction results on Objaverse. We find the our pretrained multi-view transformer could generalize to unseen object instances and reconstruct multi-view images from their masked versions.	95
7.3	Results on COLOSSEUM [207]. We report the average task completion success rate for 12 environmental perturbations and no perturbation. Manipulation policies which do explicit 3D reasoning (RVT [81] works significantly better and 2D pretraining approaches (MVP [288] and R3M [197])). 3D-MVP is more robust than RVT on most perturbations. M0 = manipulation object. R0 = receiver object.	98
7.4	Pretraining MAE on only RLbench scenes leads poor generalization performance. (Left): MAE reconstruction results on unseen RLbench renderings. (Right): MAE reconstruction results on Objaverse renderings. While the reconstruction is reasonable on RLbench unseen renderings, it overfits to RLbench and does not learn a general representation.	100
A.1	t-SNE visualization of the object embedding space. Left: We assign the same color to embeddings with the same semantic labels. Right: We assign the same color to embeddings with the same 3D models.	108
A.2	Top-K accuracy of the camera branch on the validation set. The orange line shows the Top-K accuracy of K we choose for the stitching stage.	109
A.3	Additional Qualitative results on the SUNCG test set. The final 3D predictions are shown in three different camera poses (1) the same camera as image 1; (2) the same camera as image 2; (3) a bird view to see all the objects in the whole scene. In the prediction, red/orange objects are from the left image, blue objects are from the right image, green/yellow objects are stitched from both the images.	111
A.4	Comparison between Associative3D and alternative approaches. All outputs are shown from bird’s eye view.	112

A.5	Visualization of the stitching stage. The affinity matrix generates proposals of corresponding objects, and then the stitching stage removes outliers by inferring the most likely explanation of the scene. Before stitching, the thickness and darkness of the line represent the value of affinity score. The thicker / darker, the higher the value in the affinity matrix.	113
B.1	Detailed architecture of our semantic 3D representation f . Each cuboid is a linear layer where the number around it represents the input dimension. The output dimension is always 128 except the last layer of RGB and semantic prediction. Before the mean aggregation, the network takes inputs from each source view but weights are shared.	114
B.2	Predictions on Replica with noisy cameras. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). Our model does not have access to the true observations from the target view at test time. Depth colormap (scaled per example): min  max.	115
B.3	More predictions on Hypersim. Depth colormap (scaled per example): min  max.	117
B.4	More predictions on Replica. Depth colormap (scaled per example): min  max.	118
C.1	The screenshot of recognize articulated clips.	123
C.2	The screenshot of annotating bounding boxes.	124
C.3	The screenshot of annotating rotation axes.	125
C.4	The screenshot of annotating translation axes.	126
C.5	The screenshot of annotating surface normals.	127
C.6	Random examples from Sapien renderings.	128
D.1	Statistics of our 3DOI dataset. (Row 1) We show the distribution of query points, box centers, and affordance in normalized image coordinates, similar to LVIS [87] and Omni3D [13]. (Row 2) We show the distribution of object types, articulation types and movable types.	131
D.2	Example annotations of our 3DOI dataset. Row 1-2 come from Internet videos [211]. Row 3-4 come from egocentric videos [47]. Row 5-6 come from renderings of 3D dataset [62]. The dot is the query point, and ▼ is the affordance.	132

LIST OF TABLES

2.1	We report the average precision (AP) in evaluation of the 3D detection setting. All means a prediction is a true positive only if all of translation, scale, rotation and shape are correct. Shape , Translation , Rotation , and Scale mean a prediction is a true positive when a single error is lower than thresholds. We include results on the whole test set, and top 25%, 50% and 75% examples ranked by single-view predictions.	18
2.2	AUROC and rank correlation between the affinity matrix and <i>category</i> , <i>model</i> , <i>shape</i> , and <i>instance</i> , respectively. Model Category means the ability of the affinity matrix to distinguish different models given the same category / semantic label.	19
2.3	Evaluation of object correspondence with and without the stitching stage. . . .	21
2.4	Evaluation of relative camera pose from the camera branch and picked by the stitching.	21
3.1	Comparisons on Hypersim val. We report the performance for semantic segmentation (blue) and depth estimation (green) for our method, ViewSeg, an oracle which applies supervised single-image semantic segmentation and depth estimation models on the true target RGB views, two NVS variants based on PixelNeRF [305] and an explicit 3D point cloud model, CloudSeg, inspired by SynSin [280].	32
3.2	Ablating loss components. We report semantic (blue) and depth (green) performance on Hypersim val without the photometric, the semantic and the source view loss and when excluding the viewing direction from the input. Our model is reported in the last row.	35
3.3	Performance on Hypersim val with different semantic segmentation backbones. We show the performance of ViewSeg with DeepLabv3+ (DLv3+) [30] pretrained on ADE20k [318] and on ImageNet [52] and a ResNet34 [95] backbone pretrained on ImageNet [52]. The latter is used in PixelNeRF [305]. DLv3+ improves performance significantly, while ADE20k helps ever so slightly.	35
3.4	Input study on Hypersim val for varying number of source views. More views improve semantic and depth performance.	36
3.5	Performance for semantic segmentation (blue) and depth (green) on the Replica [253] test set before finetuning (noft) and after finetuning ViewSeg on Replica’s training set. We additionally report the target view oracle.	38

4.1	We report AUROC for Articulation Recognition, as well as AP for Articulation Description. To separate out difficulties in detecting articulation and characterizing its parameters, we assist Flow+Normal and 3DADN+SAPIEN with ground truth bounding box and denote it as gtbox. 3DADN+SAPIEN cannot detect most objects without the help of gtbox.	50
4.2	Evaluation on Charades dataset [243]. We only report rotation AP since Charades does not have sufficient translation motion.	53
5.1	Quantitative results on our 3DOI dataset. Cat. means category. We report accuracy for all category classification, including movable, rigid, articulation and action. We report mean IoU for box and mask, EA-Score for articulation axis, and SIM for affordance. For all metrics, the higher the better.	65
5.2	Quantitative results on robotics data [6]. Cat. means category. We report accuracy for all category classification, including movable, rigid, articulation and action. We report mean IoU for the boxes and masks, EA-Score for articulation axis, and SIM for affordance probability map. For all metrics, the higher the better.	69
6.1	Difficulty score of different splits. The lower the score, the more similar are the object categories in the train and test set.	81
6.2	Quantitative results on the <i>Easy</i> split of AGD20K [177]. Interaction-Hotspots, Cross-View-AG(+), AffCorrs and LOCATE are trained on AGD20K images with weak supervision (13,323 images). LOCATE-Sup and LOCATE-Sup-OWL, and AffordanceLLM are trained on AGD20K images with dense annotation (1,135 images). 3DOI is trained on their own dataset with dense annotation (10,000 images) [209]. AffordanceLLM is comparable to LOCATE [147] on the easy split, where test objects have similar counterparts in the training set. The best and <u>second-best</u> results are highlighted in bold and underlined, respectively.	81
6.3	Quantitative results on the <i>Hard</i> split of AGD20K [177]. Cross-View-AG(+) and LOCATE are trained on AGD20K images with weak supervision (11,889 images). LOCATE-Sup and LOCATE-Sup-OWL, and AffordanceLLM are trained on AGD20K images with dense annotation (868 images). 3DOI is trained on their own dataset with dense annotation (10,000 images) [209]. On the hard split, AffordanceLLM outperforms all baselines by a large margin, which demonstrates the superior generalization ability of our model. We do not run InteractionHotspots [196] and AffCorrs [88], as the reported model has ambiguous implementation details, or is not publicly available. The best and <u>second-best</u> results are highlighted in bold and underlined, respectively.	83
6.4	Ablation on the hard split. We validate the importance of text prompts, image encoders and pseudo depth to performance.	84
7.1	Results on RL Bench [113]. We report the task completion success rate for 18 RL Bench tasks, as well as the average success rate. 3D-MVP reaches the state-of-the-art performance on the benchmark. The pretraining is mainly helpful for tasks with medium difficulty.	97

7.2	Ablation studies on the RL Bench benchmark. We analyze the contribution of our network architecture, pretraining datasets, and the masking strategy. For each variant, we report the average task completion success rate on RL Bench [113].	99
A.1	Comparison between averaging the rotation and picking up one at random. . . .	110
A.2	Comparison between averaging the shape and picking up one at random. . . .	110
B.1	Performance for semantic segmentation (blue) and depth (green) on the Replica [253] test set with noisy cameras.	115
B.2	Extended version of Table 1 in the main paper.	119
B.3	Extended version of Table 2 in the main paper.	119
B.4	Extended version of Table 3 in the main paper.	119
B.5	Extended version of Table 4 in the main paper.	120
B.6	Extended version of Table 5 in the main paper.	120
C.1	Overall architecture for our proposed network. The backbone, RPN and plane branches are identical to [120]. The RPN predicts a bounding box for each of A anchors in the input feature map. C is the number of categories (here = 2 for rotation and translation). We use class agnostic mask because the mask head is trained on ScanNet. TConv is a transpose convolution with stride 2. ReLU is used between all Linear, Conv and TConv operations. Depth branch uses Conv and Deconv layers to generate a depthmap with the same resolution as the input image.	121

LIST OF APPENDICES

A Supplementary: Volumetric Reconstruction from Sparse Views	107
B Supplementary: 3D Semantic Segmentation from Novel Viewpoints . . .	114
C Supplementary: Understanding 3D Articulation in Internet videos . . .	121
D Supplementary: Understanding 3D Object Interaction from a Single Image	129
E Supplementary: Grounding Affordance from Vision Language Models .	133

ABSTRACT

Enabling machines to perceive, understand, and interact with the 3D world is a fundamental challenge in Computer Vision and Robotics, crucial for embodied agents operating in real-world environments. This dissertation addresses this challenge by proposing novel approaches to unify 3D reconstruction, affordance learning, and object manipulation. By developing a system that can understand and interact with arbitrary objects in diverse scenes, this research aims to enhance the ability of AI agents to navigate and operate in both physical and digital worlds.

This dissertation is structured into three primary parts, each tackling a key challenge in 3D perception and interaction. First, we introduce novel approaches for 3D scene understanding from visual observations, including Associative3D and ViewSeg, which enable machines to perceive and understand the 3D world from limited input data.

Next, we develop methodologies for learning and grounding affordances in 3D scenes, leveraging unstructured Internet videos and Vision Language Models to enhance the system’s understanding of object interactions. We propose a novel approach for predicting 3D object interactions from a single image and investigate the distillation of comprehensive world knowledge from Vision Language Models.

Finally, we extend our approach to active object manipulation, employing the developed system as a visual pretraining mechanism for robotics to improve the performance and generalization of manipulation policies.

The key contributions of this dissertation lie in the development of techniques spanning from passive 3D perception to active object manipulation. By leveraging the vast knowledge from demonstration videos and Vision Language Models and applying the system to robotic pretraining, this research takes significant steps towards endowing machines with the ability to intelligently interact with the 3D world. The proposed methodologies collectively advance the state-of-the-art in machine perception and interaction, paving the way for more capable and adaptable AI agents.

CHAPTER 1

Introduction

Enabling machines to perceive, understand and interact with the 3D world is a fundamental challenge in computer vision and robotics. The ability to reason about the 3D structure of scenes and objects, infer their affordances, and plan interactions is crucial for embodied agents operating in real-world environments. This capability would allow robots to assist humans in everyday tasks, navigate autonomously, and manipulate objects to achieve goals. However, developing such 3D perception and interaction abilities in machines remains an open problem due to several key challenges.

First, recovering the 3D structure of scenes from 2D images or videos is an ill-posed inverse problem. Multiple 3D interpretations can explain the same 2D projections, and inferring depth requires resolving these ambiguities through priors or additional views. Traditional multi-view 3D reconstruction approaches like structure-from-motion make restrictive assumptions about the scene (e.g. static, rigid) and require identifying correspondences across views, which is difficult in low-texture regions [234, 1]. More recent learning-based methods can relax these assumptions but need large datasets of ground-truth 3D annotations which are costly to acquire [153, 274, 63].

Second, even with an accurate 3D reconstruction, understanding which parts of the scene are relevant for a given interaction and how to manipulate objects to achieve an objective is non-trivial. According to J.J. Gibson, affordances refer to the opportunities for interaction provided by the environment to an agent, essentially defining the possible actions that can be performed on an object or surface [76]. In other words, affordances describe the functional and interactive properties of objects and scenes, such as the ability to grasp, push, or manipulate them. While humans can quickly infer the affordances and functionality of novel objects, this is challenging for machines since affordances depend on both the object’s geometric structure as well as its intended use which can vary based on context [177].

Third, moving from affordance to manipulation policy is nontrivial. Learning a general purpose manipulation policy through trial-and-error exploration in 3D environments is a

complex robot learning problem. Since interactions can have irreversible consequences (e.g. breaking objects), naively exploring in the real world is expensive and dangerous. Simulators and 3D virtual environments provide a safe avenue for training interaction policies [113, 306]. However, the learned policies may not generalize well to real-world scenarios due to the domain gap between synthetic and real data.

In summary, enabling machines to intelligently interact with the 3D world requires addressing three key challenges: 1) reconstructing 3D scenes from visual observations, 2) understanding scene semantics and object affordances, and 3) learning manipulation policies through autonomous exploration. While there has been significant progress on each of these fronts individually, integrating them into a complete and scalable 3D perception and interaction framework remains an open problem. The goal of this dissertation is to take a step towards addressing this challenge by proposing novel approaches that unify 3D reconstruction, affordance learning, and object manipulation.

The remainder of the dissertation is organized as follows: We will first address the challenge of reconstructing 3D scenes from visual observations in Chapters 2 and 3, where we propose novel approaches for 3D scene understanding from visual observations. More specific,

Chapter 2 explores the challenge of perceiving the 3D world from two views with unknown camera poses, which is a crucial first step for interacting with the environment. We introduce Associative3D, a novel approach that jointly estimates volumetric reconstructions, establishes inter-view object associations, and infers relative camera poses. Our method is trained and evaluated on the synthetic SUNCG dataset [250], demonstrating strong performance. We also validate its generalization capability on the real-world NYUv2 dataset [246]. However, the approach has limitations in generalizing to diverse real-world images beyond NYUv2, primarily due to the constraints of the backbone architecture [138, 265]. Retraining the backbone requires watertight 3D meshes, which are often unavailable for real-world datasets such as Matterport3D [26], ScanNet [42], and Habitat [216]. Addressing this limitation and enabling robust 3D perception on real-world data is an important direction for future research.

Chapter 3 addresses the limitation of data availability by leveraging implicit representations, specifically Neural Radiance Fields (NeRFs) [305, 188, 316]. We introduce ViewSeg, a novel approach that predicts 3D semantics from any viewpoint in a scene, using only a few input images. Our model is trained on multi-view 2D annotations from hundreds of scenes in the Hypersim dataset [225], without requiring explicit 3D supervision beyond camera poses. Notably, these annotations are easily obtainable – given a video of any scene, we can estimate camera poses using structure-from-motion techniques like Colmap [234] and sample frames for annotation. This setup is sufficient to train our model effectively. ViewSeg demonstrates

strong performance on real-world data from the Replica dataset [253], showcasing its ability to generalize to novel scenes and viewpoints. Our approach opens up new possibilities for 3D semantic understanding from limited input data, making it more practical for real-world applications.

We then tackle the challenge of understanding scene semantics and object affordances in Chapters 4, 5, and 6, where we propose novel approaches for learning and grounding affordances in 3D scenes. Chapter 4 takes a step towards this goal by learning to understand 3D object articulation from Internet videos. In this chapter, we focus on articulated objects, which has moving parts or joints that can change its shape or configuration (e.g. door, drawer). We introduce a novel system that produces a 3D planar representation of the observed articulation given an ordinary video. This problem is particularly challenging because it requires dynamic 3D understanding, and most existing 3D datasets, such as Matterport3D [26], ScanNet [42], and Gibson [285], only contain static scenes. To overcome this challenge, we design a semi-automatic pipeline that extracts articulation clips and labels from a large corpus of Internet videos, leveraging prior work on learning from videos [236, 72]. Our pipeline automatically discovers articulated objects, tracks their poses, and optimizes for consistent 3D articulation models. We train our models on this newly collected dataset and demonstrate strong generalization performance on the Charades dataset [243]. However, since the model is trained on objects that are actively being articulated in the videos, it only works on the articulation as it occurs, rather than predicting potential articulations from a single image. In reality, a single image typically contains many objects that can be interacted with and manipulated in various ways.

Chapter 5 presents a significant advancement in enabling machines to understand and interact with the 3D world from a single image. We introduce a novel task of predicting 3D object interactions given just an RGB image and a set of query points. Our transformer-based approach, powered by the newly collected 3D Object Interaction (3DOI) dataset, can predict crucial interaction-related properties for each queried object, including its movability, rigidity, articulation type, affordance, and potential actions. Movability is classified into fixtures, one-hand, and two-hand objects. The model also localizes the object in 3D by predicting its segmentation mask and depth. For articulated objects, it estimates the articulation type (rotation or translation) and the corresponding axis. Furthermore, it predicts the most likely affordance region and potential actions to interact with the object. The 3DOI dataset, comprising a diverse set of images from Internet videos [211], egocentric videos [46], and 3D scene datasets [62], enables the model to learn rich priors on object interactions. Our experiments demonstrate strong performance on 3DOI and impressive generalization to real-world robotics data [6], highlighting the model’s potential to assist intelligent agents in

understanding and manipulating objects in novel environments. This work takes a significant step towards embodied visual reasoning and bridges the gap between passive perception and active interaction in 3D scenes.

Chapter 6 further advances the goal of enabling machines to understand and interact with the 3D world from a single image. We introduce AffordanceLLM, a novel approach that leverages the rich world knowledge embedded in large-scale vision-language models (VLMs) to ground affordances for in-the-wild objects unseen during training. In contrast, 3DOI relies solely on its own dataset of 10,000 images with dense annotations for training and does not utilize any pretrained VLMs. Furthermore, AffordanceLLM supports grounding affordances for arbitrary actions beyond the ones seen during training, while 3DOI is limited to the predefined set of actions in its dataset. Our method builds upon the LLaVA backbone and introduces a mask decoder and special `<mask_token>` to predict affordance maps. We also incorporate pseudo-depth maps as additional 3D information to aid the affordance reasoning. Through extensive experiments on the AGD20K benchmark, we demonstrate that AffordanceLLM significantly outperforms 3DOI and other state-of-the-art methods, especially in terms of generalization to new object categories and novel actions. This work highlights the importance of leveraging the knowledge in vision-language models and 3D geometric cues for learning to perceive and interact with objects in the 3D world. It represents a major step towards endowing machines with the capability to reason about object affordances in diverse real-world settings, paving the way for more intelligent and interactive embodied agents.

After that, we address the challenge of learning manipulation policies in Chapter 7, where we propose a novel approach for learning manipulation policies through 3D multi-view pretraining. We propose 3D-MVP, a novel approach for 3D multi-view pretraining using masked autoencoders to improve the performance and generalization of robotic manipulation policies. 3D-MVP leverages the Robotic View Transformer (RVT) architecture [81], which uses a multi-view transformer to understand the 3D scene and predict gripper pose actions for manipulation. We pretrain the visual encoder of RVT on the large-scale Objaverse 3D object dataset using masked autoencoding [51, 92]. This pretraining allows the model to learn rich 3D representations that capture the geometry and semantics of objects. We evaluate the pretrained 3D-MVP model by finetuning it on downstream manipulation tasks from the RL Bench and COLOSSEUM benchmarks [113, 207]. The results demonstrate that the learned 3D representations lead to improved performance and robustness compared to training RVT from scratch or using 2D pretraining. 3D-MVP achieves state-of-the-art performance on these benchmarks, suggesting that 3D-aware pretraining on diverse object datasets is a promising approach for developing general-purpose robotic manipulation systems. By enabling active manipulation of objects in the 3D world, 3D-MVP represents a

significant advance over the previous chapters' focus on passive perception and understanding. It showcases the potential of self-supervised 3D representation learning to bridge the gap between visual understanding and physical interaction, paving the way for more capable and adaptable robotic systems.

Finally, we propose a few future directions in Chapter 8, including multimodal LLM agents and dynamic digital twins. For multimodal LLM agents, future works include building digital agents and robotics foundation model. For dynamic digital twins, future works include building dynamic digital twins from videos or generative models.

CHAPTER 2

Volumetric Reconstruction from Sparse Views

In this chapter, we study the problem of 3D volumetric reconstruction from two views of a scene with an unknown camera. While seemingly easy for humans, this problem poses many challenges for computers since it requires simultaneously reconstructing objects in the two views while also figuring out their relationship. We propose a new approach that estimates reconstructions, distributions over the camera/object and camera/camera transformations, as well as an inter-view object affinity matrix. This information is then jointly reasoned over to produce the most likely explanation of the scene. We train and test our approach on a dataset of indoor scenes, and rigorously evaluate the merits of our joint reasoning approach. Our experiments show that it is able to recover reasonable scenes from sparse views, while the problem is still challenging. The material in this chapter is derived from [210].

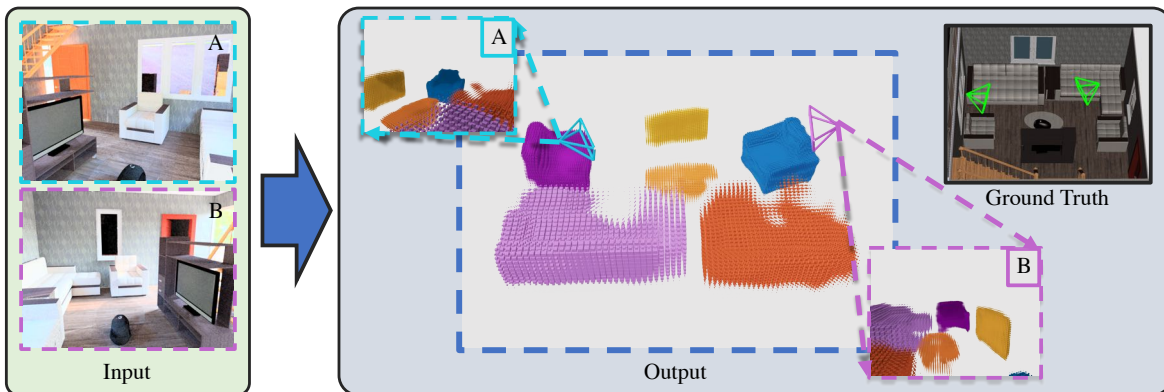


Figure 2.1: Given two views from unknown cameras, we aim to extract a coherent 3D space in terms of a set of volumetric objects placed in the scene. We represent the scene with a factored representation [265] that splits the scene into per-object voxel grids with a scale and pose.

2.1 Introduction

How would you make sense of the scene in Fig. 2.1? After rapidly understanding the individual pictures, one can fairly quickly attempt to match the objects in each: the TV on the left in image A must go with the TV on the right in image B, and similarly with the couch. Therefore, the two chairs, while similar, are not actually the same object. Having pieced this together, we can then reason that the two images depict the same scene, but seen with a 180° change of view and infer the 3D structure of the scene. Humans have an amazing ability to reason about the 3D structure of scenes, even with as little as two sparse views with an unknown relationship. We routinely use this ability to understand images taken at an event, look for a new apartment on the Internet, or evaluate possible hotels (e.g., for ECCV). The goal of this chapter is to give the same ability to computers.

Unfortunately, current techniques are not up to this challenge of volumetric reconstruction given two views from unknown cameras: this approach requires both reconstruction and pose estimation. Classic methods based on correspondence [90, 41] require many more views in practice and cannot make inferences about unseen parts of the scene (i.e., what the chair looks like from behind) since this requires some form of learning. While there has been success in learning-based techniques for this sort of object reconstructions [38, 77, 265, 138], it is unknown how to reliably stitch together the set of reconstructions into a single coherent story. Certainly there are systems that can identify pose with respect to a fixed scene [126] or a pair of views [65]; these approaches, however cannot reconstruct.

This chapter presents a learning-based approach to this problem, whose results are shown in Fig. 2.1. The system can take two views with unknown relationship, and produce a 3D scene reconstruction for both images jointly. This 3D scene reconstruction comprises a set of per-object reconstructions rigidly placed in the scene with a pose as in [265, 138, 149]. Since the 3D scene reconstruction is the union of the posed objects, getting the *3D scene reconstruction* correct requires getting both the *3D object reconstruction* right as well as correctly identifying *3D object pose*. Our key insight is that jointly reasoning about objects and poses improves the results. Our method, described in Section 2.3, predicts evidence including: (a) voxel reconstructions for each object; (b) distributions over rigid body transformations between cameras and objects; and (c) an inter-object affinity for stitching. Given this evidence, our system can stitch them together to find the most likely reconstruction. As we empirically demonstrate in Section 2.4, this joint reasoning is crucial – understanding each image independently and then estimating a relative pose performs substantially worse compared to our approach. These are conducted on a challenging and large dataset of indoor scenes. We also show some common failure modes and demonstrate transfer to NYUv2 [245]

dataset.

Our primary contributions are: (1) Introducing a novel problem – volumetric scene reconstruction from two unknown sparse views; (2) Learning an inter-view object affinity to find correspondence between images; (3) Our joint system, including the stitching stage, is better than adding individual components.

2.2 Related Work

The goal of this work is to take two views from cameras related by an unknown transformation and produce a single volumetric understanding of the scene. This touches on a number of important problems in computer vision ranging from the estimation of the pose of objects and cameras, full shape of objects, and correspondence across views. Our approach deliberately builds heavily on these works and, as we show empirically, our success depends crucially on their fusion.

This problem poses severe challenges for classic correspondence-based approaches [90]. From a purely geometric perspective, we are totally out of luck: even if we can identify the position of the camera via epipolar geometry and wide baseline stereo [206, 192], we have no correspondence for most objects in Fig. 2.1 that would permit depth given known baseline, let alone another view that would help lead to the understanding of the full shape of the chair.

Recent work has tackled identifying this full volumetric reconstruction via learning. Learning-based 3D has made significant progress recently, including 2.5D representations [63, 274, 33, 144], single object reconstruction [281, 312, 85, 224, 39], and scene understanding [35, 104, 160, 159, 60]. Especially, researchers have developed increasingly detailed volumetric reconstructions beginning with objects [38, 77, 79] and then moving to scenes [265, 138, 149, 198] as a composition of object reconstructions that have a pose with respect to the camera. Focusing on full volumetric reconstruction, our approach builds on this progression, and creates an understanding that is built upon jointly reasoning over parses of two scenes, affinities, and relative poses; as we empirically show, this produces improvements in results. Of these works, we are most inspired by Kulkarni et al. [138] in that it also reasons over a series of relative poses; our work builds on top of this as a base inference unit and handles multiple images. We note that while we build on a particular approach to scenes [138] and objects [77], our approach is general.

While much of this reconstruction work is single-image, some is multiview, although usually in the case of an isolated object [124, 38, 106] or with hundreds of views [103]. Our work aims at the particular task of as little as two views, and reasons over multiple

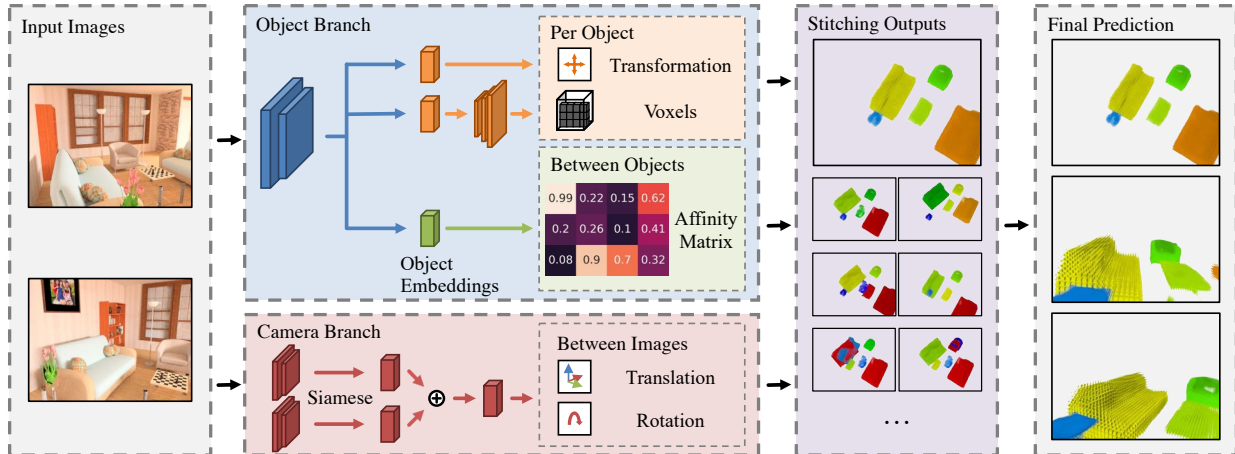


Figure 2.2: **Our approach.** We pass the two RGB image inputs into two branches that extract evidence, which is then fused together to stitch a final result. Our first network, **object branch**, is a detection network following [138] that produces a set of **objects** in terms of voxels and a transformation into the scene. We also predict an object embedding which we can use to form an **affinity matrix** between objects across images. Our second network, **camera branch**, is a siamese network that predicts a distribution over translations and rotations between the cameras. Finally our **stitching stage** examines the evidence from the networks and produces a final prediction.

objects. While traditional local features [173] are insufficient to support reasoning over objects, semantic features are useful [61, 272, 12].

At the same time, there has been considerable progress in identifying the relative pose from images [186, 126, 65, 8], RGB-D Scans [297, 298] or video sequences [321, 230, 255]. Of these, our work is most related to learning-based approaches to identifying relative pose from RGB images, and semantic Structure-from-Motion [8] and SLAM [230], which make use of semantic elements to improve the estimation of camera pose. We build upon this work in our approach, especially work like RpNet [65] that directly predicts relative pose, although we do so with a regression-by-classification formulation that provides uncertainty. As we show empirically, propagating this uncertainty forward lets us reason about objects and produce superior results to only focusing on pose.

2.3 Approach

The goal of the system is to map a pair of sparse views of a room to a full 3D reconstruction. As input, we assume a pair of images of a room. As output, we produce a set of objects represented as voxels, which are rigidly transformed and anisotropically scaled into the scene in a single coordinate frame. We achieve this with an approach, summarized in Fig. 2.2,

that consists of three main parts: an object branch, a camera branch, and a stitching stage.

The output space is a factored representation of a 3D scene, similar to [265, 138, 149]. Specifically, in contrast to using a single voxel-grid or mesh, the scene is represented as a set of per-object voxel-grids with a scale and pose that are placed in the scene. These can be converted to a single 3D reconstruction by taking their union, and so improving the 3D reconstruction can be done by either improving the per-object voxel grid or improving its placement in the scene.

The first two parts of our approach are two neural networks. An **object branch** examines each image and detects and produces single-view 3D reconstructions for objects in the camera’s coordinate frame, as well as a per-object embedding that helps find the object in the other image. At the same time, an **camera branch** predicts relative pose between images, represented as a *distribution* over a discrete set of rigid transformations between the cameras. These networks are trained separately to minimize complexity.

The final step, a **stitching stage**, combines these together. The output of the two networks gives: a collection of objects per image in the image’s coordinate frame; a cross-image affinity which predicts object correspondence in two views; and a set of likely transformations from one camera to the other. The stitching stage aims to select a final set of predictions minimizing an objective function that aims to ensure that similar objects are in the same location, the camera pose is likely, etc. Unlike the first two stages, this is an optimization rather than a feedforward network.

2.3.1 Object Branch

The goal of our object branch is to take an image and produce a set of reconstructed objects in the camera’s coordinate frame as well as an embedding that lets us match across views. We achieve this by extending 3D-RelNet [138] and adjust it as little as possible to ensure fair comparisons. We refer the reader for a fuller explanation in [138, 265], but briefly, these networks act akin to an object detector like Faster-RCNN [222] with additional outputs. As input, 3D-RelNet takes as input an image and a set of 2D bounding box proposals, and maps the image through convolutional layers to a feature map, from which it extracts per-bounding box convolutional features. These features pass through fully connected layers to predict: a detection score (to suppress bad proposals), voxels (to represent the object), and a transformation to the world frame (represented by rotation, scale, and translation and calculated via both per-object and pairwise poses). We extend this to also produce an n -dimensional embedding $\mathbf{e} \in \mathbb{R}^n$ on the unit sphere (i.e., $\|\mathbf{e}\|_2^2 = 1$) that helps associate objects across images.

We use and train the embedding by creating a cross-image affinity matrix between objects. Suppose the first and second images have N and M objects each with embeddings \mathbf{e}_i and \mathbf{e}'_j respectively. We then define our affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ as

$$\mathbf{A}_{i,j} = \sigma(k\mathbf{e}_i^T \mathbf{e}'_j) \quad (2.1)$$

where σ is the sigmoid/logistic function and where $k = 5$ scales the output. Ideally, $A_{i,j}$ should indicate whether objects i and j are the same object seen from a different view, where $A_{i,j}$ is high if this is true and low otherwise.

We train this embedding network using ground-truth bounding box proposals so that we can easily calculate a ground-truth affinity matrix $\hat{\mathbf{A}}$. We then minimize L_{aff} , a balanced mean-square loss between \mathbf{A} and $\hat{\mathbf{A}}$: if all positive labels are $(i, j) \in \mathcal{P}$, and all negative labels are $(i, j) \in \mathcal{N}$, then the loss is

$$L_{\text{aff}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} (A_{ij} - \hat{A}_{ij})^2 + \frac{1}{|\mathcal{N}|} \sum_{(i,j) \in \mathcal{N}} (A_{ij} - \hat{A}_{ij})^2. \quad (2.2)$$

which balances positive and negative labels (since affinity is imbalanced).

2.3.2 Camera Branch

Our camera branch aims to identify or narrow down the possible relationship between the two images. We approach this by building a siamese network [16] that predicts the relative camera pose T_c between the two images. We use ResNet-50 [95] to extract features from two input images. We concatenate the output features and then use two linear layers to predict the translation and rotation.

We formulate prediction of rotation and translation as a classification problem to help manage the uncertainty in the problem. We found that propagating uncertainty (via top predictions) was helpful: a single feedforward network suggests likely rotations and a subsequent stage can make a more detailed assessment in light of the object branch’s predictions. Additionally, even if we care about only one output, we found regression-by-classification to be helpful since the output tended to have multiple modes (e.g., being fairly certain of the rotation modulo 90° by recognizing that both images depict a cuboidal room). Regression tends to split the difference, producing predictions which satisfy neither mode, while classification picks one, as observed in [265, 142].

We cluster the rotation and translation vectors into 30 and 60 bins respectively, and predict two multinomial distributions over them. Then we minimize the cross entropy loss.

At test time, we select the cartesian product of the top 3 most likely bins for rotation and top 10 most likely bins for translation as the final prediction results. The results are treated as proposals in the next section.

2.3.3 Stitching Object and Camera Branches

Once we have run the object and camera branches, our goal is to then produce a single stitched result. As input to this step, our object branch gives: for view 1, with N objects, the voxels V_1, \dots, V_N and transformations T_1, \dots, T_N ; and similarly, for M objects in view 2, the voxels V'_1, \dots, V'_M and transformations T'_1, \dots, T'_M ; and a cross-view affinity matrix $\mathbf{A} \in [0, 1]^{N \times M}$. Additionally, we have a set of potential camera transformations P_1, \dots, P_F between two views.

The goal of this section is to integrate this evidence to find a final cross-camera pose P and correspondence $\mathbf{C} \in \{0, 1\}^{M \times N}$ from view 1 to view 2. This correspondence is one-to-one and has the option to ignore an object (i.e., $\mathbf{C}_{i,j} = 1$ if and only if i and j are in correspondence and for all i , $\sum_j \mathbf{C}_{i,j} \leq 1$, and similarly for \mathbf{C}^T).

We cast this as a minimization problem over P and \mathbf{C} including terms in the objective function that incorporate the above evidence. The cornerstone term is one that integrates all the evidence to examine the quality of the stitch, akin to trying and seeing how well things match up under a camera hypothesis. We implement this by computing the distance \mathcal{L}_D between corresponding object voxels according to \mathbf{C} , once the transformations are applied, or:

$$\mathcal{L}_D = \frac{1}{|\mathbf{C}|_1} \sum_{(i,j) \text{ s.t. } \mathbf{C}_{i,j}=1} D(P(T_i(V_i)), T'_j(V'_j)). \quad (2.3)$$

Here, D is the chamfer distance between points on the edges of each shape, as defined in [205, 238], or for two point clouds X and Y :

$$D(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2. \quad (2.4)$$

Additionally, we have terms that reward making \mathbf{C} likely according to our object and image networks, or: the sum of similarities between corresponding objects according to the affinity matrix \mathbf{A} , $\mathcal{L}_S = \sum_{(i,j), \mathbf{C}_{i,j}=1} (1 - A_{i,j})$; as well as the probability of the camera pose transformation P from the image network $\mathcal{L}_P = (1 - Pr(P))$. Finally, to preclude trivial solutions, we include a term rewarding minimizing the number of un-matched objects, or

$\mathcal{L}_U = \min(M, N) - |\mathbf{C}|_1$. In total, our objective function is the sum of these terms, or:

$$\min_{P, \mathbf{C}} \mathcal{L}_D + \lambda_P \mathcal{L}_P + \lambda_S \mathcal{L}_S + \lambda_U \mathcal{L}_U. \quad (2.5)$$

The search space is intractably large, so we optimize the objective function by RANSAC-like search over the top hypotheses for P and feasible object correspondences. For each top hypothesis of P , we randomly sample K object correspondence proposals. Here we use $K = 128$. It is generally sufficient since the correspondence between two objects is feasible only if the similarity of them is higher than a threshold according to the affinity matrix. We use random search over object correspondences because the search space increases factorially between the number of objects in correspondence. Once complete, we average the translation and scale, and randomly pick one rotation and shape from corresponding objects. Averaging performs poorly for rotation since there are typically multiple rotation modes that cannot be averaged: a symmetric table is correct at either 0° or 180° but not at 90° . Averaging voxel grids does not make sense since there are partially observable objects. We therefore pick one mode at random for rotation and shape.

2.3.4 Implementation Details

Detection proposals. We use more advanced object proposals compared to prior works [138, 265], which used edge boxes [324]. We found that edge boxes were often the limiting factor. Instead, we train a class-agnostic Faster-RCNN [222] to generate proposals, treating all objects as the foreground.

Object Branch. For each object, our object branch will predict a 300-dimensional vector, which represents its 3D properties. Linear layers are used to predict its shape embedding, translation, rotation, scale and object embedding. For the object embeddings, we use three linear layers. The size of outputs is 256, 128, 64, respectively. These linear layers predict a 64-dimensional embedding finally.

We train the object branch in two stages. In the first stage, we follow the training of 3D-RelNet [138] with ground truth bounding boxes. The loss of affinity matrix is ignored in this stage. In the second stage, we freeze all layers except the linear layers to predict the object embeddings. We only apply the affinity loss in this stage. For all two stages, we use Adam with learning rate $\varepsilon = 10^{-4}$ to optimize the model, with momentum 0.9. The batch size is 24. Although 3D-RelNet is finetuned on detection proposals, we only use the intermediate model trained with ground truth bounding box because (1) 3D-RelNet is finetuned on edgebox proposals and our Faster-RCNN proposals are good enough; (2) ground

truth affinity is only available with ground truth bounding box.

Camera Branch. The object and camera branches are trained independently. The translation is represented as 3D vectors, and the rotation is represented as quaternions. We run k-means clustering on the training set to produce 60 and 30 bins for translation and rotation. For rotation, we use spherical k-means to ensure the centroids are unit vectors.

The input image pairs are resized to 224x224. They are passed through a siamese network with ResNet-50 pre-trained on ImageNet [95] as the backbone. The outputs from each instance of the siamese network are concatenated, and passed through a linear layer, producing a 128-dimensional vector. The vector is then passed through a translation branch and a rotation branch. Each branch is a linear layer which outputs 60 and 30 dimensional vectors for translation and rotation bins.

Our loss function is the cross entropy loss. The loss for the translation prediction and the rotation prediction are weighted equally. We use stochastic gradient descent with learning rate $\varepsilon = 10^{-3}$ and momentum 0.9. The batch size is 32. We also augment the data by reversing the order of image pairs.

Tuning the stitching stage. The search space contains top-3 rotation, top-10 translation and top-128 object correspondence hypotheses. The threshold of affinity is 0.5. λ_P , λ_S , and λ_U are tuned as hyperparameters on the validation set to preclude the trivial solution. We use $\lambda_S = 5$, $\lambda_U = 1$. For λ_P , we use 5 for rotation and 1 for translation.

2.4 Experiments

We now describe a set of experiments that aim to address the following questions: (1) how well does the proposed method work and are there simpler approaches that would solve the problem? and (2) how does the method solve the problem? We first address question (1) by evaluating the proposed approach compared to alternate approaches both qualitatively and by evaluating the full reconstruction quantitatively. We then address question (2) by evaluating individual components of the system. We focus on what the affinity matrix learns and whether the stitching stage can jointly improve object correspondence and relative camera pose estimation. Throughout, we test our approach on the SUNCG dataset [250, 314], following previous work [265, 138, 149, 297, 314]. To demonstrate transfer to other data, we also show qualitative results on NYUv2 [245].

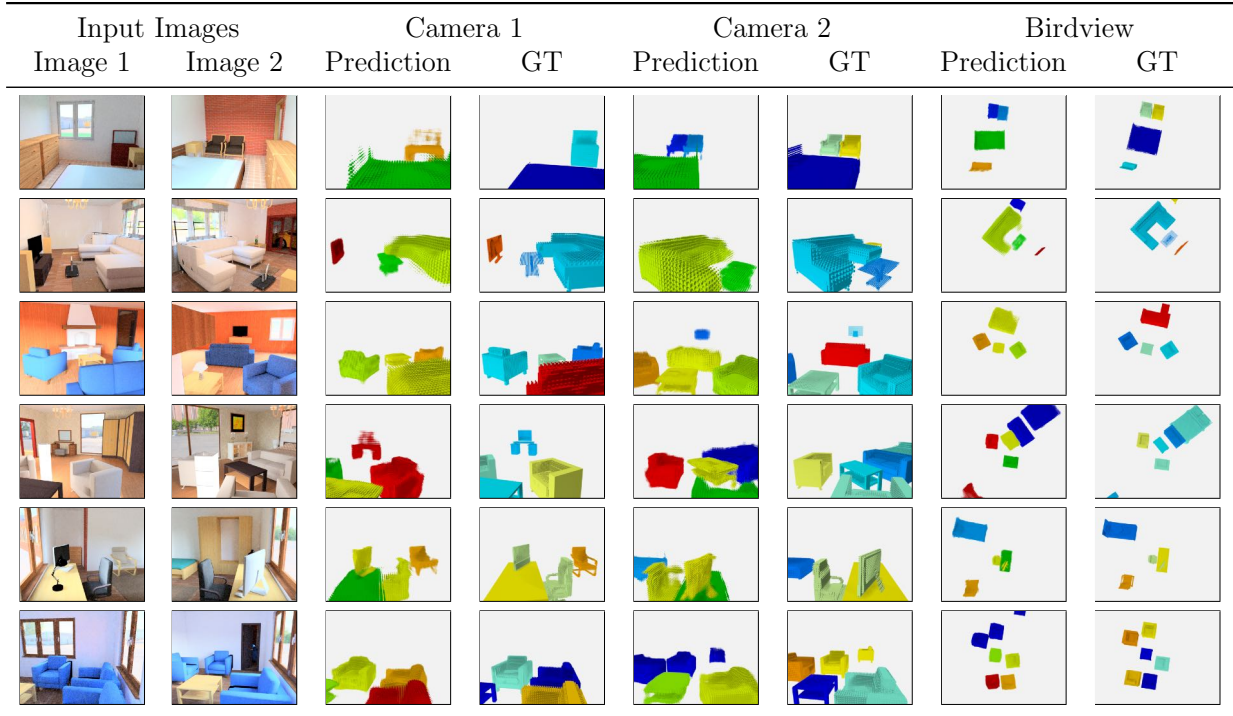


Figure 2.3: Qualitative results on the SUNCG test set [250]. The final 3D predictions are shown in three different camera poses (1) the same camera as image 1; (2) the same camera as image 2; (3) a bird view to see all the objects in the whole scene. In the prediction, red/orange objects are from the left image, blue objects are from the right image, green/yellow objects are stitched.

2.4.1 Experimental Setup

Following common practice in this field [265, 138, 149, 297, 314], We train and do extensive evaluation on SUNCG [250] since it provides 3D scene ground truth including voxel representation of objects. There are realistic datasets such as ScanNet [42] and Matterport3D [26], but they only provide non-watertight meshes. Producing filled object voxel representation from non-watertight meshes remains an open problem. For example, Pix3D [257] aligns IKEA furniture models with images, but not all objects are labeled. and there are no multiple views Therefore, we decide to train and evaluate our approach on the SUNCG dataset [250].

Datasets. We follow the 70%/10%/20% training, validation and test split of houses from [138]. For each house, we randomly sample up to ten rooms; for each room, we randomly sample one pair of views. Furthermore, we filter the validation and test set: we eliminate pairs where there is no overlapping object between views, and pairs in which all of one image’s objects are in the other view (i.e., one is a proper subset of the other). We do

not filter the training set since learning relative pose requires a large and diverse training set. Overall, we have 247532/1970/2964 image pairs for training, validation and testing, respectively. Following [265], we use six object classes - bed, chair, desk, sofa, table and tv.

Full-Scene Evaluation: Our output is a full-scene reconstruction, represented as a set of per-object voxel grids that are posed and scaled in the scene. A scene prediction can be totally wrong if one of the objects has correct shape while its translation is off by 2 meters. Therefore, we quantify performance by treating the problem as a 3D detection problem in which we predict a series of 3D boxes and voxel grids. This lets us evaluate which aspect of the problem currently hold methods back. Similar to [138], for each object, we define error metrics as follows:

- Translation (**t**): Euclidean distance, or $\delta_t = \|t - \hat{t}\|_2$, thresholded at $\delta_t = 1\text{m}$.
- Scale (**s**): Average log difference in scaling factors, or $\delta_s = \frac{1}{3} \sum_{i=1}^3 |\log_2(s_1^i) - \log_2(s_2^i)|$, thresholded at $\delta_s = 0.2$.
- Rotation (**R**): Geodesic rotation distance, or $\delta_q = (2)^{-1/2} \|\log(\mathbf{R}^T \hat{\mathbf{R}})\|_F$, thresholded at $\delta_q = 30^\circ$.
- Shape (**V**): Following [260], we use F-score@0.05 to measure the difference between prediction and ground truth, thresholded at $\delta_V = 0.25$.

A prediction is a true positive only if all errors are lower than our thresholds. We calculate the precision-recall curve based on that and report average precision (AP). We also report AP for each single error metric.

Baselines. Since there is no prior work on this task, our experiments compare to ablations and alternate forms of our method. We use the following baseline methods, each of which tests a concrete hypothesis.

(Feedforward): This method uses the object branch to recover single-view 3D scenes, and our camera branch to estimate the relative pose between different views. We ignore the affinity matrix and pick the top-1 relative pose predicted by the camera branch. There can be many duplicate objects in the output of this approach. This tests if a simple feedforward method is sufficient.

(NMS): In addition to the feedforward approach, we perform non-maximum suppression on the final predictions. If two objects are close to each other, we merge them. This tests if a simple policy to merge objects would work.

(Raw Affinity): Here, we use the predicted affinity matrix to merge objects based on top-1 similarity from the affinity matrix. This tests whether our stitching stage is necessary.

(Associative3D): This is our complete method. We optimize the objective function by searching possible rotations, translations and object correspondence.

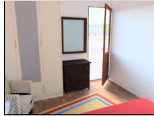

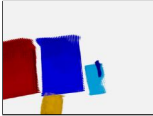
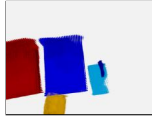





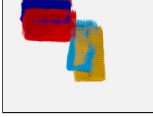
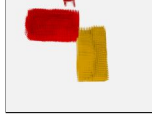
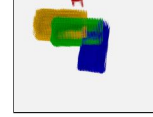
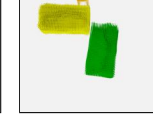
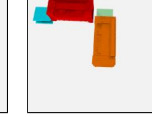


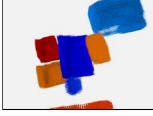
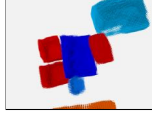
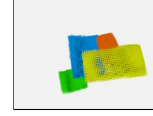

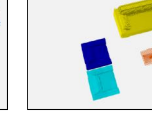
Image 1	Image 2	Feedforward	NMS	Raw Affinity	Ours	GT
						
						
						

Figure 2.4: Comparison between Associative3D and alternative approaches. **Row 1:** Associative3D fixes the incorrect top-1 relative camera pose in light of a single bed in the room. **Row 2:** NMS works when the relative camera pose is accurate. **Row 3:** Associative3D outperforms all alternative approaches in finding correspondence in object clutter.

2.4.2 Full Scene Evaluation

We begin by evaluating our full scene reconstruction. Our output is a set of per-object voxels that are posed and scaled in the scene. The quality of reconstruction of a single object is decided by both the voxel grids and the object pose.

First, we show qualitative examples from the proposed method in Fig. 2.3 as well as a comparison with alternate approaches in Fig. A.4 on the SUNCG test set. The Feedforward approach tends to have duplicate objects since it does not know object correspondence. However, figuring out the camera pose and common objects is a non-trivial task. Raw Affinity does not work since it may merge objects based on their similarity, regardless of possible global conflicts. NMS works when the relative camera pose is accurate but cannot work when many objects are close to each other. Instead, Associative3D demonstrates the ability to jointly reason over reconstructions, object pose and camera pose to produce a reasonable explanation of the scene. More qualitative examples are available in the supplementary material.

We then evaluate our proposed approach quantitatively. In a factored representation [265], both *object* poses and shapes are equally important to the full *scene* reconstruction. For instance, the voxel reconstruction of a scene may have no overlap if all the shapes are right, but they are in the wrong place. Therefore, we formulate it as a 3D detection problem, as a prediction is a true positive only if all of translation, scale, rotation and shape are correct. However, 3D detection is a very strict metric. If the whole scene is slightly off in one aspect, we may have a very low AP. But the predicted scene may still be reasonable. We mainly use

Table 2.1: We report the average precision (AP) in evaluation of the 3D detection setting. **All** means a prediction is a true positive only if all of translation, scale, rotation and shape are correct. **Shape**, **Translation**, **Rotation**, and **Scale** mean a prediction is a true positive when a single error is lower than thresholds. We include results on the whole test set, and top 25%, 50% and 75% examples ranked by single-view predictions.

Methods	All Examples					Top 25%	Top 50%	Top 75%
	All	Shape	Trans	Rot	Scale	All	All	All
Feedforward	21.2	22.5	31.7	28.5	26.9	41.6	34.6	28.6
NMS	21.1	23.5	31.9	29.0	27.2	42.0	34.7	28.7
Raw Affinity	15.0	24.4	26.3	28.2	25.9	28.6	23.5	18.9
Associative3D	23.3	24.5	38.4	29.5	27.3	48.3	38.8	31.4

it quantify our performance.

Table 2.1 shows our performance compared with all three baseline methods. Our approach outperforms all of them, which verifies what we see in the qualitative examples. Moreover, the improvement mainly comes from that on translation. The translation-only AP is around 7 points better than Feedforward. Meanwhile, the improvement of NMS over Feedforward is limited. As we see in qualitative examples, it cannot work when many objects are close to each other. Finally, raw affinity is even worse than Feedforward, since raw affinity may merge objects incorrectly. We will discuss why the affinity is informative, but top-1 similarity is not a good choice in Sec. 2.4.3.

We notice our performance gain over Feedforward and NMS is especially large when single-view predictions are reasonable. On top 25% examples which single-view prediction does a good job, Associative3D outperforms Feedforward and NMS by over 6 points. On top 50% examples, the improvement is around 4 points. It is still significant but slightly lower than that of top 25% examples. When single-view prediction is bad, our performance gain is limited since Associative3D is built upon it. We will discuss this in Sec. 2.4.5 as failure cases.

We also compare with single-view baselines to show whether multi-view helps. We take a prediction from 3D-RelNet [138] on one view of the pair, selected randomly. On the whole test set, the AP is 13.7, which significantly underperforms all approaches. It shows our proposed approach has significant improvement built upon single-view baselines, and multi-view helps reconstruct the scene.

Table 2.2: AUROC and rank correlation between the affinity matrix and *category*, *model*, *shape*, and *instance*, respectively. **Model | Category** means the ability of the affinity matrix to distinguish different models given the same category / semantic label.

	Category	Model Category	Shape Category	Instance Model
AUROC	0.92	0.73	-	0.59
Correlation	0.72	0.33	0.34	0.14

2.4.3 Inter-view Object Affinity Matrix

We then turn to evaluating how the method works by analyzing individual components. We start with the affinity matrix and study what it learns.

We have three non-mutually exclusive hypotheses: (1) **Semantic labels.** The affinity is essentially doing object recognition. After detecting the category of the object, it simply matches objects with the same category. (2) **Object shapes.** The affinity matches objects with similar shapes since it is constructed from the embedding vectors which are also used to generate shape voxels and the object pose. (3) **Correspondence.** Ideally, the affinity matrix should give us ground truth correspondence. It is challenging given duplicate objects in the scene. For example, people can have three identical chairs in their office. These hypotheses are three different levels the affinity matrix may learn, but they are not in conflict. Learning semantic labels do not mean the affinity does not learn anything about shapes.

We study this by examining a large number of pairs of objects and testing the relationship between affinity and known relationships (e.g., categories, model ids) using ground truth bounding boxes. We specifically construct three binary labels (same *category*, same *model*, same *instance*) and a continuous label shape similarity (namely F-score @ 0.05 [260]). When we evaluate shape similarity, we condition on the category to test if affinity distinguishes between different models of the same category, (e.g. chair). Similarly, we condition on the model when we evaluate instance similarity.

We compute two metrics: a binary classification metric that treats the affinity as a predictor of the label as well as a correlation that tests if a monotonic relationship exists between the affinity and the label. For binary classification, we use AUROC to evaluate the performance since it is invariant to class imbalance and has a natural interpretation. For correlation, we compute Spearman’s rank correlation coefficient [325] between the affinity predictors and labels. This tests how well the relationship between affinity and each label (e.g., shape overlap) fits a monotonic function (1 is perfect agreement, 0 no agreement).

The results are shown in Table 2.2. Both the binary classification and the rank correlation show that the affinity matrix is able to distinguish different categories and objects of different

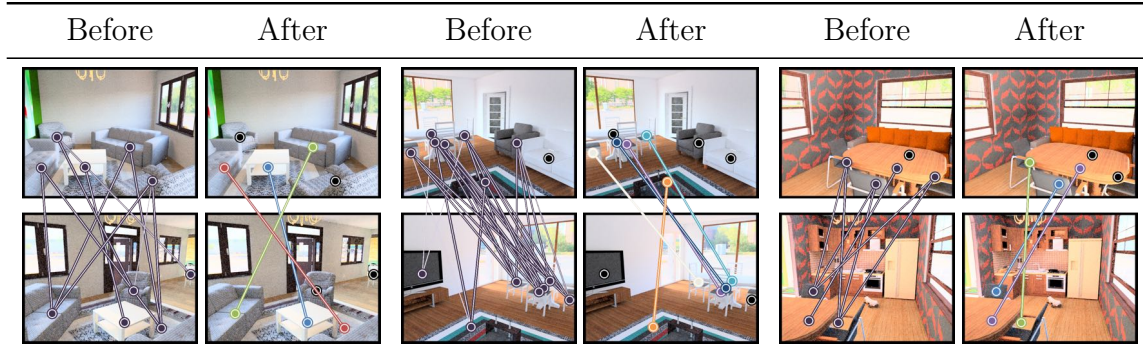


Figure 2.5: Visualization of the stitching stage. The affinity matrix generates proposals of corresponding objects, and then the stitching stage removes outliers by inferring the most likely explanation of the scene.

shapes, but is sub-optimal in distinguishing the same instance. These results justify our stitching stage, which addresses the problem based on joint reasoning. It also explains why Raw Affinity underperforms all other baselines by a large margin in the full-scene evaluation. Additionally, the ability to distinguish categories and shapes provides important guidance to the stitching stage. For example, a sofa and bed are similar in 3D shapes. It is infeasible to distinguish them by simply looking at the chamfer distance, which can be distinguished by the affinity matrix.

2.4.4 Stitching Stage

We evaluate the stitching stage by studying two questions: (1) How well can it predict object correspondence? (2) Can it improve relative camera pose estimation? For example, if the top-1 relative pose is incorrect, could the stitching stage fix it by considering common objects in two views?

Object Correspondence. To answer the first question, we begin with qualitative examples in Fig. A.5, which illustrate object correspondence before and after the stitching stage. Before our stitching stage, our affinity matrix has generated correspondence proposals based on their similarity. However, there are outliers since the affinity is sub-optimal in distinguishing the same instance. The stitching stage removes these outliers.

We evaluate object correspondence in the same setting as Sec 2.4.3. Suppose the first and second images have N and M objects respectively. We then have $N \times M$ pairs. The pair is a positive example if and only if they are corresponding. We use average precision (AP) to measure the performance since AP pays more attention to the low recall [49, 66]. For i^{th} object in view 1 and j^{th} object in view 2, we produce a confidence score by γA_{ij}

Table 2.3: Evaluation of object correspondence with and without the stitching stage.

	All Negative	Affinity	Affinity Top1	Associative3D
AP	10.1	38.8	49.4	60.0

Table 2.4: Evaluation of relative camera pose from the camera branch and picked by the stitching.

Method	Translation (meters)			Rotation (degrees)		
	Median	Mean	(Err \leq 1m)%	Median	Mean	(Err \leq 30°)%
Top-1	1.24	1.80	41.26	6.96	29.90	77.56
Associative3D	0.88	1.44	54.89	6.97	29.02	78.31

where $\gamma = 1$ if the pair is predicted to be corresponding and $\gamma = 0.5$ otherwise. This γ term updates the confidence based on stitching stage to penalize pairs which have a high affinity score but are not corresponding.

We compare Associative3D with 3 baselines. (**All Negative**): The prediction is always negative (the most frequent label). This serves as a lower bound. (**Affinity**): This simply uses the affinity matrix as the confidence. (**Affinity Top1**): Rather than using the raw affinity matrix, it uses affinity top-1 similarity as the correspondence and the same strategy to decide confidence as Associative3D. Table 2.3 shows that our stitching stage improves AP by 10% compared to using the affinity matrix only as correspondence.

Relative Camera Pose Estimation. We next evaluate the performance of relative camera pose (i.e., camera translation and rotation) estimation and see if the stitching stage improves the relative camera pose jointly. We compare the camera pose picked by the stitching stage and top-1 camera pose predicted by the camera branch. We follow the rotation and translation metrics in our full-scene evaluation to measure the error of our predicted camera poses. We summarize results in Table 2.4. There is a substantial improvement in translations, with the percentage of camera poses within 1m of the ground truth being boosted from 41.3% to 54.9%. The improvement in rotation is smaller and we believe this is because the network already starts out working well and can exploit the fact that scenes tend to have three orthogonal directions. In conclusion, the stitching stage can mainly improve the prediction of camera translation.

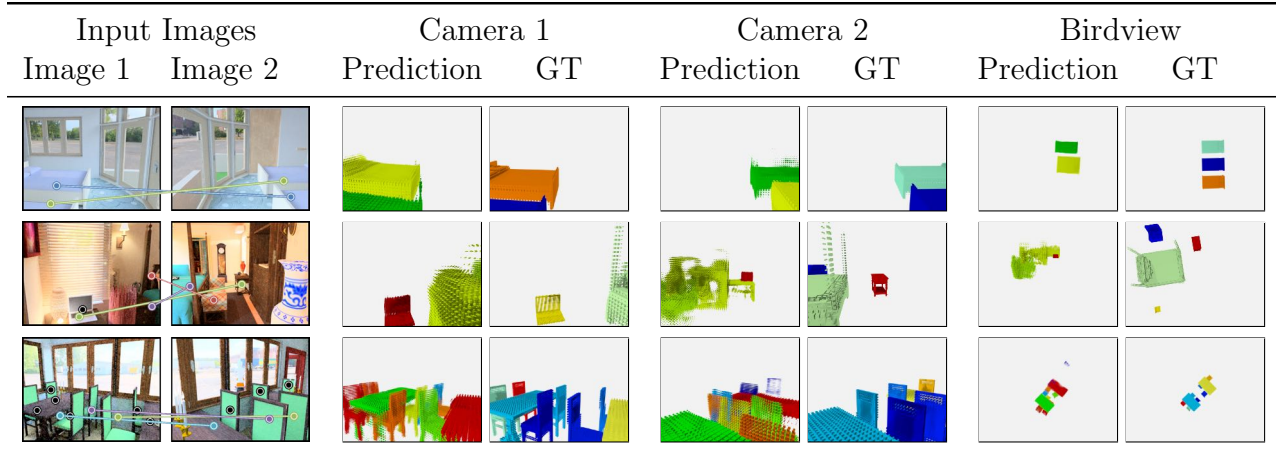


Figure 2.6: Representative failure cases on the SUNCG test set [250]. **Row 1:** The input images are ambiguous. There can be two or three beds in the scene. **Row 2:** The single-view backbone does not produce a reasonable prediction. **Row 3:** This is challenging because all chairs are the same.

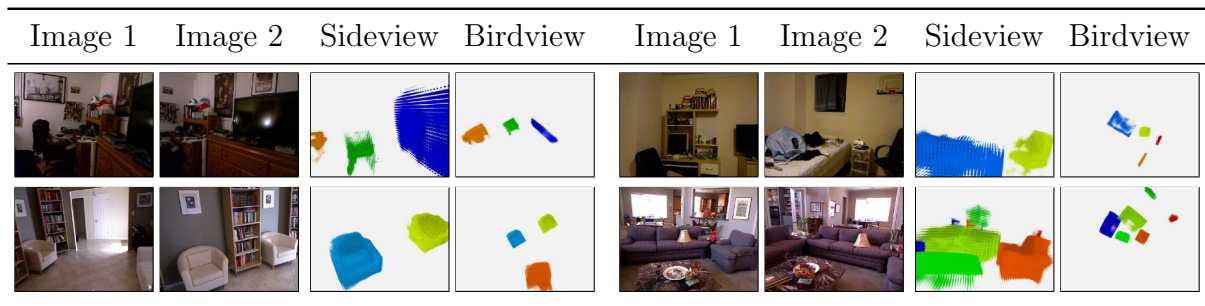


Figure 2.7: Qualitative results on NYUv2 dataset [245]. Sideview corresponds to the camera view slightly transformed from the image 2 camera position.

2.4.5 Failure Cases

To understand the problem of reconstruction from sparse views better, we identify some representative failure cases and show them in Fig. 2.6. While our method is able to generate reasonable results on SUNCG, it cannot solve some common failure cases: (1) The image pair is ambiguous. (2) The single-view backbone does not produce reasonable predictions as we discuss in Sec. 2.4.2. (3) There are too many similar objects in the scene. The affinity matrix is then not able to distinguish them since it is sub-optimal in distinguishing the same instance. Our stitching stage is also limited by the random search over object correspondence. Due to factorial growth of search space, we cannot search all possible correspondences. The balancing of our sub-losses can also be sensitive.

2.4.6 Results on NYU Dataset

To test generalization, we also test our approach on images from NYUv2 [245]. Our only change is using proposals from Faster-RCNN [222] trained on COCO [157], since Faster-RCNN trained on SUNCG cannot generalize to NYUv2 well. We do not finetune any models and show qualitative results in Fig. 2.7. Despite training on synthetic data, our model can often obtain a reasonable interpretation.

2.5 Conclusion

We have presented Associative3D, which explores 3D volumetric reconstruction from sparse views. While the output is reasonable, failure modes indicate the problem is challenging to current techniques. Directions for future work include joint learning of object affinity and relative camera pose, and extending the approach to many views and more natural datasets other than SUNCG.

CHAPTER 3

3D Semantic Segmentation from Novel Viewpoints

Humans can understand scenes in 3D from a few views. In AI, the ability to recognize a scene from any viewpoint given only a few images would enable agents to interact with the scene and drive exciting applications in AR and VR. In this work, we attempt to endow machines with this ability. We propose a model which takes as input a few RGB images of a new scene and produces the 3D semantic segmentation from novel viewpoints without access to the RGB images from those views. We pair 2D scene recognition with an implicit 3D representation and learn from multi-view 2D annotations of hundreds of scenes without 3D supervision beyond camera poses. Our experiments on challenging datasets demonstrate our model’s ability to jointly capture semantics and geometry of novel scenes with diverse layouts, objects and shapes. The material in this chapter is derived from [212].

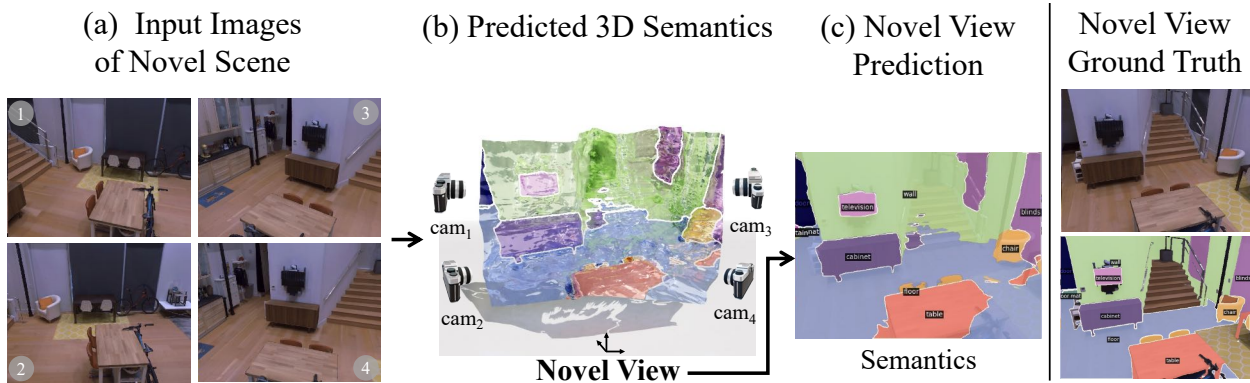


Figure 3.1: We propose ViewSeg which takes as input (a) a few images of a novel, previously unobserved scene, and recognizes the scene from novel viewpoints. The novel viewpoint, in the form of camera coordinates, queries (b) our predicted 3D representation to produce (c) semantic segmentations from the view *without access to the view’s RGB image*. ViewSeg trains on hundreds of scenes using multi-view 2D annotations and no 3D supervision.

3.1 Introduction

Humans can build a rich understanding of scenes from a handful of images. Consider the scene in Fig. 3.1. From only 4 views we can understand there is a table with chairs on one side and a bike on the other, a kitchen counter is at one end and a window opposite to it, *etc.* One outcome of our understanding is the ability to predict what objects we see and where when we turn our head in any direction, even when our eyes are closed. This skill allows us to navigate in the dark and is so intuitive that industries like hotels and real estate depend on persuading users with just a few photos. For machines, it is still a challenge to have this level of understanding from a few images of a novel scene.

In this work, we explore semantic 3D scene recognition and formally define the problem of *novel view semantic understanding (NVSU)*. In NVSU, the task is to predict the scene’s semantic maps from target views different than the input. The input is one or few posed RGB images of a *previously unobserved scene* and the output is pixel-level semantic segmentation from a new and unseen target viewpoint. NVSU differs from the task of single-view semantic segmentation as the target view’s RGB image is not provided; only the RGB images of the input views are given. NVSU also differs from novel view synthesis (NVS) as it shifts the focus from appearance to semantics. In essence, NVSU evaluates 3D understanding and semantic recognition – a model needs to understand the scene and its objects from a different viewpoint – without 3D semantic ground truth.

We propose to tackle NVSU by predicting a queryable 3D semantic representation of the scene, inspired by NeRFs [188], from just a few posed RGB views. A novel view, in the form of camera coordinates, queries the 3D representation to produce a semantic segmentation of the scene from that view *without access to the novel view’s RGB image*. Our method, called ViewSeg, is shown in Fig. 3.1. Unlike Semantic-NeRF [316] which interpolates between dense views of the same scene, we do this from sparse input views of *novel scenes*. To achieve this, we learn from hundreds of diverse scenes, using multi-view 2D annotations and no 3D supervision other than camera poses.

Scientifically, NVSU connects three key areas of computer vision: semantic understanding (i.e., naming objects), 3D understanding, and novel view synthesis (NVS). This puts the problem out of reach of work in any one area. Advances in NVS [188, 280, 305] perform well but focus only on appearance. Using appearance as a bottleneck representation can cause issues for semantic recognition: one may be sure of the presence of a rug behind the sofa but unsure of its color. As shown in Fig. 3.2, RGB synthesis is often blurry or collapses when extrapolating from few views. Similarly, there have been advances in learning to infer 3D scene properties from image cues [265, 198, 79], or with differentiable rendering [125, 166,



Figure 3.2: RGB is a poor bottleneck representation for semantic synthesis. While one may be certain of the labels of the new view (e.g., walls, tables), synthesizing the particular wall or table pixels from few views is challenging: outputs are often blurry (top) or collapse when extrapolating (bottom).

33, 219, 20] and other ways to bypass the need for direct 3D supervision [140, 123, 137, 300]. However, these works mainly focus on a small number of objects and do not scale to complex scenes. More importantly, we empirically show that merely using learned 3D to propagate scene semantics to the new view is insufficient.

ViewSeg fuses advances from semantic segmentation, 3D understanding, and NVS with task-specific modifications to solve NVSU. Our experiments validate ViewSeg’s contributions on two challenging datasets, Hypersim [225] and Replica [253], which contain complex scenes with diverse layouts, object types, and shapes. We substantially outperform approaches to NVSU that build on the state-of-the-art: image-based NVS [305] followed by semantic segmentation [30], which tests whether NVS is sufficient for the task; and lifting semantic segmentations [30] to 3D and differentially rendering which compares explicit 3D learning with ViewSeg’s implicit representation. Our ablations reveal the impact of semantic reconstruction on geometry, and show semantics lead to more accurate depth, which is produced by ViewSeg “for free”. Our results demonstrate ViewSeg’s ability to jointly capture semantics and geometry when tested on novel diverse scenes and we hope that our analysis will inspire more future work in this direction.

3.2 Related Work

We draw from 2D recognition, novel view synthesis and 3D learning to recognize scenes from novel viewpoints.

Semantic Segmentation. Segmenting objects and stuff (*i.e.* wall, floor, ceiling) from images is extensively researched. Initial efforts apply Bayesian classifiers on local features [134] or perform grouping on low-level cues [240]. Others [24, 43] score bottom-up mask proposals [2, 25]. With the advent of deep learning, FCNs [170] perform per-pixel segmentation by training a CNN to classify each pixel. DeepLab [29] use atrous convolutions and an encoder-decoder architecture [30] to handle scale and resolution.

Regarding multi-view semantic segmentation, [141] improve the temporal consistency of semantic segmentation in videos by linking frames with optical flow and learned feature similarity. [185] maps semantic segmentations from RGBD inputs on 3D reconstructions from SLAM. [96] fuse predictions from video frames using super-pixels and optical flow. [174] learns scene dynamics to predict semantic segmentations of future frames given several past frames.

Novel View Synthesis. Novel view synthesis is a popular topic in computer vision and graphics. [70, 251, 252, 266, 291, 322] synthesize new views from two or more narrow baseline images. Implicit voxel representations have been used to fit a scene from many views [169, 241, 248]. NeRF [188] learn a continuous volumetric scene function which emits density and radiance at spatial locations and show impressive results when fitted on a single scene with hundreds of views. We extend NeRF to emit a distribution over semantic categories at each 3D location. Semantic-NeRF [316] and NeSF [268] also predict semantic classes but we differ in two critical ways: (a) at test time we generalize to novel, previously unobserved scenes instead of in-place interpolation within a single scene, and (b) we require sparse (up to 4) instead of hundreds of input views. NeRF extensions [97, 221], such as PixelNeRF [305], generalize to novel scenes from few views and show results on single-object benchmarks for RGB synthesis. We differ from [305] in two critical ways: (a) we tackle semantic understanding instead of RGB synthesis and (b) we experiment on realistic multi-object scenes instead of simplistic single-object benchmarks. We show that RGB synthesis cannot address the task of semantic understanding. Real multi-object scenes make the task harder, but they bring us closer to real-world applications.

3D Reconstruction from Images. Scene reconstruction from many views is traditionally tackled with classical binocular stereo [90, 232] or with the help of shape priors [9, 11, 45, 107]. Modern techniques learn disparity from image pairs [127], estimate correspondences with contrastive learning [233], perform multi-view stereopsis via differentiable ray projection [124],

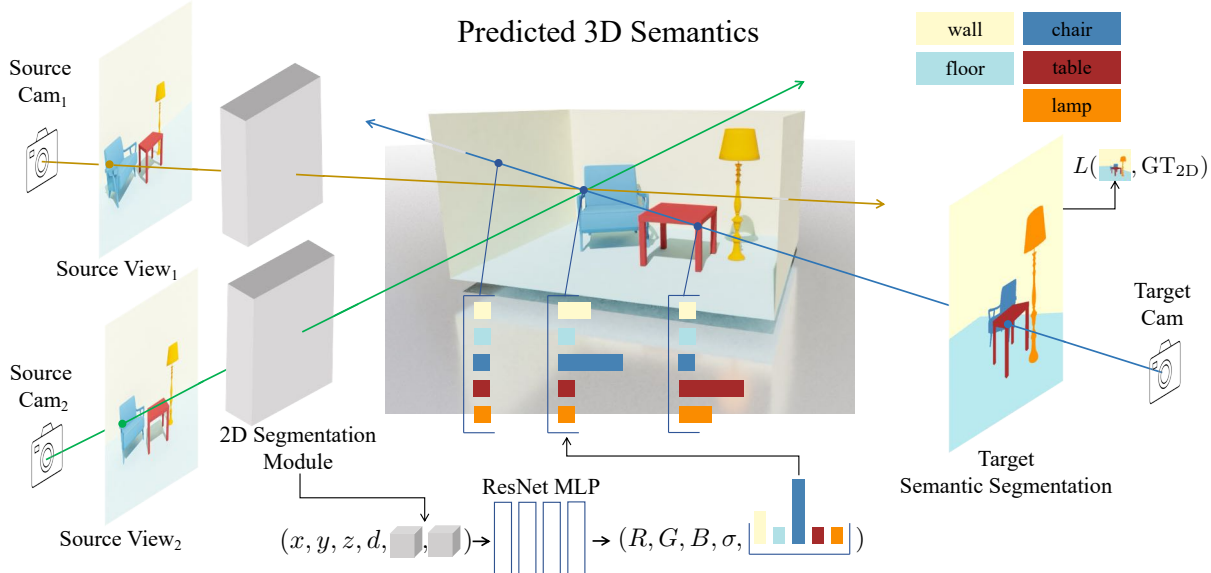


Figure 3.3: Our model, ViewSeg, uses a few source RGB views (here 2) of a *novel scene* and predicts the semantic segmentation of that scene from a novel target viewpoint. Our approach embeds each source view into a latent semantic space with a 2D segmentation module; this space is used to predict radiance, density and distribution over semantic classes at each spatial 3D location via an MLP. The final semantic segmentation is created by volumetric rendering from the target viewpoint. We generalize to unseen scenes by training on hundreds of diverse scenes and thousands of source-target pairs, and no 3D semantic supervision.

or learn to reconstruct scenes while optimizing for cameras [210, 120]. Differentiable rendering lets gradients flow to 3D via 2D re-projections [171, 125, 166, 33, 150, 199, 219] and reconstructs single objects [125, 166, 33, 219] or complex scenes [80] from one view via rendering from more views during training. We use differentiable rendering to learn 3D via re-projections in semantic space.

Depth Estimation from Images. Commonly, single-view depth is learned from videos [321, 180, 33] or 3D supervision [63, 32, 153, 303, 218]. We do not use depth supervision but predict depth via training for semantic reconstruction.

3.3 Approach

We tackle novel view semantic understanding end-to-end with a new model, ViewSeg. ViewSeg takes as input RGB images from N *source* views and segments objects and stuff from a novel *target* viewpoint without access to the target image. We pair semantic segmentation with an implicit 3D representation to learn semantics and geometry from hundreds of scenes

and thousands of source-target pairs. An overview of our approach is shown in Fig. 3.3.

Our model consists of a 2D semantic segmentation module which embeds each source view to a higher dimensional semantic space. An implicit 3D representation [188] samples features from the output of the segmentation module and predicts radiance, density and a distribution over semantic categories at each spatial 3D location with an MLP. The 3D predictions are projected to the target view to produce the segmentation from that view via volumetric rendering.

3.3.1 Semantic Segmentation Module

The role of the semantic segmentation module is to project each source RGB view to a learnt feature space, which is subsequently used to make 3D semantic predictions. Our 2D segmentation backbone takes as input an image I and outputs a spatial map M of the same spatial resolution, $b^{\text{seg}} : I^{H \times W \times 3} \rightarrow M^{H \times W \times K}$, using a convolutional neural network (CNN). Here, K is the dimension of the feature space ($K = 256$).

We build on the state-of-the-art for single-view semantic segmentation and follow the encoder-decoder DeepLabv3+ [30, 29] architecture which processes an image by downsampling and subsequently upsampling it to its original resolution with a sequence of convolutions. We remove the final layer, which predicts class probabilities, and use the output from the penultimate layer.

We initialize our segmentation module by pre-training on ADE20k [318], a dataset of over 20k images with semantic annotations. We empirically show the impact of the network architecture and pre-training in our experiments.

3.3.2 Semantic 3D Representation

To recognize a scene from novel viewpoints, we learn a 3D representation which predicts the semantic class for each 3D location. To achieve this we learn a function f which maps semantic features from our segmentation module to distributions over semantic categories conditioned on the 3D coordinates of each spatial location.

Assume N source views $\{I_j\}_{j=1}^N$ and corresponding cameras $\{\pi_j\}_{j=1}^N$. For each view we extract the semantic maps $\{M_j\}_{j=1}^N$ with our 2D segmentation module, or $M_j = b^{\text{seg}}(I_j)$. We project every 3D point \mathbf{x} to the j -th view with the corresponding camera transformation, $\pi_j(\mathbf{x})$, and then sample the K -dimensional feature vector from M_j from the projected 2D location. This yields a semantic feature from the j -th image denoted as $\phi_j^{\text{seg}}(\mathbf{x}) = M_j(\pi_j(\mathbf{x}))$.

Following NeRF [188] and PixelNeRF [305], f takes as input a positional embedding of the 3D coordinates of \mathbf{x} , $\gamma(\mathbf{x})$, and the viewing direction \mathbf{d} . We additionally feed the

semantic embeddings $\{\phi_j^{\text{seg}}\}_{j=1}^N$. As output, f produces

$$(\mathbf{c}, \sigma, \mathbf{s}) = f(\gamma(\mathbf{x}), \mathbf{d}, \phi_0^{\text{seg}}(\mathbf{x}), \dots, \phi_{N-1}^{\text{seg}}(\mathbf{x})) \quad (3.1)$$

where $\mathbf{c} \in \mathbb{R}^3$ is the RGB color, $\sigma \in \mathbb{R}$ is the occupancy, and $\mathbf{s} \in \mathbb{R}^{|\mathcal{C}|}$ is the distribution over semantic classes \mathcal{C} .

We model f as a fully-connected ResNet [95], similar to PixelNeRF [305]. The positional embedding of the 3D location and the direction are concatenated to each semantic embedding ϕ_j^{seg} , and each is passed through 3 residual blocks with 512 hidden units. The outputs are subsequently aggregated using average pooling and used to predict the final outputs of f via two branches: one predicts the semantics \mathbf{s} , the other the color \mathbf{c} and density σ . Each branch consists of two residual blocks, each with 512 hidden units. The detailed network architecture is in the Supplementary.

Predicting Semantics. Rendering the semantic predictions, \mathbf{s} , from a given viewpoint gives the semantic segmentation of the scene from that view. Following NeRF [188], we accumulate predictions on rays, $\mathbf{r}(t) = \mathbf{o} + t \cdot \mathbf{d}$, originating at the camera center \mathbf{o} with direction \mathbf{d} ,

$$\hat{S}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathbf{s}(t)dt \quad (3.2)$$

where $T(t) = \exp(-\int_{t_n}^t \sigma(s)ds)$ is the accumulated transmittance along the ray, and t_n and t_f are near and far sampling bounds, which are hyperparameters.

The values of (t_n, t_f) are crucial for good performance. In NeRF, they are manually set to tightly bound the scene. PixelNeRF selects them manually *for each object type*, *i.e.* different values for chair, car, *etc.* In Semantic-NeRF [316], the values are selected for the Replica rooms, which vary little in size. In the datasets we experiment on (Sec. 3.4), scene scale varies drastically from human living spaces with regular depth extents (*e.g.* living rooms) to industrial facilities (*e.g.* warehouses), lofts or churches with large far fields. Aiming for scene generalization, we set (t_n, t_f) globally, regardless of the true near/far bounds of each scene. This more realistic setting makes the problem harder: our model needs to predict the right density values for a large range of depth fields, reasoning about occupancy within each scene but also about the depth extent of the scene as a whole.

Replacing the semantic predictions \mathbf{s} with the RGB predictions \mathbf{c} in Eq. 3.2 produces the RGB view $\hat{C}(\mathbf{r})$ from the target viewpoint, as in [188]. While photometric reconstruction is not our goal, we use \hat{C} during learning and show that it helps capture the scene’s geometry more accurately.

Predicting Depth. In addition to the semantic segmentation \hat{S} and the RGB reconstruc-

tion \hat{C} , we also predict the pixel-wise depth of the scene from a target viewpoint, as in [53]

$$\hat{D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)tdt. \quad (3.3)$$

We use the depth output \hat{D} only for evaluation. By comparing single-view depth (\hat{D}) and semantic segmentation (\hat{S}) from many novel viewpoints, we measure our model’s ability to capture geometry and semantics.

3.3.3 Learning Objective

Our primary objective is to segment the scene from the target view. We jointly train the segmentation module b^{seg} and implicit 3D function f to directly solve this task. We also find that auxiliary photometric and source-view losses are crucial for performance. Our objectives require RGB images and 2D semantics from various views in the scene as well as poses to perform re-projection.

Since our goal is to predict a semantic segmentation in the target view, our primary objective is a per-pixel cross-entropy loss between the true class labels S and predicted class distribution \hat{S} ,

$$L_{\text{target}}^S = - \sum_{\mathbf{r} \in \mathbf{R}} \sum_{j=1}^{|\mathcal{C}|} S^j(\mathbf{r}) \log \hat{S}^j(\mathbf{r}) \quad (3.4)$$

where \mathcal{C} is the set of semantic classes and \mathbf{R} is the set of rays in the target view. Here, $S^j(\mathbf{r})$ is the $\{0, 1\}$ true label for the j -th class at the intersection of ray \mathbf{r} and the image.

In addition to this, we minimize auxiliary losses that improve performance on our primary task. The first is a photometric loss on RGB images, namely the squared L_2 distance between the prediction \hat{C} and actual image C , or

$$L_{\text{target}}^P = \sum_{\mathbf{r} \in \mathbf{R}} \left\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \quad (3.5)$$

where $C(\mathbf{r})$ is the true RGB color at the intersection of ray \mathbf{r} and the image.

Finally, in addition to standard losses on the target view [188, 305, 316], we find it is important to apply losses on the source views. Specifically, we create L_{source}^S and L_{source}^P that are the semantic and photometric losses, respectively, applied to rays from the source views. These losses help enforce consistency with the input views.

Our final objective is given by

$$L = L_{\text{target}}^S + L_{\text{source}}^S + \lambda \cdot (L_{\text{target}}^P + L_{\text{source}}^P) \quad (3.6)$$

where λ scales the semantic and photometric losses.

3.4 Experiments

We experiment on Hypersim [225] and Replica [253]. Both provide posed views of complex scenes with over 30 object types and under varying conditions of occlusion and lighting. At test time, we evaluate on novel scenes not seen during training. Due to its large size, we treat Hypersim as our main dataset where we run an extensive quantitative analysis. We then show generalization to the smaller Replica, which contains real-world indoor scenes.

Metrics. We report novel view metrics for semantics and geometry to evaluate our model’s ability to capture both. For a novel view of a test scene, we project the semantic predictions (Eq. 3.2) and depth (Eq. 3.3) and compare to the ground truth semantic and depth maps, respectively. Ideally, we would also evaluate directly in 3D, which requires access to full 3D ground truth. However, 3D ground truth is not publicly available for Hypersim and is generally hard to collect. Thus, we treat novel view metrics as proxy metrics for 3D semantic segmentation and depth estimation.

For semantic comparisons, we report semantic segmentation metrics [19] implemented in Detectron2 [284]: **mIoU** is the intersection-over-union (IoU) averaged across classes, **IoU^T** and **IoU^S** report IoU by merging all things (object) and stuff classes (wall, floor, ceiling), respectively. **fwIoU** is the per class IoU weighted by the pixel-level frequency of each class, **pACC** is the percentage of correctly labelled pixels and **mACC** is the pixel accuracy averaged across classes. For all, performance is in % and higher is better.

For depth comparisons, we report depth metrics following [64]: **L₁** is the per-pixel average L₁ distance between ground truth and predicted depth, **Rel** is the L₁ distance normalized by the true depth value, **Rel^T** and **Rel^S** is the Rel metric for all things and stuff, respectively. **$\delta < \tau$** is the percentage of pixels with predicted depth within $[\frac{1}{\tau}, \tau] \times$ the true depth. L₁ is in meters and $\delta < \tau$ is in %. For $\delta < \tau$ metrics, higher is better. For all other, lower is better (\downarrow).

Model	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (\downarrow)	Rel (\downarrow)	Rel ^T (\downarrow)	Rel ^S (\downarrow)	$\delta < 1.25$	$\delta < 1.25^2$
PixelNeRF+SS	1.6	14.9	47.9	17.9	35.9	3.6	2.80	0.746	0.856	0.653	0.300	0.531
PixelNeRF++	4.4	28.8	43.2	25.7	37.6	7.5	2.80	0.746	0.856	0.653	0.300	0.531
CloudSeg	0.5	29.6	4.4	1.8	3.2	3.3	3.81	0.856	0.997	0.737	0.145	0.277
ViewSeg (ours)	17.1	33.2	58.9	44.8	62.2	23.9	2.29	0.646	0.721	0.584	0.409	0.656
Oracle	40.0	58.1	71.3	66.6	79.1	52.1	0.96	0.235	0.317	0.163	0.731	0.898

Table 3.1: Comparisons on Hypersim val. We report the performance for semantic segmentation (blue) and depth estimation (green) for our method, ViewSeg, an oracle which applies supervised single-image semantic segmentation and depth estimation models on the true target RGB views, two NVS variants based on PixelNeRF [305] and an explicit 3D point cloud model, CloudSeg, inspired by SynSin [280].

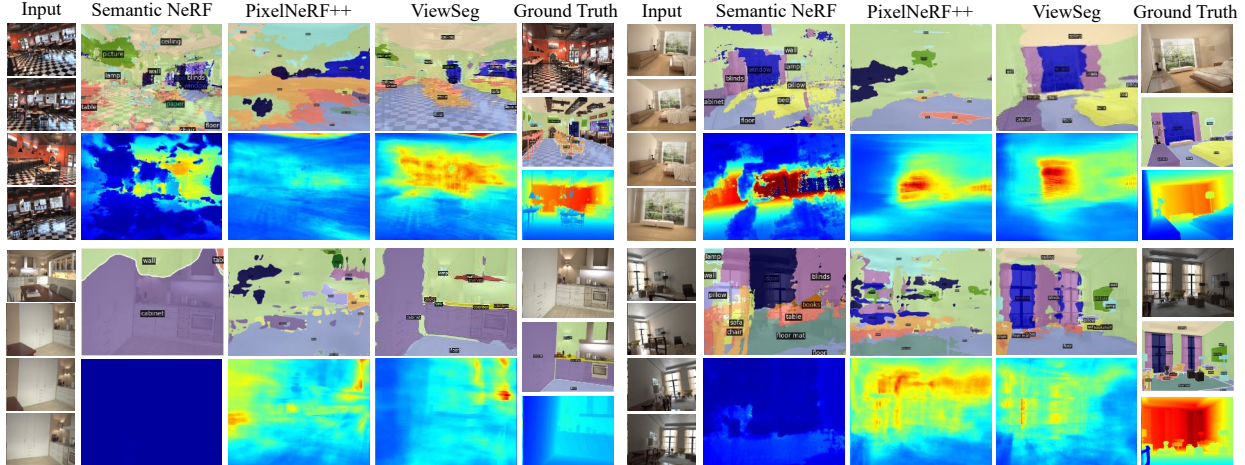


Figure 3.4: Predictions on Hypersim. For each example, we show the 4 input RGB views of a novel scene, the semantic and depth predictions from Semantic NeRF, PixelNeRF++ and our ViewSeg. We also show the true RGB, semantic and depth map of the target view. Our model does not have access to ground truth observations from the target view at test time. Depth colormap (scaled per example): min max.

3.4.1 Experiments on Hypersim

Hypersim [225] is a dataset of 461 complex scenes. Camera trajectories across scenes result in 77,400 images with camera poses, masks for 40 semantic classes [245], along with true depth maps. Hypersim contains on average 50 objects per image, and is a challenging dataset for object recognition despite it being synthetic. For reference, Mask R-CNN [94] achieves 17% AP for 2D object detection.

Dataset. For each scene, we create source-target pairs from the available views. An image is labelled as target and is paired with another image from a different viewpoint if: (1) the view frustums intersect by no less than 10%; (2) the camera translation is greater than 0.5m; and (3) the camera rotation is at least 30° . This ensures that source and target views are from different camera viewpoints and broadly depict the same parts of the scene but without large overlap. We follow the original Hypersim split, which splits train/val/test to 365/46/50 disjoint scenes, respectively. Overall, there are 120k/14k/14k pairs in train/val/test splits, respectively.

Training details. We implement ViewSeg in PyTorch with Detectron2 [284] and PyTorch3D [219]. We train on the Hypersim training set for 13 epochs with a batch size of 32 across 32 Tesla V100 GPUs. The input and render resolution are set to 1024×768 , maintaining the size of the original dataset. We optimize with Adam [129] and a learning rate of $5e-4$. We follow the PixelNeRF [305] strategy for ray sampling: We sample 64 points per ray

in the coarse pass and 128 points per ray in the fine pass. In addition to the target view, we randomly sample rays on each source view and additionally minimize the source view loss, as we describe in Sec. 3.3.3. We set $t_n = 0.1\text{m}$, $t_f = 20\text{m}$ in Eq. 3.2 and 3.3. More details are in the Supplementary.

Baselines. In addition to extensive ablations that reveal which components of our method are most important, we compare with competitive baselines and oracle methods. Our baselines test alternate strategies, including inferring the target view RGB image and then predicting per-pixel semantic classes as well as lifting a predicted semantic segmentation map to 3D and re-projecting to the target view.

To provide context, we report a **Target View Oracle** that has access to the true target view image. The target RGB image fundamentally resolves many ambiguities in 3D about what is where, and is not available to our method. Instead, our method is tasked with predicting segmentations and depth from new viewpoints *without the target RGB images*. Our oracle applies appropriate supervised models directly on the true target RGB. For semantic segmentation, we use a model [30] that is identical to ours pre-trained on ADE20k [318] and finetuned on all images of the Hypersim training set. For depth, we use the model from [303], which predicts normalized depth. We obtain metric depth by aligning with the optimal shift and scale following [218, 303].

We then compare to PixelNeRF [305] using a two-stage approach: it performs novel view synthesis (NVS) to infer the target RGB and then applies image-based semantic segmentation. This comparison tests the importance of our method’s end-to-end nature. For a fair result, PixelNeRF is trained on the Hypersim training set and we compare to two variants: *PixelNeRF+SS* uses the segmentation model of the oracle, and *PixelNeRF++* trains a segmentation model on PixelNeRF’s predicted RGB views on the training set. Depth is produced by Eq. 3.3.

Our final baseline, named **CloudSeg**, tests the importance of an implicit representation by comparing with an explicit 3D point cloud representation. Inspired by SynSin [280], we train a semantic segmentation backbone similar to our ViewSeg, along with a depth model, from [303], to lift each source view to a 3D point cloud with per point class probabilities. A differentiable point cloud renderer [219] projects the point clouds from the source images to the target view to produce a semantic and a depth map. CloudSeg is trained on the Hypersim training set and uses the same 2D supervision as our ViewSeg.

Results. Table 3.1 compares ViewSeg, PixelNeRF variants and CloudSeg with 4 source views and the oracle on Hypersim val. PixelNeRF++ performs slightly better than PixelNeRF+SS, and both perform significantly worse than ViewSeg. This is due to the poor RGB predictions (see Fig. 3.2 & Supplementary). Note that our method’s goal is *not* to predict

ViewSeg loss	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta < 1.25^2$
w/o photometric loss	16.9	30.8	58.7	44.8	62.5	22.8	2.49	0.677	0.750	0.615	0.359	0.611
w/o semantic loss	-	-	-	-	-	-	2.58	0.787	0.919	0.678	0.345	0.587
w/o source view loss	14.3	28.2	57.9	28.2	61.1	19.3	2.37	0.683	0.764	0.615	0.397	0.649
w/o viewing dir	16.0	33.1	59.2	44.9	62.1	21.5	2.53	0.708	0.783	0.646	0.354	0.602
final	17.1	33.2	58.9	44.8	62.2	23.9	2.29	0.646	0.721	0.584	0.409	0.656

Table 3.2: Ablating loss components. We report semantic (blue) and depth (green) performance on Hypersim val without the photometric, the semantic and the source view loss and when excluding the viewing direction from the input. Our model is reported in the last row.

ViewSeg backbone	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$
DLv3+ [30] + ADE20k [318]	17.1	33.2	58.9	44.8	62.2	23.9	2.29	0.645	0.721	0.584	0.409
DLv3+ [30] + IN [52]	16.3	33.2	59.2	45.2	62.5	22.0	2.28	0.614	0.682	0.559	0.415
ResNet34 [95] + IN [52]	7.5	21.7	55.9	37.1	56.1	11.2	2.67	0.712	0.815	0.626	0.320

Table 3.3: Performance on Hypersim val with different semantic segmentation backbones. We show the performance of ViewSeg with DeepLabv3+ (DLv3+) [30] pretrained on ADE20k [318] and on ImageNet [52] and a ResNet34 [95] backbone pretrained on ImageNet [52]. The latter is used in PixelNeRF [305]. DLv3+ improves performance significantly, while ADE20k helps ever so slightly.

RGB; the opposite, we bypass it and go directly to semantics. Predicting high fidelity RGB of novel complex scenes from just four wide baselines views is hard, and a testament to the difficulty of NVS rather than the specifics of the models. This is also revealed in the original PixelNeRF paper [305] where RGB synthesis significantly degrades when applied to more complex scenes compared to single-object simplistic benchmarks, even for narrow baselines. Hypersim scenes are far more complex and with wide baselines making it even harder to predict accurate RGB. Our experiments support that a two-stage solution performs worse than ViewSeg, showing the importance of learning semantics end-to-end and bypassing the hard RGB synthesis step. ViewSeg also outperforms PixelNeRF for depth, suggesting that learning for semantics has a positive impact on geometry as well. Finally, CloudSeg, inspired by [280], has a hard time predicting semantics and geometry. We believe it is because CloudSeg learns depth from scratch. Only few points are projected into the target view when the baseline is wide and the depth is inaccurate. Therefore, the supervisory signal is limited. Note that in [280], the views are narrow, compared with our large viewpoint change (shown in Fig. 3.4). Our representation is continuous and allows us to extrapolate from the input even in unseen areas of the scene.

Fig. 3.4 shows qualitative results on Hypersim val for diverse scenes (restaurant, bedroom, kitchen, living room, loft) with many objects (chair, table, counter, cabinet, window, blinds, lamp, picture, floor mat, *etc.*). We show the 4 input views and the semantic and depth predictions by Semantic-NeRF [316], PixelNeRF++ and our ViewSeg; we also show the ground truth target RGB, semantic and depth maps for reference. Semantic-NeRF optimizes a model for each example from the corresponding 4 input semantic maps. Semantic-NeRF struggles to reconstruct the scene from only 4 views. Like NeRF, more (≥ 100) input views

ViewSeg	mIoU	pACC	L_1 (\downarrow)	Rel (\downarrow)
w/ 4 views	17.1	62.2	2.29	0.584
w/ 3 views	15.5	61.5	2.39	0.652
w/ 2 views	13.6	60.2	2.57	0.765
w/ 1 view	11.6	57.9	2.62	0.734

Table 3.4: Input study on Hypersim val for varying number of source views. More views improve semantic and depth performance.

might be necessary to get good results. On the other hand, ViewSeg detects stuff (floor, wall, ceiling) well, predicts object segments for the right object types more effectively than PixelNeRF++ and produces more accurate depth maps. Fig. 3.4 also reveals ViewSeg’s limitations. Small objects and precise boundary segmentations are hard to estimate well. In addition, object segments are sometimes shifted due to the model’s inability to predict their accurate location in 3D. These limitations leave lots of potential for future work.

Ablations and Input Study. Table 3.2 ablates various terms in our objective. For reference, the performance of our ViewSeg trained with 4 source views and with the final objective (Eq. 6.4) is shown in the last row. When we remove the photometric loss, L^P , semantic performance remains roughly the same but depth performance drops (-20cm in L_1), which proves that appearance helps capture scene geometry. When we remove the semantic loss, L^S , and train solely with a photometric loss, we observe a drop in depth (-29cm in L_1). This suggests that semantics helps geometry; we made a similar observation when comparing to PixelNeRF in Table 3.1. When training without source view losses both semantic and depth performance drop, with semantic performance deteriorating the most (-2.8% in mIoU). This confirms our insight that enforcing consistency with the source views improves learning. Finally, when we remove the viewing direction from the model’s input, depth performance suffers the most (-24cm in L_1).

Table 3.3 compares different backbones for the 2D segmentation module. We compare DeepLabv3+ (DLv3+) [30] pre-trained on ImageNet [52] and ADE20k [318] and ResNet34 [95] pre-trained on ImageNet [52]. The latter is used in PixelNeRF [305]. DLv3+ significantly boosts performance for both semantics and depth while pre-training on ADE20k slightly adds to the final performance.

Table 3.4 compares ViewSeg with varying number of source views on Hypersim val. More views improve both semantic segmentation and depth. More than 4 views could lead to further improvements but substantially increase memory and time requirements during training. Fig. 3.7 shows 3D semantic reconstructions from 4 scene views.



Figure 3.5: We test on real images taken with an iPhone 12. Despite noisy camera poses estimated by COLMAP [234] following [189], ViewSeg obtains reasonable segmentation results.

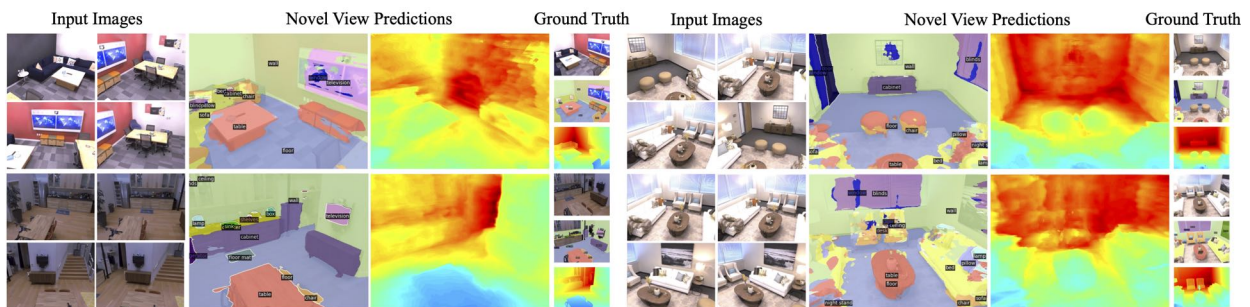



Figure 3.6: Predictions on Replica. For each example, we show the 4 input views (left), ground truth RGB, semantic and depth from the novel view (middle) and ViewSeg’s semantic and depth predictions (right). Depth colormap (scaled per example): min  max.

3.4.2 Generalization to Replica and iPhone Examples

We experiment on the Replica dataset [253] which contains real-world scenes such as living rooms and offices. Scenes are complex with many objects in various layouts. We show generalization by applying ViewSeg pre-trained on Hypersim and then further fine-tune it on Replica to better fit to Replica’s statistics. We further demonstrate ViewSeg’s generalization by showing predictions on novel scenes from images captured with an iPhone in Fig. 3.5.

Replica. We use AI-Habitat [231] to extract multiple views per scene. For each view we collect the RGB image, semantic labels and depth. For each Replica scene, we simulate an agent circling around the center of the 3D scene and render the observations. Note that this is unlike Hypersim [225], where camera trajectories are extracted by the authors a-priori. We use the same camera intrinsics and resolution as Hypersim: the horizontal field of view is 60° and the image resolution is 1024×768 . Finally, we map the 88 semantic classes from

Model	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta < 1.25^2$
PixelNeRF++	25.5	38.4	53.9	48.1	64.8	37.2	0.680	0.160	0.161	0.158	0.734	0.968
ViewSeg noft	13.2	44.8	56.0	51.4	66.8	27.1	0.982	0.222	0.194	0.254	0.623	0.880
ViewSeg	30.2	56.2	62.8	62.3	75.6	48.4	0.550	0.130	0.130	0.130	0.851	0.961
Oracle	56.2	76.8	78.0	90.1	93.8	79.4	0.226	0.058	0.065	0.050	0.976	0.998

Table 3.5: Performance for semantic segmentation (blue) and depth (green) on the Replica [253] test set before finetuning (noft) and after finetuning ViewSeg on Replica’s training set. We additionally report the target view oracle.

Replica to NYUv3-13, following [316, 42]. Our dataset consists of 12/3 scenes for train/test, respectively, resulting in 360/90 source-target pairs. Note that this is 330× smaller than Hypersim. Despite the small dataset size, we show compelling results on Replica via pre-training.

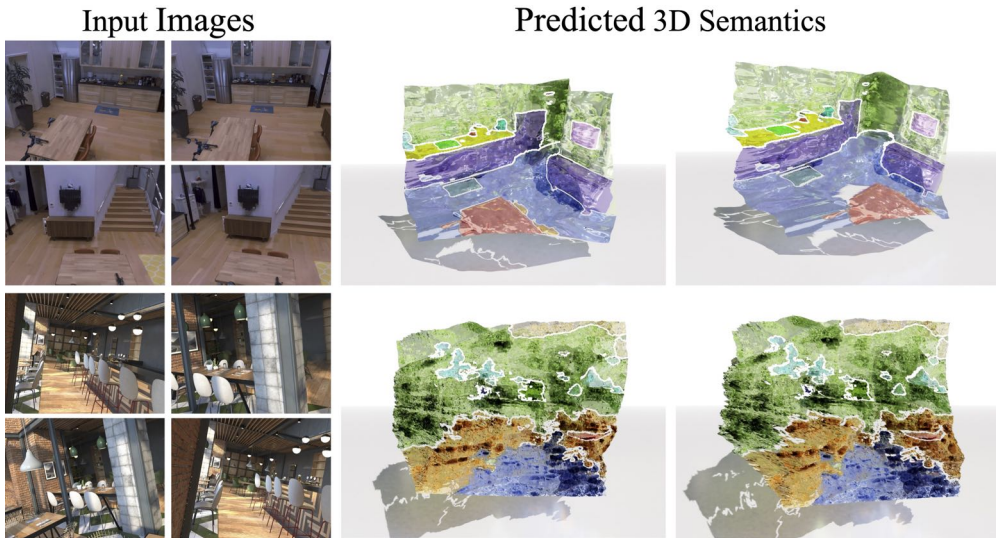


Figure 3.7: 3D semantic reconstructions of novel scenes.

Results. Table 3.5 reports the performance of our ViewSeg, trained on Hypersim, after and before fine-tuning (denoted as noft). PixelNeRF++ finetunes two models: one that perform NVS and followed by semantic segmentation on predicted RGB views. The oracle fine-tunes the supervised semantic segmentation model on images from the Replica dataset and finds the optimal depth scale and shift for the test set. We observe that ViewSeg’s performance improves after fine-tuning on Replica compared to no-finetuning for both semantic segmentation and depth across all metrics; this is not surprising as the scenes across the two datasets vary in both object appearance and geometry. We also observe that performance is much higher than Hypersim (Table 3.1), which is expected as Hypersim is more diverse and challenging. ViewSeg outperforms PixelNeRF++.

Fig. 3.6 shows ViewSeg predictions on Replica. We show the input images, ViewSeg’s predictions and the ground truth RGB, semantic and depth map for the target view.

3.5 Conclusion

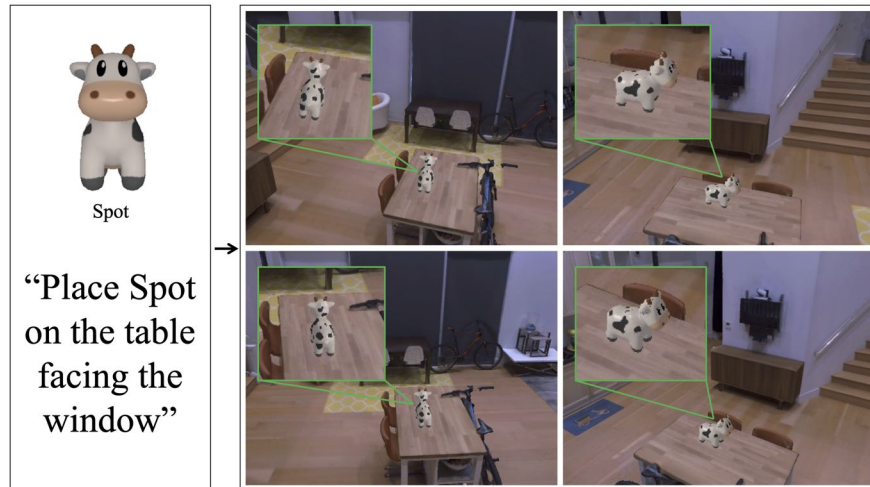


Figure 3.8: “Place-in-scene” AR demo with ViewSeg.

Inspired by the ability of humans to recognize 3D scenes from few input views, we introduce the task of novel view semantic understanding (NVSU). We argue that the task of recognizing scenes from novel views in semantic space is conceptually related to novel view synthesis in RGB space, but need not and should not rely on it. ViewSeg bypasses RGB synthesis and predicts in semantic space, directly attacking the recognition problem.

NVSU enables exciting AI applications in AR. For instance, you might want to see how an object looks in your living space in Fig. 3.1 by “placing the object on the table facing the window”. AR needs to localize the table, the window with regards to the table and subsequently place the object in 3D, all from few RGB images. Our model attempts to address these non-trivial 3D semantic understanding tasks. Fig. 3.8 shows a demo which augments the scene of Fig. 3.1 using ViewSeg’s 3D semantic prediction and without any further training or manual intervention to the scene.

We do not foresee immediate ethical risks from our work. Our datasets do not contain humans or any other sensitive information.

CHAPTER 4

Understanding 3D Object Articulation in Internet Videos

In this chapter, we propose to investigate detecting and characterizing the 3D planar articulation of objects from ordinary videos. While seemingly easy for humans, this problem poses many challenges for computers. We propose to approach this problem by combining a top-down detection system that finds planes that can be articulated along with an optimization approach that solves for a 3D plane that can explain a sequence of observed articulations. We show that this system can be trained on a combination of videos and 3D scan datasets. When tested on a dataset of challenging Internet videos and the Charades dataset, our approach obtains strong performance. The material in this chapter is derived from [211].

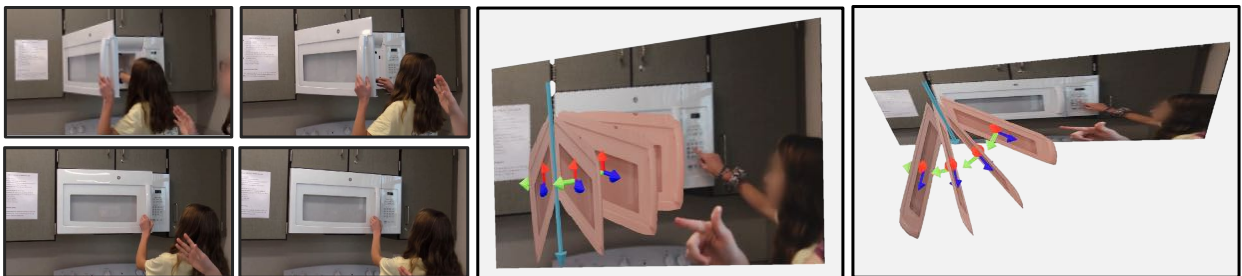


Figure 4.1: Given an ordinary video, our system produces a 3D planar representation of the observed articulation. The 3D renderings illustrate how the microwave (in **Pink**) can be articulated in 3D space. We also show the predicted rotation axis using a **Blue** arrow.

4.1 Introduction

How would you make sense of Figure 4.1? Behind the set of RGB pixels that make up the video is a real 3D transformation consisting of a 3D planar door rotating about an axis. The goal of this chapter is to give the same ability to computers. We focus on planar articulation taking the form of a rotation or translation along an axis. This special case of articulation is ubiquitous in human scenes and understanding it lets a system understand objects ranging from refrigerators and drawers to closets and cabinets. While we often learn about these shapes and articulations with physical embodiment [249], we have no difficulty understanding them from video cues alone, for instance while watching a movie or seeing another person perform an action. We formalize this ability for computers as recognizing and characterizing a class-agnostic planar articulation via a 3D planar segment, articulation type (rotation or translation), 3D articulation axis, and articulation angle.

This problem is beyond the current state of the art in scene understanding since it requires reconciling single image 3D understanding with *dynamic 3D* understanding. While there has been substantial work on 3D reconstruction from a single image [63, 274, 39, 77], including work dedicated to planes [159], these works focus on reconstructing static scenes. On the other hand, while there has been work understanding articulation, these works often require the placement of tags for tracking [203, 168], a complete 3D model or depth sensor [151, 194, 112], or successful 3D human reconstruction [289]. Moreover, making progress is challenging because of data. Unsupervised approaches based on motion analysis [254, 204] require something to track, which breaks in realistic data since many human-made articulated objects are untextured (e.g., refrigerators) or transparent (e.g., ovens). While supervised approaches [194, 193, 151] can perhaps bypass tracking features, they seemingly require access to large amounts of RGBD data of interactions. For now, this data does not exist, and training on synthetic data can fall short when tested on real data (as our experiments empirically demonstrate).

We overcome these challenges with a learning-based approach that combines both detection and 3D optimization and is trained with supervision from multiple sources (Section 4.4). The foundation of our approach is a top-down detection approach that recognizes articulation axes and types and 3D planes; this approach’s outputs are processed with an optimization method that seeks to explain the per-frame results in terms of a single coherent 3D articulation.

Via this model, we show that one can build an understanding of 3D object dynamics via a mix of 2D supervision on Internet videos of objects undergoing articulation as well as 3D supervision on existing 3D datasets that do not depict articulations. To provide 2D

supervision, we introduce (Section 4.3) a new set of 9447 Creative Commons Internet videos. These videos depict articulation with a variety of objects as well as negative samples and come with sparse frame annotations of articulation boxes, axes, and surface normals that can be used for training and evaluating planar articulation models.

Our experiments (Section 4.5) evaluate how well our approach can recognize and characterize articulation. We evaluate on our new dataset of videos as well as the Charades [244] dataset. We compare with a variety of alternate approaches, including bottom-up signals like optical flow [261] and changes in surface normals [34], training on synthetic data [287], as well as systems that analyze human-object interaction [289]. Our approach outperforms these approaches on our data, often even when the baselines are given access to ground-truth location of articulation.

Our primary contributions include: (1) The new task of detecting 3D object articulation on unconstrained ordinary RGB videos without requiring RGBD video at training time; (2) A dataset of Internet videos, with sparse frame annotations of articulation boxes, axes, and surface normals that can be used for training and evaluating planar articulation models; (3) A top-down detection network and optimization to tackle this problem, which has strong performance on the Internet video dataset and Charades.

4.2 Related Work

This chapter proposes to extract 3D models of articulation from ordinary RGB videos. This problem lies at the intersection of 3D vision, learning from videos, and touches on robotics applications. We note that there are specialized approaches for understanding *general* articulation (e.g., non-rigid structure from motion [263]) as well as for understanding specialized motion models (e.g., for a full human 3D mesh models [311] or quadrupeds [137]) or for understanding more general transformations [110, 273]. Our work focuses on understanding the articulation of general objects whose articulated pieces can be represented by a 3D plane rotating or translating.

Due to the ubiquitous nature of articulated objects, the task of understanding them has long been an interest across all of artificial intelligence. In vision, the understanding of the motion of rigid objects undergoing transformations was one of the early successes of computer vision [262, 122, 270]. Unfortunately, these early works rely on reliable motion tracks, which is made difficult by the textureless or reflective nature of many indoor planes (e.g., refrigerator doors). Our top-down detector gives 3D planes that can help provide correspondence between frames where correspondence is challenging.

More recent work in robotics has used the value of 3D and integrated it into their modeling

approaches [254, 204, 40, 55, 187]; however their approaches often use an RGBD sensor, unlike our use of ordinary RGB sensors. This dependence on RGBD has been carried forward to the most recent work that uses deep learning frameworks [287, 10, 151, 164, 112, 275]. In fact, some methods require full 3D models [194], which is typically unavailable in real world 3D scans. [193] by Mo et al. can be run on 2D images as long as the point cloud encoder is replaced with a RGB encoder, but its 2D images contain a single object without any background, instead of challenging Internet videos. While there has been increasing amounts of work aimed at virtual articulated objects [258, 287], simultaneously achieving scale and quality is challenging. For instance ReplicaCAD [258] has only 92 objects. In contrast, our approach works at test time on standard RGB videos by bringing its own 3D via a learned detector [159] trained on RGBD data [42].

While our outputs are 3D planar regions, our approach is deeply connected to the task of understanding human-object interactions. In these works [27, 78, 236], the goal is to recognize the relationship between humans and the objects they interact with. The interactions that we study are caused by these humans, and so we use an approach that can predict human-object interactions [236] to help identify the data we train our systems on. The most related work in this area is [289], which aims to jointly understand dynamic 3D human-object interactions in 3D. This work, however assumes that the object CAD model is known once the articulated object is detected, which we do not need. Our method also works with articulation videos that are more varied in viewpoint and perspective. A more thorough understanding of the joint relationship between articulated objects and human-object interaction, akin to early work [135, 71], is beyond the scope of this work, but of future value.

We solve the problem of describing 3D articulation by producing 3D planar models. This uses advances in 3D from a single image. In particular, we build on PlaneRCNN [159], which is part of a growing body of works aimed at extracting planes from single images [294, 307, 161]. These planes have advantages for the articulation reasoning since they offer a compact representation to track and describe. While we use plane recognition, the plane is just one component of our output (along with rotation axes) and we analyze our output in a video with temporal optimization.

4.3 Data Collection

One critical component of our approach is accurate 2D annotations of articulation occurring in RGB data. We show that these 2D annotations can be combined with existing RGBD data and the right method to build systems that understand 3D articulation on video data. We next describe how we collect a dataset of articulations. Our goals are to

have a large collection of annotations of object box, articulation type, and axis. Rather than directly look for examples of people articulating objects, we follow the data-first approach of [72, 310, 236, 46], namely to gather data containing many related activities and then analyze and annotate it.

Data Collection. Our pipeline generates a set of candidate clips to be annotated from a collection of candidate videos via an automatic pipeline that aims to eliminate frames that are easy to see as not depicting articulation. We begin with candidate videos that are found by variants of searches for a set of 10 objects among Creative Commons videos on YouTube. Within these videos, we find stationary continuous shots in these videos with homographies [90] fit on ORB [229] features. Many of these clips cannot depict interaction since they do not contain any people or do not contain the objects of interest. We filter by responses by a hand detector [236] trained on 100K+ frames of Internet data, as well as object detectors trained on COCO [158] and LVIS [87]. These filtering steps maximize the use of annotator time by eliminating clear negatives, and generate a large number of candidate clips.

With a collection of candidate clips of interest, we then turn to manual annotation. For a given clip, we hire an annotation company to annotate frames sparsely (every 10 frames) within the clip. They annotate: (**box**) a box around the articulated plane and its type, if it exists; and (**axis**) the projection of the articulation axis, framed as a line segmentation annotation problem. This results in a set of 19411 frames, containing 19411 boxes around articulating planes with 13508 rotation axes and 2755 translation axes, as well as 39411 negative frames. The number of articulation axes is not equal to the number of boxes, since some articulation axes are outside the image. We provide training, validation, and test splits based on uploader, leading to 7845/601/1001 videos in the train/val/test split. A more complete description of our annotation pipeline appears in the supplement.

We collect two additional annotations. For the test set, we also annotate the surface normal of the plane following [34], so we can evaluate how well our model can learn 3D properties. To show generalization, we also collect the same annotations except surface normals on the Charades [244] dataset.

Data Availability and Ethics. Our data consists of videos that users uploaded publicly and chose to share as Creative Commons data. These do not involve interaction with humans or private data. We filtered obviously offensive content, videos depicting children, and cartoons. Examples appear throughout the chapter; screenshots of annotation instructions and details appear in the supplement.

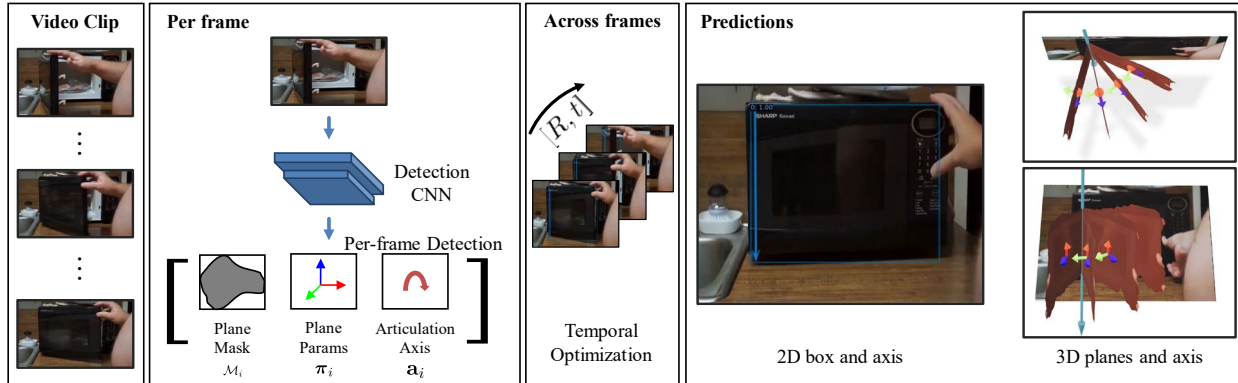


Figure 4.2: Overview of our approach. (a) Given an ordinary video clip, we first apply our 3D Articulation Detection Network (3DADN) to detect 3D planes can be articulated for each frame. (b) We then apply temporal optimization to fit the articulation model. Final results are demonstrated in both 2D image and 3D rendering.

4.4 Approach

The goal of our approach is to detect and characterize planar articulation in an unseen RGB video clip. These articulations are an important special case that are ubiquitous in human scenes. As shown in Figure 7.1, we propose a *3D Articulation Detection Network* (3DADN) to solve the task. As output, the 3DADN produces the type of motion (rotation or translation), a bounding box around where the motion is located, the 2D location of the rotation or translation axis, and the 3D location of the articulated plane. The 3DADN’s output is followed by post-processing to find a consistent explanation over the whole video.

4.4.1 3D Articulation Detection Network

The 3DADN processes each frame independently. Its output consists of: a segment mask \mathcal{M}_i ; plane parameters $\boldsymbol{\pi}_i = [\mathbf{n}_i, o_i]$ giving the plane equation $\boldsymbol{\pi}_i^T [x, y, z, -1] = 0$ (where \mathbf{n}_i is the the plane normal with $\|\mathbf{n}_i\|_2 = 1$ and o_i is the plane offset); a projected rotation or translation axis $\mathbf{a}_i = [\theta, p]$ which is the projection of the 3D articulation axis; and articulation type.

We use a top-down approach to detect this representation, which we train on RGB videos that depict articulation *without* 3D information as well as RGBD images *with* 3D information that do not depict articulation. Our backbone is a Faster R-CNN [222] style network that first detects bounding boxes for the articulating objects and classifies them into two classes (rotation and translation). These boxes provide ROI-pooled features that are passed into detection heads that predict our outputs ($\mathcal{M}_i, \boldsymbol{\pi}_i, \mathbf{a}_i$). Our heads and losses for \mathcal{M}_i follow the common practice of Mask R-CNN [94]. We describe \mathbf{a}_i and $\boldsymbol{\pi}_i$ below.

Parameterizing Rotation and Translation Axis. We model the projected articulation axis as a 2D line in the image. This projected axis is the projection of the 3D articulation axis (e.g., the hinge of a door). We describe the projected axis with the normal form of the line, $x \cos(\theta) + y \sin(\theta) = p$ where $p \geq 0$ is the distance from the box to the center and θ is the inclination of the normal of the axis in pixel coordinates. Since a translation corresponds to a direction/family of lines as opposed to a line, we define $p = 0$ for translation arbitrarily.

The articulation head contains two independent branches for predicting the rotation and translation axes. We handle the circularity of the prediction of θ by lifting predictions and ground-truth for the angle to the 2D unit circle; since the line is 180-degree-ambiguous (i.e., $\theta + \pi$ is the same as θ), we predict a 2D vector $[\sin(2\theta), \cos(2\theta)]$. The resulting network thus predicts a 3D vector containing θ and p , which we supervise with a L1 loss.

Parameterizing Plane Parameters. Following a line of work on predicting planes in images, we use a 3D plane [161] to represent the 3D locations of the articulated objects, since many common articulated objects like doors, refrigerators, and microwaves can be modeled as planes and because past literature [159, 119, 120] suggests that R-CNN-style networks are adept at predicting plane representations.

A 3D plane is represented by plane parameters $\pi_i = [\mathbf{n}_i, o_i]$ giving the plane equation $\pi_i^T [x, y, z, -1] = 0$. With camera intrinsics, planes can be recovered in 3D and with a mask, this plane can be converted to a plane segment. Following [159, 120], we extend R-CNN by adding a plane head which directly regresses the normal of the plane. A depth head is used to predict depth of the image. The depth is only used to calculate the offset value of the plane. We supervise with L2 loss for the plane normal regression and L1 loss for the depth regression.

Training. There is no dataset that is non-synthetic and large enough to train a 3DADN directly: the 3DADN needs both realistic interactions and 3D information. However, we can train the 3DADN in parts. In the first stage, we train the backbone, RPN, and axis heads directly on our Internet video training set, which has boxes and axes. We then freeze the backbone, RPN, and axis heads and fine-tune the mask and plane head on a modified version of ScanNet [42].

In particular, we found that humans often occlude the objects they articulate and models that had not seen humans in training produced worse qualitative results. We therefore composited humans from SURREAL [267] into the scenes. We randomly sample 98,235 ScanNet images, select a synthetic human and render it on ScanNet backgrounds. In training, we do not change the ground truth, pretending the ground truth plane is partially occluded by humans and training our model to identify them.

Meanwhile, we found that the order of training the heads was crucial. Planes in Scan-

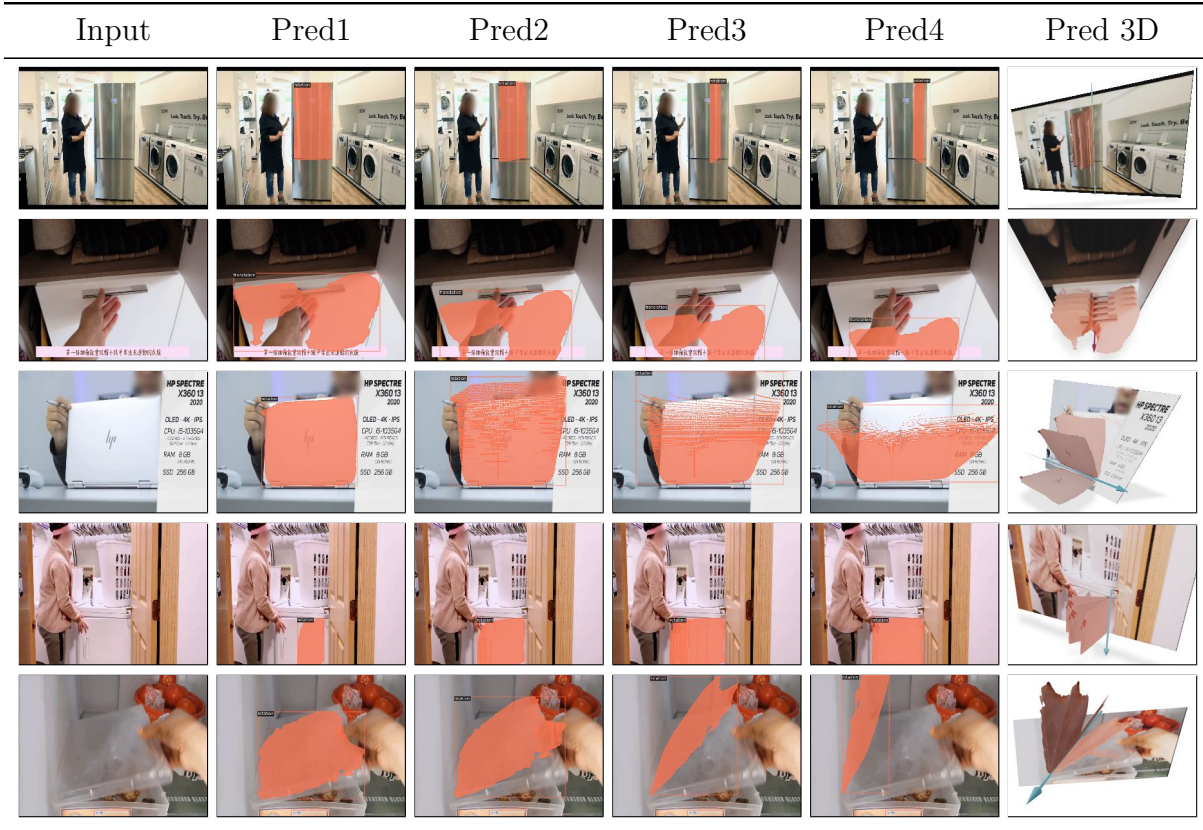


Figure 4.3: Predictions on Internet videos. For each example, we show the input (left), detected 2D planes and how they will be articulated using the predicted articulation axes and surface normals (middle). We also show 3D renderings to illustrate how these common objects are articulated in the 3D space (right). The predicted rotation axis is shown as the **Blue** arrow, and translation axis is the **Pink** arrow.

Net [42] are defined geometrically, and so unopened doors often merge with walls; similarly, ScanNet [42] does not contain transitional moments during which planes are articulating. Thus, RPNs trained on ScanNet [42] perform poorly on articulation videos. Instead, it is important to train the RPN on our Internet videos, freeze the backbone, and only rely on ScanNet to train plane parameters and masks, which are unavailable in Internet videos. During inference we keep the ScanNet camera since our data does not have camera intrinsics.

Implementation Details. Full architectural details of our approach are in the supplemental. Our model is implemented using Detectron2 [284]. The backbone uses ResNet50-FPN [155] pretrained on COCO [158].

4.4.2 Temporal Optimization

After the 3DADN provides per-frame estimates of articulations, we perform temporal optimization to find a single explanation for the detections across frames. We are given a sequence of detections indexed by a time of the form $[\mathcal{M}_i^{(t)}, \boldsymbol{\pi}_i^{(t)}, \mathbf{a}_i^{(t)}]$. We aim to find a single consistent explanation for these detections.

Tracking. Optimizing requires a sequence of planes to optimize over. We match box i with the box in the next frame according to pairwise intersection over union (IoU). Box i at t matches box $j = \arg \max_{j'} \text{IoU}(\mathcal{M}_i^{(t)}, \mathcal{M}_{j'}^{(t+1)})$ at time $t + 1$; we then track greedily to get a sequence. We subsequently drop the subscripts for clarity.

Articulation Model Fitting. Given a sequence of detections, we find a consistent explanation via a RANSAC-like approach. We begin with a hypothesis of a plane segment $\boldsymbol{\pi}$ and articulation axis \mathbf{a} , which we obtain by selecting output on a reference frame. Along with an assumed camera intrinsics \mathbf{K} , the plane parameters let us lift the plane segment and axis to 3D, producing 3D plane segment $\boldsymbol{\Pi}$ and 3D axis \mathbf{A} . Then, for each frame t , we solve for an articulation degree $\alpha^{(t)}$ maximizing the reprojection agreement with the predicted mask at time t . Let us define the reprojection score as

$$r(\alpha, t) = \text{IoU}(\mathcal{M}^{(t)}, \mathbf{K}[\mathbf{R}_{\alpha}, \mathbf{t}_{\alpha}]\boldsymbol{\Pi}), \quad (4.1)$$

where $\mathbf{R}_{\mathbf{A}, \alpha}$ and $\mathbf{t}_{\mathbf{A}, \alpha}$ are α steps over the rotation and translation for axis \mathbf{A} . We then solve for $\alpha^{(t)}$ by solving $\arg \max_{\alpha} r(\alpha)$, which gives a per-frame angle using grid search. We detect articulation by calculating how well the rotation degree $\alpha^{(t)}$ can be explained as a linear function of t (i.e., that there is constant motion). Since many scenes are not constant motion, we have loose thresholds: we consider $R^2 \geq 0.4$ and slope $k > 0.1$ to be an articulation. We exclude hypotheses where all $r(\alpha^{(t)}, t) < 0.5$.

4.5 Experiments

We have introduced a method that can infer 3D articulation in Section 4.4. In the experiments, we aim to answer the following questions: (1) how well can one detect 3D articulating objects from ordinary videos; (2) how well do alternate approaches to the problem do?

4.5.1 Experimental Setup

We first describe the setup of our experiments. Our method aims to look at an ordinary RGB video and infer information about an articulated plane on a object in 3D, including:

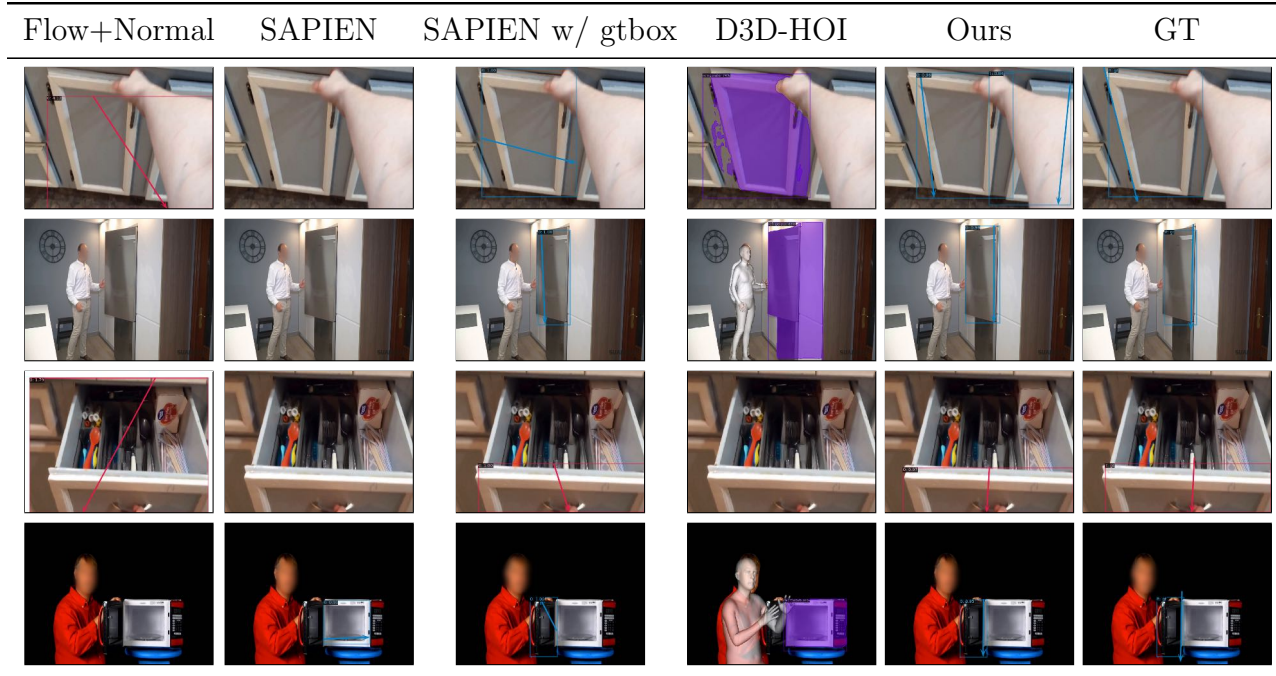


Figure 4.4: We compare our approach with four baselines. See detailed discussions in the text. We show translation in **Pink** and rotation in **Blue**, except D3D-HOI which uses a different detector.

whether the object is articulating, its extent, and the projection of its rotation or translation axis. We therefore evaluate our approach on two challenging datasets, using metrics that capture various aspects of a 3D plane articulating in 3D.

Datasets: We validate our approach on both Internet videos (described in Section 4.3) and the Charades dataset [243]. We use Charades for cross-dataset evaluations. We focus on Charades videos that are opening objects (doors, refrigerators, etc.), and annotate 2491 frames across 479 videos; we also randomly sample 479 negative videos containing 4401 negative frames. Our Charades annotation process is similar to Internet videos, with the exceptions that: we annotate only rotations as Charades contains few translation articulations; and we do not annotate surface normals.

Evaluation Criteria: Evaluation of our approach is non-trivial, since our assumed input (RGB videos) precludes measuring outputs quantitatively in 3D. We therefore evaluate our approach on a series of subsets of the problem. We stress from the start though that these metrics are what can be measured (due to the use of RGB inputs), as opposed to the full rich output.

Articulation Recognition: We first independently evaluate the ability to detect whether someone is articulating this object at a point in time. We frame this as a binary prediction

Table 4.1: We report AUROC for Articulation Recognition, as well as AP for Articulation Description. To separate out difficulties in detecting articulation and characterizing its parameters, we assist Flow+Normal and 3DADN+SAPIEN with ground truth bounding box and denote it as gtbox. 3DADN+SAPIEN cannot detect most objects without the help of gtbox.

Methods	gtbox	Recog.	Rotation			Translation		
		AUROC	bbox	bbox+axis	bbox+axis+normal	bbox	bbox+axis	bbox+axis+normal
Flow [261] + Normal [34]	✗	68.5	7.7	0.3	0.0	0.3	0.0	0.0
Flow [261] + Normal [34]	✓	-	-	3.0	0.3	-	1.4	0.7
D3D-HOI [289] Upper Bound	✗	62.7	28.8	19.7	n/a	4.70	4.7	n/a
3DADN + SAPIEN [287]	✓	-	-	16.8	1.40	-	15.1	0.40
Ours	✗	76.6	61.3	30.4	17.2	34.0	27.1	17.9

problem. This is surprisingly difficult in real scenes because objects are typically partially occluded by humans when humans articulate them, and because humans often touch articulated objects (e.g., cleaning the surface) without opening it. We use AUROC to measure the performance.

Articulation Description: We next evaluate the ability of a system to detect the articulated object, corresponding articulation type (rotation/translation), axes, and surface normals. We follow other approaches [210, 265, 138, 198, 120] that reconstruct the scene factored into components and treat it as a 3D detection problem, evaluated using average precision (AP). We define error metrics as follows: (*Bounding box*) IoU, thresholded as 0.5. We find the normal COCO AP, which measures IoU up to 0.95, to be too strict because the precise boundaries of articulating parts are often occluded by people and hard to annotate. (*Axes*) EA-score from the semantic line detection literature [315]. This metric handles a number of edge cases; we use 0.5 as the threshold as recommended by [315]. (*Surface normal*) mean angle error, thresholded at 30°, following [274, 63]. A prediction is a true positive only if all errors are lower than our thresholds. We calculate the precision-recall curve based on that and report AP for varying combinations of metrics.

Baselines: Prior approaches for articulation detection have focused on robots, synthetic datasets, and real-world RGBD scans. These are different from our setting for two reasons. First, videos of people articulating objects show a noisy background with a person interacting with and occluding the object, as opposed to an isolated articulated object in a simulator. Second, RGB videos do not have depth, which is often a requirement of existing articulation models. For example [151] requires depth, and while they show results on real-world depth scans, their RGBD scans only contain a static object without humans. We propose to compare with the following methods.

3DADN + SAPIEN [287] Data: To test whether we can solve the problem just by training on synthetic data, we create a synthetic data-based method where we train our 3DADN system

on synthetic data. We render a synthetic dataset using SAPIEN [287] by randomly sampling and driving 3D objects. We filtered 1053 objects of 18 categories with movable planes from PartNet-Mobility Dataset [287], such as doors and laptops. We render frames with the objects articulated, with location parameters picked to give plausible scenes, and extract the information needed to train 3DADN. Without a background, the detection problem becomes trivial, so to mimic real 3D scenes, we blend the renderings with random ScanNet [42] images as the background and render synthetic humans from SURREAL [267]. For fair comparison, we use the same ScanNet+SURREAL images used to train our system’s plane parameter head. When evaluated on SAPIEN data, this approach performs well and obtains an AP of (bbox) 60.3, (bbox+rot) 64.1, (bbox+rot+normal) 41.0.

Bottom-up Optical Flow [261] and Surface Normal Change [34] (Flow+Normal): To test whether the data can be solved by the use of fairly simple cues, we construct a baseline that uses Optical Flow [261] (since articulating objects tend to cause movement) and Surface Normals [34] (since rotating planes change their orientation). Both flow and normals provide a $H \times W$ map that can be analyzed. We also use the output of a human segmentation system [109] that was trained on multiple datasets and mask normal and flow magnitude maps wherever it improves performance. Given these maps, we recognize the presence of articulation via logistic regression on a feature vector consisting of the fraction of pixels above multiple thresholds; we recognize bounding boxes via thresholding and finding the tightest enclosing box; we estimate rotation axis as perpendicular to the mean flow change in the bounding box (flow tends to increase away from hinges); we find translation axis using mean flow direction in the box; we find articulation normal using mean predicted normals in the box. Throughout, we use the optimal option of surface normals and flow; this hybrid system performs substantially better than either flow or normals alone.

Baselines with + GT Box: To separate out difficulties in detecting articulation and characterizing its parameters, we also experiment with giving baselines ground-truth bounding box information about the articulating object. This gives an upper-bound on performance.

D3D-HOI [289] Upper Bound: We compare with D3D-HOI since it accepts RGB video as input and detects how humans articulate objects. A direct comparison with D3D-HOI is challenging since it only works when EFT [121] reconstructs 3D human poses and Pointrend [132] detects the objects that are assumed to articulate and correct CAD models are chosen for the object. However, EFT does not work well on the dataset due to truncated or multiple humans on Internet videos [226, 133]. We therefore report upper-bounds on the performance. We assume it predicts the ground truth bounding box, when EFT mask and pseudo ground truth 2D human segmentation mask [108] has IoU > 0.5 and PointRend [132] produces a mask on articulated objects with confidence > 0.7 .

Ours: This is our proposed method. It includes both the per-frame approach described in Section 4.4.1 and the optimization approach of Section 4.4.2. We note that this approach also produces outputs that are not being quantitatively measured, such as a 3D plane articulating in 3D. These are qualitatively shown in Figures 4.1 and 4.3.

4.5.2 Results

We first show qualitative results in Figure 4.3. On challenging Internet videos, our approach usually detects and recovers the 3D articulated plane regardless of categories.

In Figure 4.4, we compare our approach with four baselines visually. Flow can occasionally locate articulation (third row), but in most cases, flow is not localized to only the object articulating (e.g. camera movement, top row). Training purely on SAPIEN [287] data has difficulty detecting articulated objects in Internet videos, even if we show all detected objects with confidence score > 0.1 . It learns some information of articulation axes when we assist it with ground truth bounding boxes. D3D-HOI [289] relies on both EFT [121] to detect humans and PointRend [132] to detect objects. However, EFT has difficulty predicting 3D humans on Internet videos.

Quantitative Results. We evaluate the approach quantitatively on the three tasks in Table 4.1. Our approach substantially outperforms the alternate methods. While statistically-combined bottom-up cues [261, 34] do better than chance at predicting the presence of an articulation, they are substantially worse than the proposed approach and fail to obtain sensible bounding boxes. Even when given the ground-truth box, this method fails to obtain good axes. Due to the frequency of truncated humans in Internet videos [226, 133], D3D-HOI [289]’s performance upper-bound is substantially lower than our method’s performance. The detection system when trained on synthetic data from [287] fails on our system; when given a good bounding box, synthetic training data obtains reasonable, but inferior numbers and poor accuracy in predicting normals.

Ablations – Optimization. Our optimization produces modest gains in recognition accuracy and axis localization in 2D: It improves recognition AUROC from 74.0 to 76.6, rotation AP from 16.6 to 17.2 and translation AP from 14.3 to 17.9. This small gain is understandable because the evaluation is per-frame and the optimization mainly seeks to make the predictions more consistent. If we quantify the consistency in the results before and after optimization, we find that the EAScore [315] between tracked predicted frames increases from 0.69 (before optimization) to 0.96 (after optimization).

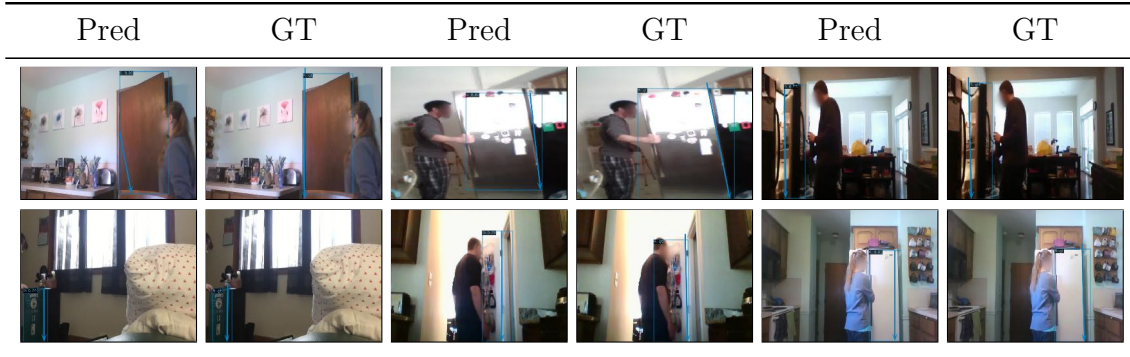


Figure 4.5: Qualitative results on Charades dataset. Without finetuning on Charades data, our model obtains strong performance on detecting and characterizing 3D articulation.

Table 4.2: Evaluation on Charades dataset [243]. We only report rotation AP since Charades does not have sufficient translation motion.

Methods	gtbox	Recog.	Rotation	
		AUROC	bbox	bbox+axis
Flow [261] + Normal [34]	✗	53.7	3.1	0.2
Flow [261] + Normal [34]	✓	-	-	4.2
D3D-HOI Upper Bound	✗	55.9	14.9	13.7
3DADN + SAPIEN [287]	✓	-	-	1.54
Ours	✗	58.4	12.0	12.8

4.5.3 Generalization Results

We next test our trained models *without fine-tuning* on Charades [244]. We show results in Figure 4.5. Our approach typically generates reasonable estimations. We find that the video quality and resolution of Charades is lower relative to our videos, with many dark or blurry videos.

We also show quantitative evaluations in Table 4.2. Here, our performance is slightly diminished. However, we substantially outperform the baselines. We are only marginally outperformed by D3D-HOI upper bound, which assumes perfect performance so long as the data can be obtained.

4.5.4 Limitations and Failure Modes

We finally discuss our limitations and typical failure modes in Figure 4.6. We find some examples are particularly challenging: (1) Column 1: some images may contain hard examples where the axis types are hard to figure out. (2) Column 2: the axis is outside of the image frame or its location is ambiguous due to symmetry or occlusion. (3) Column 3:

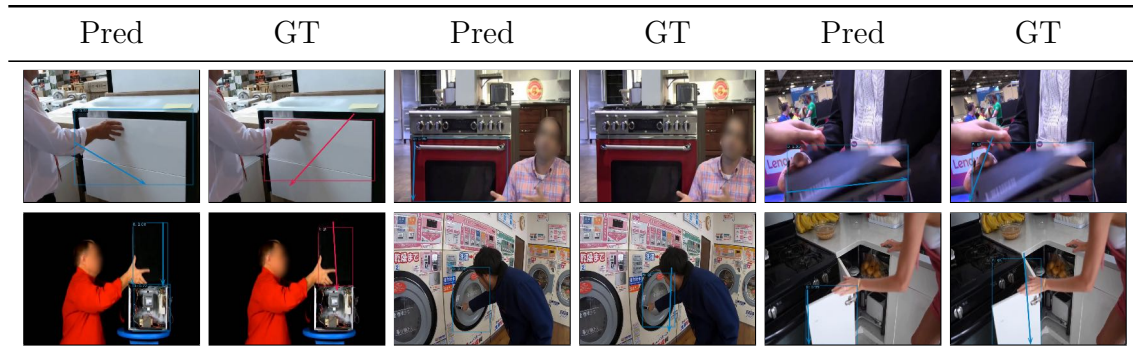


Figure 4.6: Typical failure modes. (1) Ambiguity of articulation type; (2) Axis outside of the frame or ambiguity of articulation axis location due to symmetry; (3) Object has complex motions (a person moving an object while articulating it; the rotation axis is outside of the articulating surface).

the object has complex dynamics or dual axes; for example, a person moving a laptop while opening it or the cabinet has multiple joints.

4.6 Conclusion

We have demonstrated our approach’s ability to detect and characterize 3D planar articulation of objects from ordinary videos. Future work includes combining 3D shape reconstruction with the articulation detection pipeline.

Our approach can have positive impacts by helping build smart robots that are able to understand and manipulate articulated objects. On the other hand, our approach may be useful for surveillance activities.

CHAPTER 5

Understanding 3D Object Interaction from a Single Image

Humans can easily understand a single image as depicting multiple potential objects permitting interaction. We use this skill to plan our interactions with the world and accelerate understanding new objects without engaging in interaction. In this chapter, we would like to endow machines with the similar ability, so that intelligent agents can better explore the 3D scene or manipulate objects. Our approach is a transformer-based model that predicts the 3D location, physical properties and affordance of objects. To power this model, we collect a dataset with Internet videos, egocentric videos and indoor images to train and validate our approach. Our model yields strong performance on our data, and generalizes well to robotics data. The material in this chapter is derived from [209].

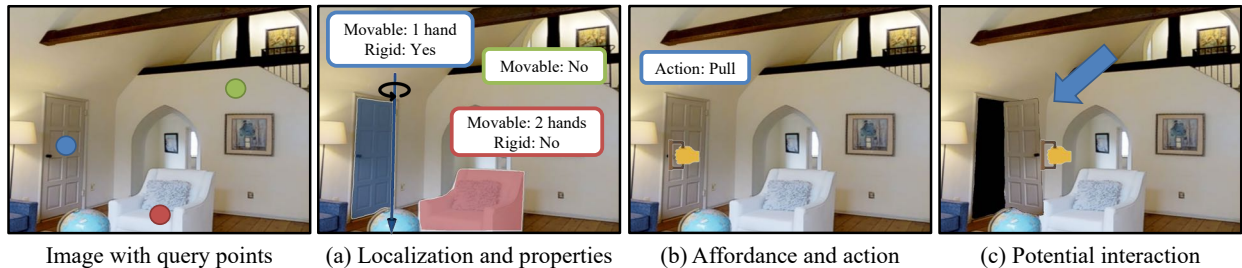


Figure 5.1: Given a single image and a set of query points $\bullet \circ \bullet$, our approach predicts: (a) whether the object at the location can be moved 👤 , its rigidity 🔪 and articulation class 🚪 , and location 📍 ; (b) an affordance 👉 and action 👋 ; and (c) potential 3D interaction for articulated objects. This ability can assist intelligent agents to better manipulate objects or explore the 3D scene.

5.1 Introduction

What can you do in Figure 5.1? This single RGB image conveys a rich, interactive 3D world where you can interact with many objects. For instance, if you grab the chair with two hands, you can move it as a rigid object; the pillow can be picked up freely and squished; and door can be moved, but only rotated. This ability to recognize and interpret potential affordances in scenes helps humans plan our interactions and more quickly learn to interact with objects. The goal of this work is to give the same ability to computers.

Obtaining such an understanding of potential interactions from a single 3D image is beyond the current state of the art in scene understanding because it spans multiple disparate subfields of computer vision. For instance, single image 3D has made substantial progress [217, 198, 303, 79], but primarily focuses on the scene *as it exists*, as opposed to *as it could be*. There has been an increasing interest in understanding articulation [151, 287, 211], but these works primarily focus on articulation *as it occurs* in a 3D model or carefully collected demonstrations, instead of *as it could occur*. Finally, while there is long-standing work on enabling robots to learn interaction and potential interaction points [204, 254], these works focus primarily on evaluation in primarily the same environment (*e.g.* the lab) and do not focus on applying the understanding in entirely new environments.

We propose to bootstrap this interactive understanding by developing (1) a problem formulation, (2) a rich dataset of annotations on challenging images, and (3) a transformer-based approach. We frame the problem of recognizing the articulation as a prediction-at-a-query-location problem: given an image and 2D location, our method aims to answer “what can I do here?” in the style of classic point-and-click games like *Myst*. We frame “what can I do here” via a set of common questions: whether the object can be moved, its extent when moved and location in 3D, rigidity, whether there are constraints on its motion, as well as estimates of how one would interact the object. To maximize the potential for downstream transfer, our questions are chosen to be generic rather than specific to particular hands or end-effectors: knowing where to act or the degrees of freedom of an object may accelerate reinforcement learning even if one must still learn end-effector-specific skills.

In order to tackle the task, we introduce a transformer-based model. Our approach, described in Section 5.5 builds on a detection backbone such as Segment-Anything [131] in order to build on the advances and expertise of object detection. We extend the backbone with additional heads that predict each of our “what I can I do here” tasks, and which can be trained end-to-end. As an advantage of our formulation, we can train the system on sparse annotations; we believe this will be helpful for eventually converting our direct supervision to supervision via video.

Powering our approach is a new dataset, described in Section 5.4, which we name the 3D Object Interaction dataset (*3DOI*). In order to maximize the likelihood of generalizing to new environments, the underlying data comes from diverse sources, namely Internet and egocentric videos as well as 3D renderings of scene layouts. We provide annotations of our tasks on this data and, due to the source of the data, we also naturally obtain 3D supervision in the form of depth and normals. In total, the dataset has over 50K objects across 10K images, as well as over 31K annotations of non-interactable objects (e.g., floor, wall).

Our experiments in Section 5.6 test how well our approach recognizes potential interaction, testing on both unseen data in 3DOI as well as robotics data. We compare with a number of alternatives, including generalizing from data of demonstrations [211, 196] and synthetic data [287], as well alternate network designs. Our approach outperforms these models and shows strong generalization to the robotics dataset WHIRL [6].

To summarize, we see our primary contributions as: (1) the novel task of detecting 3D object interactions from a single RGB image; (2) 3D Object Interaction dataset, which is the first large-scale dataset containing objects that can be interacted and their corresponding locations, affordance and physical properties; (3) A transformer-based model to tackle this problem, which has strong performance on the 3DOI dataset and robotics data.

5.2 Related Works

This chapter proposes to extract 3D object interaction from a single image. This problem lies at the intersection of 3D vision, object detection, human-object interaction and scene understanding. It is also closely related to downstream robotics applications.

Interactive scene understanding. Recently, the computer vision community is increasingly interested in understanding 3D dynamics of objects. It is motivated by human-object interaction [27, 78, 236], although humans do not need to be present in our setting. Researchers try to understand the 3D shapes, axes, movable parts and affordance on synthetic data [193, 287, 194, 117, 279, 151, 277, 279], videos [211, 89, 82, 196, 177, 147] or point clouds [118, 100]. Our work is mainly related to [211, 89, 82] since they work on real images, but is different from them on two aspects: (1) they need video or multi-view inputs, but our input is only a single image; (2) their approaches recover objects which are being interacted, while our approach understands potential interactions before any interactions happen. Finally, OPD [117, 256] tackles a similar problem for articulated objects, but ours also work for non-articulated objects.

Object detection. The training anchor-based object detection pipeline basically follows the pipeline of Mask R-CNN [94, 132, 222, 130]. As the development of transformer-based


models goes, DETR [21], AnchorDETR [278] and MaskFormer [36] approach object detection as a direct set prediction problem. Recently, Kirillov *et al.* proposes Segment Anything Model [131], which predicts object masks from input prompts such as points or boxes. Our network needs to be built on decoder-based backbones [21, 36, 131], and we choose SAM [131] due to its state-of-the-art performance.

Single image 3D. Since our problem requires us recover 3D object interaction instead of 2D from a single image, it is also related to single image 3D. In the recent few years, researchers have developed many different approaches to recover 3D from a single image, including depth [303, 217, 153, 33, 63], surface normals [274, 67], 3D planes [161, 159, 120] and shapes [39, 179, 79, 198]. Our work is built upon their works. Especially, our architecture is motivated by DPT [217] which trains ViT for both segmentation and depth estimation.

Robotics manipulation. Manipulation of objects is a long-term goal of robotics. Researchers have developed various solutions for different kinds of objects in different scenes, ranging from articulated objects [254, 204, 40, 55, 283] to deformable objects [292, 293, 276, 37]. While manipulation is not the goal of this chapter, understanding objects and the environment in 3D is typically an important part of a manipulation pipeline. The chapter mainly improves the perception part, which can potentially improve manipulation. Therefore, we also test our approach on robotics data [6], to show it can generalize.

5.3 Overview

Given a single image, our goal is to be able to answer “What could I do here?” with the object at a query point. We introduce annotations in Section 5.4 as well as a method for the task in Section 5.5. Before we do so, we present a unified explanation for the questions we answer as well as the rationale for choosing these questions. We group our questions into six property types, some of which are further subdivided. Not all objects support all questions: objects that cannot be moved, for instance, do not have other properties and objects that can be freely moved do not have rotation axes. We further note that some objects defy these properties – ball joints, for example, permit a 2D subspace of motion – our goal is to identify a large subspace of potential interactions.

Movable  The most important subdivision is whether the object at the query point can be moved. This follows work in both 3D scene understanding [246] and human-object interaction [236] that subdivide objects into how movable they are. We group objects into three categories based on how easily the object can be moved: (1) *fixtures* which effectively cannot be moved, such as walls and floor; (2) *one hand* objects that can be moved with a

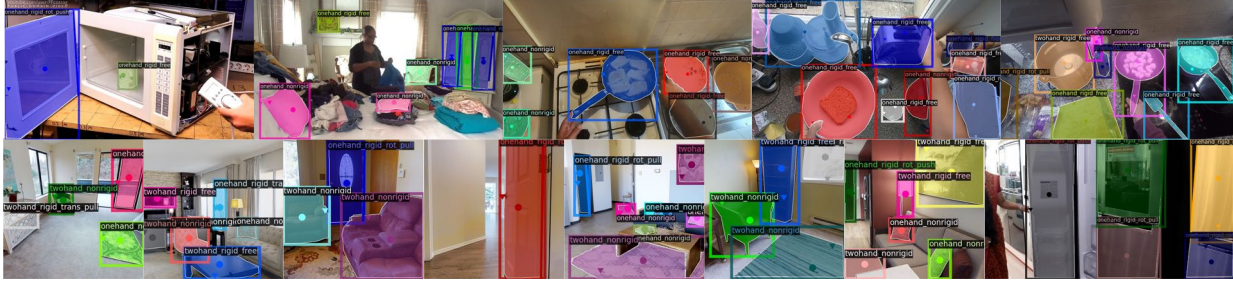


Figure 5.2: Example annotations of our 3DOI dataset. Our images come from Internet videos [211], egocentric videos [47] and renderings of 3D dataset [62]. ● is the query point, and ▼ is the affordance.

single hand, such as a water bottle or cabinet door; (3) *two hand* objects that require two hands to move, such as a large TV. We frame the task as three-way classification.

Localization 📍 Understanding the extent of an object is important, and so we localize the object in the world. Since our objects consist of a wide variety of categories, we frame localization as 2D instance segmentation as in [94, 21], as well as a depthmap to localize the object in 3D [217, 303]. These properties can be estimated for most objects.

Rigidity 🛠️ To understand action, one primary distinction is rigid-vs-non-rigid since rigid objects are subject to substantially simpler rules of motion [152]. We therefore classify whether the object is rigid or not.

Articulation 📐 Most rigid objects can further decomposed as permitting freeform, rotational / revolute, or translation / prismatic motion [254]. Each of these requires different end-effector interactions to effectively interact with. We frame the articulation category as a three-way classification problem, and recognizing the rotation axis as a line prediction problem following [211].

Action 🖐️ We also want to understand what the potential action could be to interact with the object. Here we focus on three types of actions: pull, push or other.

Affordance 📍 Finally, we want to know where we should interact with the object. For example, we need to manipulate the handle if we want to open a door. We predict a probability map which is over the location of the affordance.

5.4 3D Object Interaction Dataset

One critical component of our contribution is accurate annotations of object interactions, as there is no publicly available data. In this chapter, we introduces 3D Object Interaction dataset (3DOI), which is the first dataset. We picked data that can can be easily integrated

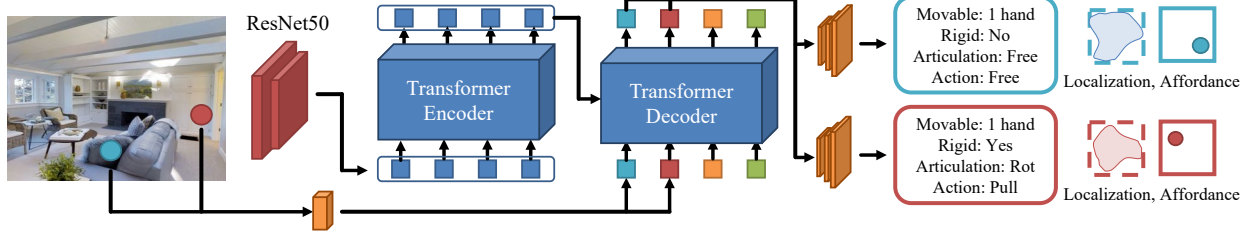


Figure 5.3: Overview of our approach. The inputs of our network is a single image and a set of query points. For each query point, it predicts the potential 3D interaction, in terms of movable, location, rigidity, articulation, action and affordance.

with 3D, including a 3D dataset, so that we have accurate 3D ground truth to train our approach. Examples of our data are shown in Figure 5.2.

Images. Our goal is to pick up diverse images representing real-world scenarios. In particular, we want our images contain a lot of everyday objects we can interact with. Therefore, we sample 10K images from a collection of publicly available datasets: (1) Articulation [211] comes from third-person Creative Commons Internet videos. Typically, a video clip contains humans manipulating an articulated objects in households. We randomly sample 3K images from the articulation dataset; (2) EpicKitchen [47] contains egocentric videos making foods in kitchen environments. We sample 2K images from EpicKitchen; (3) Taskonomy [308] is an indoor 3D dataset with real 2D image and corresponding 3D ground truth. We use the renderings by Omnidata [62]. We sample 5k images from the taskonomy split of Omnidata starter dataset. Overall, there are 10K images.

Annotation. With a collection of images with potential objects we can interact, we then turn to manual annotation. For a single image, we select around 5 interactable query points, including both large and small objects. For each query point, we annotate: (*Movable* 🧑🧑) one hand, two hand, or fixture. (*Localization* 📏) The bounding box and mask of the part this point belonging to. (*Rigidity* 🪛) Rigid, or nonrigid. (*Articulation* 📏) Rotation, translation or freeform. We also annotate their rotation axes. (*Action* 🖐️) Pull, push or others. (*Affordance* 📍) A keypoint which indicates where we should interact with the object. At the same time, our taskonomy [308] images come with 3D ground truth, including depth and surface normals. We also annotate 31K query points of fixtures. Finally, we split 10K images into a train/val/test set of 8k/1k/1k split, respectively.

Availability and Ethics. Our images come from three publicly available datasets. Taskonomy does not contain any humans. The video articulation dataset comes from Creative Commons Internet videos. We do not foresee any ethical issues in our dataset.

5.5 Approach

We now introduce a model which can take an image and a set of query point and answer all of questions we asked in Section 5.3, including movable, localization, rigidity, articulation, action and affordance. A brief overview of our approach is shown in Figure 5.3.

Since our inputs include a set of query points and our outputs include both bounding boxes and segmentation masks, we mainly extend SAM [131] to build our model. Compared with traditional detection pipeline such as Mask R-CNN [94], we can use a query point to naturally guide SAM to detect the corresponding object. Mask R-CNN generates thousands of anchors for each image, which is challenging to find the correct matching. However, we also compare with alternative network architectures in our experiments for completeness. We find they can also work despite being worse than SAM. For simplicity, we assume there is only a single query point. But our model can accept hundreds of query points at a time.

5.5.1 Backbone

The goal of our backbone is to map an image I and a query point $[x, y]$ to a pooled feature $h = f(I; [x, y])$. Full details are in the supplemental.

Image Encoder. Our image encoder is a MAE [92] pretrained Vision Transformer (ViT) [57], following SAM [131]. They map a single image I to the memory of the transformer decoder.

Query Point Encoder. We transfer the query point $[x, y]$ to positional encodings [259], which is then feed into the transformer decoder. We use the embedding k to guide the transformer to produce the feature h for different query points.

Transformer Decoder. The decoder accepts inputs of the memory from the encoder, and an embedding k of the query point. It produces a embedding h for each query point, and we use it to predict all the properties, like a ROI feature.

5.5.2 Prediction Heads

We now describe how to map from the pooled feature h to the features. Each prediction is done by a separate head that handles each output type.

Movable 🚶 We add a linear layer and map the hidden embedding h to the prediction of movable. We use the standard cross entropy loss to train it.

Localization 📍 We follow DETR standard practice to predict bounding boxes and segmentation masks. For the bounding box, we represent it use a 4d vector and use a 3-layer MLP to regress it. We train boxes using L1 loss and generalized IoU [223] as the loss function.

For segmentation masks, we adapt the panoptic segmentation module (similar to a decoder) to produce a single object mask, and use focal loss [156] and DICE [190] as loss functions. For depth, we add an additional query to the transformer decoder, and produce a dense depth map using the segmentation module as a decoder. We train depth using scale- and shift-invariant L1 loss and gradient-matching loss following [303, 217, 153]. The shift and scale are normalized per image.

Rigidity 🛠️ Similar to movable, we add a linear layer to predict whether the object is rigid or not. We train the linear layer using a standard binary cross entropy loss.

Articulation 📐 We first add a linear layer to predict whether the interactive object is rotation, translation or freeform, and we use the standard cross entropy loss to train it. For the rotation axis, we follow [211, 315] to represent an axis as a 2D line (θ, r) . Any points on this line satisfy $x \cos(\theta) + y \sin(\theta) = r$ where θ represents the angle and r represents the distance from the object center to the line. In training, we represent the 2D line as $(\sin 2\theta, \cos 2\theta, r)$, so that the axis angle is in a continuous space [323]. We use a 3-layer MLP to predict the axis, similar to bounding boxes as both tasks require localization. We use L1 loss to train it.

Action 🖐️ Similar to movable, we add a linear layer to predict what the potential action is to interact with the object. We train the linear layer using a standard binary cross entropy loss.

Affordance 🏠 Our prediction of affordance is a probability map, while our annotation is a single keypoint. However, affordance can have multiple solutions. Therefore, we transform the annotation of affordance to a 2D gaussian bump [145] and train the network using a binary focal loss [156]. We set the weight of positive examples to be 0.95 and that of negative ones to be 0.05 to balance positives and negatives, as there are more negatives than positives.

Our total loss is a weighted linear combination of all losses mentioned above. Details are in supplemental.

5.5.3 Implementation Details

Full architectural details of our approach are in the supplemental. In practice, we use three different transformer decoders for mask, depth and affordance. The image encoder, query point encoder and mask decoder are pretrained on SAM [131]. Other parts, including affordance head and depth head, are trained from scratch. We use an AdamW optimizer of the learning rate 10^{-4} , and train our model for 200 epochs.

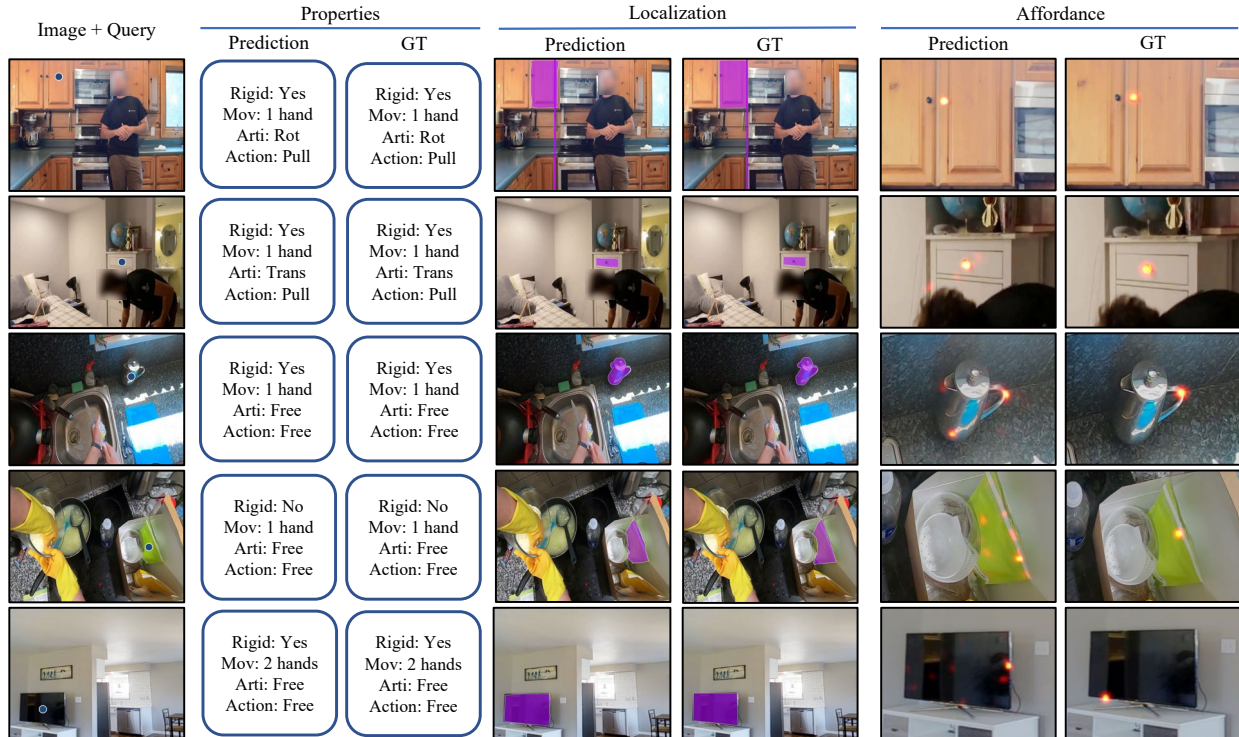


Figure 5.4: Results on our 3DOI dataset. ● indicates the query point. **(Row 1, 2)** Our approach can correctly recognize articulated objects, as well as its type (rotation or translation), axes, and affordance. **(Row 3, 4)** Our approach can recognize rigid and nonrigid objects in egocentric video. **(Row 5)** Our approach can recognize objects need to be moved by two hands, such as a TV. We note that the affordance of these objects have multiple solutions. Affordance is zoomed manually for better visualization. Affordance colormap: min max.

5.6 Experiments

We have introduced an approach that can localize and predict the properties of the moving part from an image. In the experiments, we aim to answer the following questions: (1) how well can one localize and predict the properties of the moving part from an image; (2) how well do alternative approaches to the problem do? We evaluate our approach on our 3DOI dataset and test the generalization to robotics data WHIRL [6].







5.6.1 Experimental Setup

We first describe the setup of our experiments. Our method aims to look at a single RGB image and infer information about the moving part given a keypoint. We therefore evaluate our approach on two challenging datasets, using metrics that capture various aspects.

Datasets. We train and validate our approach on two datasets: 3DOI dataset (described in

Section 5.4), and the WHIRL dataset [6]. WHIRL [6] is a robotics dataset including everyday objects and settings, for example drawers, dishwashers, fridges in different kitchens, doors to various cabinets. We use WHIRL to validate the generalization of our approach and downstream applications in the robotics settings. We split the first frame of all WHIRL videos and annotate them using the same pipeline as our datasets. Typically, humans are not present in the first frame and it’s before any manipulation.







Metrics. We report standard practices of evaluation for all of our predictions. For all metrics, the higher the better. These metrics are detailed as follows:

- Movable , Rigidity , and Action : We report accuracy as these are multiple choice questions.
- Localization : We report Intersection-over-Union (IoU) for our predictions of bounding boxes and masks [157]. We report threshold accuracy for depth [63].
- Articulation : We report accuracy for articulation type. The rotation axis is a 2D line. Therefore, we report EA-Score between the prediction and the ground truth, following [211, 315]. EA-Score [315] is a score in $[0, 1]$ to measure the angle and euclidean distance between two lines.
- Affordance : It’s a probability map and we report the histogram intersection (or SIM) following [196, 18].

Baselines. We compare our approach with a series of baselines, to evaluate how well alternative approaches work on our problem. We first evaluate 3DADN [211], SAPIEN [287], and InteractionHotspots [196] using their pretrained checkpoints, to test how well learning from videos or synthetic data works on our problem. We then train two query-point-based model, ResNet MLP [95] and COHESIV [237], to test how well alternative network architectures work on our problem. The details are introduced as follows.

- (3DADN [211]): 3DADN detects articulated objects which humans are interacting with, extending Mask R-CNN [94]. It is trained on Internet videos. We drop the temporal optimization part since we work on a single image. For each image, it can detect articulated objects, as well as the type (translation or rotation), bounding boxes, masks and axes. Since the inputs of 3DADN do not include a query point, we compare the predicted bounding boxes and the ground truth to find the matching detection, and evaluate other metrics. We lower the detection threshold to 0.05 to ensure we have enough detections to match our ground truth.
- (SAPIEN [287]): The training frames of 3DADN [211] typically have human activities. However, our dataset does not require humans to be present, which may lead to generalization issues. Alternatively, we are interested in whether we can just learn the skill from synthetic data. We train 3DADN [211] on renderings of synthetic objects generated by SAPIEN.

Table 5.1: Quantitative results on our 3DOI dataset. Cat. means category. We report accuracy for all category classification, including movable, rigid, articulation and action. We report mean IoU for box and mask, EA-Score for articulation axis, and SIM for affordance. For all metrics, the higher the better.

Methods	Movable 	Localization 		Rigidity 	Articulation 		Action 	Affordance 
	Cat.	Box	Mask	Cat.	Cat.	Axis	Cat.	Probability
3DADN [211]	-	8.53	6.45	-	44.3	5.63	-	-
SAPIEN [287]	-	5.94	4.57	-	41.6	1.79	-	-
InteractionHotspots [196]	-	-	-	-	-	-	-	0.047
ResNet MLP [95]	72.5	21.4	-	81.9	51.9	68.3	58.8	-
COHESIV [237]	71.5	28.3	35.2	81.2	68.0	67.2	71.5	0.013
Ours	85.3	69.9	77.1	90.1	89.4	80.3	89.7	0.167

SAPIEN is a simulator which contains a large scale set of articulated objects. We use the renderings provided by 3DADN and the same evaluation strategies.

- (InteractionHotspots [196]): While 3DADN and SAPIEN can detect articulated objects as well as their axes, they cannot tell the affordance. InteractionHotspots learns affordance from watching OPRA [68] or Epic-Kitchen [47] videos. Since InteractionHotspots cannot detect objects, we apply a center crop of the input image based on the query point, and resize it to the standard input shape (224, 224). We use the model trained on Epic-Kitchen as it transfers better than OPRA.

Additionally, we want to test alternative network architectures trained on our 3DOI dataset. We use the same loss as ours to train it on 3DOI, to ensure fair comparison.

- (ResNet MLP [95]): ResNet MLP uses a ResNet-50 encoder to extract features from input images. We then sample the corresponding spatial features from the feature map using the 2D coordinates of keypoints. We train ResNet MLP on all tasks except mask, affordance and depth, as these tasks requires dense predictions for each pixel. Adding a separate decoder to ResNet makes it a UNet-like architecture [228], which is beyond the scope of ResNet.

- (COHESIV [237]): We also pick another model COHESIV, which designed for the prediction-at-a-query-location problem. Given an input image and corresponding hand location as a query, COHESIV predicts the segmentation of hands and hand-held objects. We adopt the network, as it produces a feature map of queries. we sample an embedding from the feature map according to the query point, concatenate it with image features, and produce multiple outputs.

5.6.2 Results

First, we show qualitative results in Figure 5.4. For articulated objects (drawers, cabinets, etc.), our approach can recognize its location, kinematic model (rotation or translation), axes

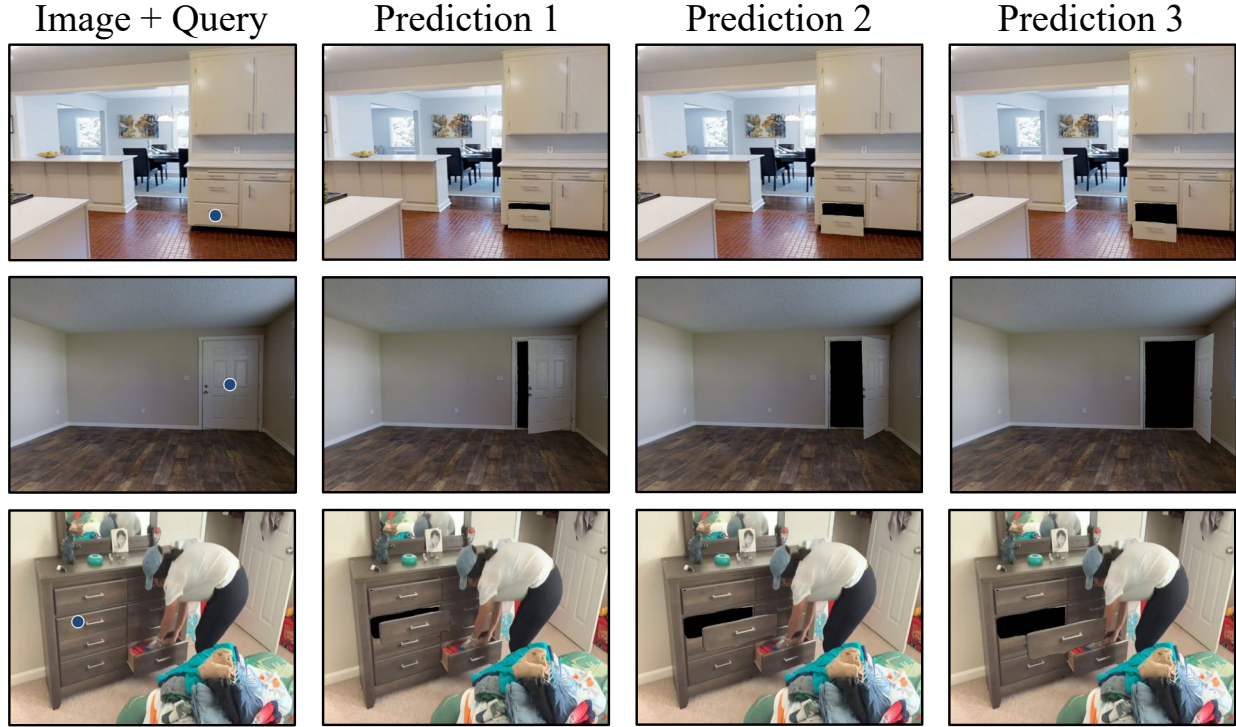


Figure 5.5: Prediction of 3D potential interaction of articulated objects. In prediction 1, 2, and 3, we rotate the object along its rotation axis, or translate the object along its normal direction.

and handle. It can also recognize rigid or nonrigid objects, as well as light or heavy ones. It works on both third-person images or egocentric videos. And all of these are achieved in a single model. For articulated objects, we utilize the outputs and further show their potential 3D interaction in Figure 5.5. Full details in supplemental.

We then compare our approach with a set of baselines. The quantitative results are reported in Table 5.1. 3DADN [211] is much worse than our approach, since it can only detect objects which are being articulated. It fails to detect objects humans are not interacting. Instead, our approach can detect any objects can be interacted, regardless of human activities. SAPIEN is worse than 3DADN, which suggests learning from synthetic objects has a huge domain gap. This is consistent with the observation of 3DADN. Visual comparisons are shown in Figure 5.6.

We compare our prediction of the affordance map with InteractionHotspots [196]. Our approach outperforms InteractionHotspots significantly, with a 75% improvement. A visual comparison is shown in Figure 5.7. While InteractionHotspots predicts a cloud-like probability map, our approach is typically very confident about its prediction. However, the overall performance is relatively low, mainly due to ambiguity of affordance on deformable objects.



Figure 5.6: Comparison of 3DADN [211], SAPIEN [287] and our approach. 3DADN has a strong performance when humans are present. However, it has difficulty detecting objects without human activities. SAPIEN does not generalize well to real images. However, it is sometimes better than 3DADN when humans are not present.

To explore alternative network architectures, we compare our approach with ResNet MLP [95] and COHESIV [237], which are trained on our data with the same loss functions. ResNet MLP is reasonable on movable, rigidity, and action. It is especially bad on bounding box localization, which is why we typically rely on a detection pipeline such as Mask R-CNN [94]. COHESIV learns reasonable bounding boxes and masks, which is a huge improvement over ResNet MLP. The performance of movable drops compared with ResNet MLP, while that of kinematic and action improves. Overall, our approach outperforms both ResNet MLP and COHESIV, mainly due to the introduction of transformers.

Finally, we evaluate depth on our data and EDINA [56]. Having state-of-the-art depth estimation is orthogonal to our goal, since we only need reasonable depth to localize objects in 3D and render potential 3D interactions. In fact, state-of-the-art depth estimation models are trained on over ten datasets and one million images [217, 303, 62], while our dataset only has 5K images with depth ground truth. We just report the evaluation of depth estimation, in order to show our model has learned reasonable depth. On our data, 84.2% pixels are within the 1.25 threshold, 98.5% pixels are within the 1.25^2 threshold. On EDINA, 65.4% pixels are within the 1.25 threshold, 95.9% pixels are within the 1.25^2 threshold.

5.6.3 Generalization Results

To test whether our approach and models trained on our 3DOI dataset can generalize, we further evaluate our approach on WHIRL [6], a robotics dataset manipulating everyday objects. Since WHIRL is a small-scale dataset, we test our model on WHIRL without finetuning. Our results are shown in Figure 5.8. For both articulated objects and deformable

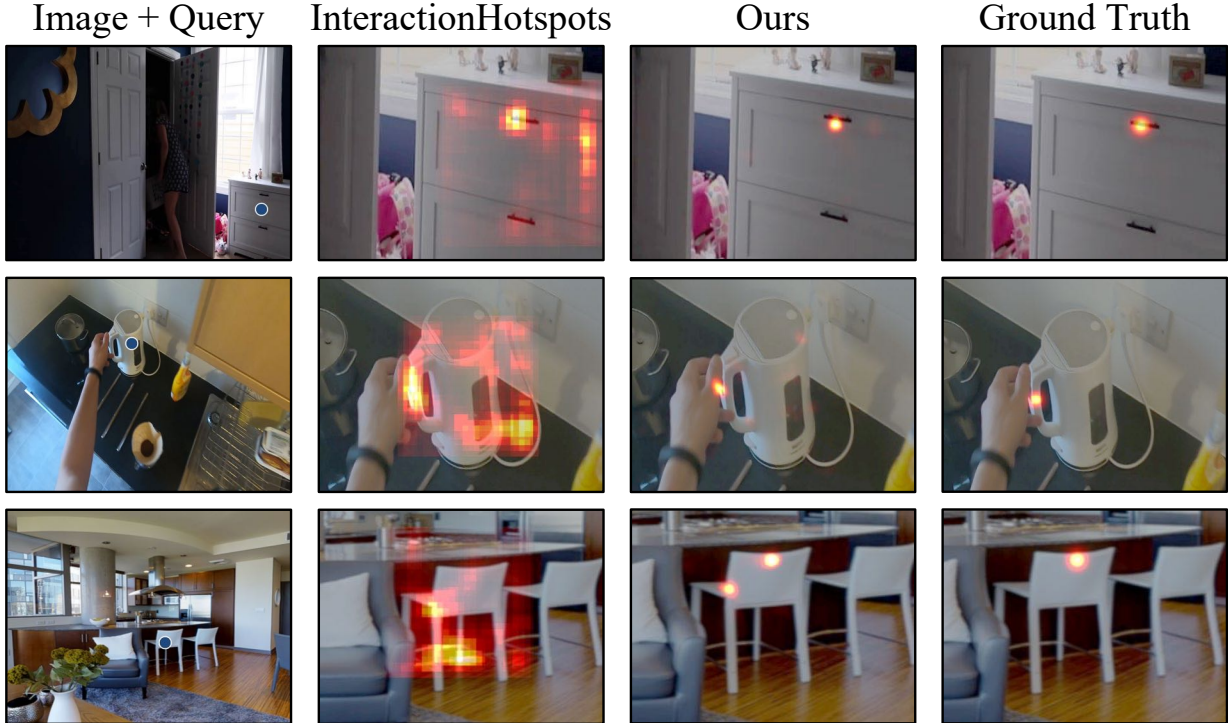



Figure 5.7: Comparison of InteractionHotspots [196] and our approach. We find InteractionHotspots typically makes a cloud like probability map on our data. Our model is very confident about its prediction, while there can be multiple solutions. Prediction and GT are zoomed manually for better visualization. Affordance colormap: min  max.

objects, our approach can successfully recover its kinematic model, location and affordance.







We also quantitatively evaluate our approach on WHIRL. We report our results in Table 5.2. Similar to our 3DOI dataset, our approach outperforms 3DADN [211], SAPIEN [287] and InteractionHotspots [196] significantly. The performance gap is even larger. We believe it is because humans are not present in most images of the dataset.

We compare our approach with ResNet MLP [95] and COHESIV [236], which are also trained on our 3DOI dataset. Compared with ResNet MLP, our approach is slightly worse on Movable and Axis, while much better on Rigid, Kinematic Action, and Box. This is probably due to the noise of a small-scale dataset. Compared with COHESIV, our model outperforms it consistently, although the improvement on box is small. It illustrates models trained on our 3DOI dataset generalize well to robotics data, regardless of network architectures.

5.6.4 Limitations and Failure Modes

We finally discuss our limitations and failure modes. In Figure 5.9, we show some predictions are hard to make from visual cues: Some articulated objects are symmetric and

Table 5.2: Quantitative results on robotics data [6]. Cat. means category. We report accuracy for all category classification, including movable, rigid, articulation and action. We report mean IoU for the boxes and masks, EA-Score for articulation axis, and SIM for affordance probability map. For all metrics, the higher the better.

Methods	Movable 	Localization 		Rigidity 	Articulation 		Action 	Affordance 
	Cat.	Box	Mask	Cat.	Cat.	Axis	Cat.	Probability
3DADN [211]	-	13.8	10.1	-	53.3	4.03	-	-
SAPIEN [287]	-	9.14	6.15	-	51.1	0.0	-	-
InteractionHotspots [196]	-	-	-	-	-	-	-	0.045
ResNet MLP [95]	88.8	14.1	-	80.0	51.1	57.1	51.1	-
COHESIV [237]	86.7	37.1	38.7	82.2	73.3	66.1	73.3	0.015
Ours	91.1	68.7	70.2	95.6	80.0	68.5	84.4	0.148

humans rely on common sense to guess its rotation axis. There are also hard examples when predicting the rigidity and movable. Finally, we only annotate a single keypoint for each object instance as affordance. But some objects may have multiple keypoints as affordance.

5.7 Conclusion

We have presented a novel task of predicting 3D object interactions from a single RGB image. To solve the task, we collected the 3D Object Interaction dataset, and proposed a transformer-based model which predicts the potential interactions of any objects according to query points. Our experiments show that our approach outperforms existing approaches on our data and generalizes well to robotics data.

Our approach can have positive impacts by helping build smart robots that are able to understand the 3D scene and manipulate everyday objects. On the other hand, our approach may be useful for surveillance activities.

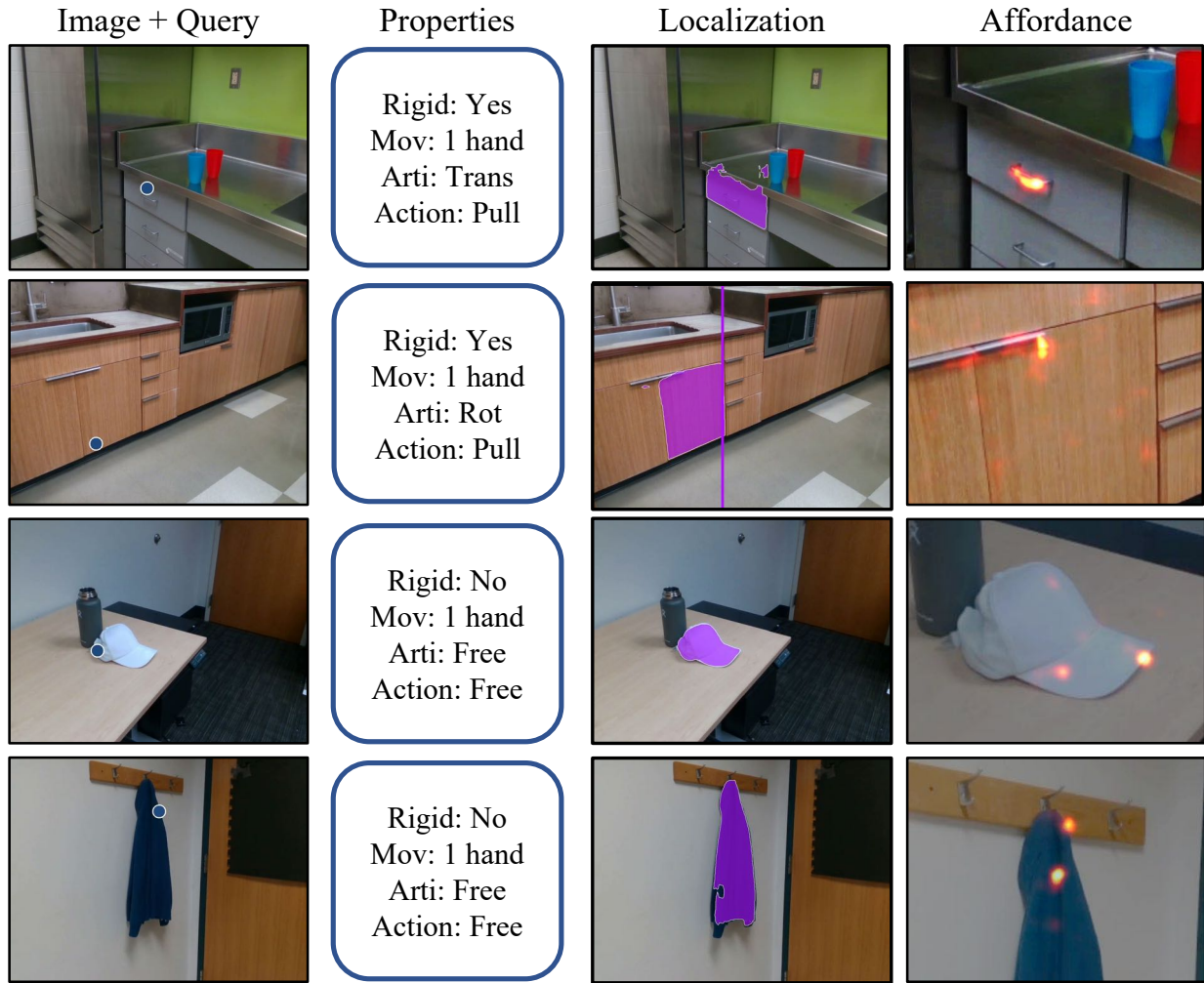



Figure 5.8: Results on robotics data [6]. Without finetuning, our approach generalizes well to robotics data, which indicates its potential to help intelligent agents to better manipulate objects. Row 1 and 2 are articulated objects. Row 3 and Row 4 are deformable objects. Affordance is zoomed manually for better visualization. Affordance colormap: min  max.

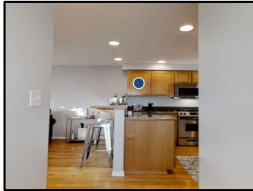
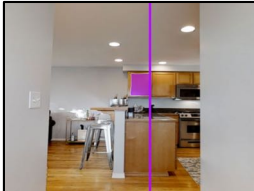
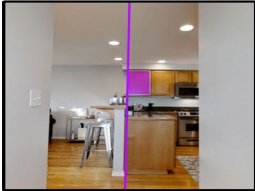
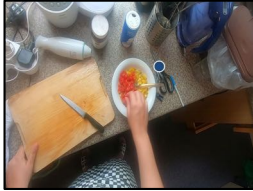


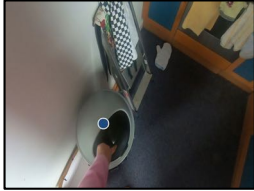
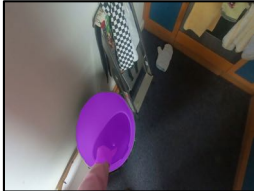

Image + Query	Properties		Localization	
	Prediction	GT	Prediction	GT
	Rigid: Yes Mov: 1 hand Arti: Rot Action: Pull	Rigid: Yes Mov: 1 hand Arti: Rot Action: Pull		
	Rigid: Yes Mov: 1 hand Arti: Free Action: Free	Rigid: Yes Mov: 1 hand Arti: Free Action: Free		
	Rigid: Yes Mov: 1 hand Arti: Free Action: Free	Rigid: Yes Mov: 2 hands Arti: Free Action: Free		

Figure 5.9: Typical failure modes of our approach. **Row 1:** Our predicted rotation axis is on the wrong side when the objects look symmetric. **Row 2:** Our predicted mask is partial when the scissors are occluded. **Row 3:** Our model thinks the trash bin can be picked up by 1 hand, potentially since its material looks plastic.

CHAPTER 6

Grounding Affordance from Vision Language Models

Affordance grounding refers to the task of finding the area of an object with which one can interact. It is a fundamental but challenging task, as a successful solution requires the comprehensive understanding of a scene in multiple aspects including detection, localization, and recognition of objects with their parts, of geo-spatial configuration/layout of the scene, of 3D shapes and physics, as well as of the functionality and potential interaction of the objects and humans. Much of the knowledge is hidden and beyond the image content with the supervised labels from a limited training set. In this chapter, we make an attempt to improve the generalization capability of the current affordance grounding by taking the advantage of the rich world, abstract, and human-object-interaction knowledge from pretrained large-scale vision language models [163]. Under the AGD20K benchmark, our proposed model demonstrates a significant performance gain over the competing methods for in-the-wild object affordance grounding. We further demonstrate it can ground affordance for objects from random Internet images, even if both objects and actions are unseen during training. The material in this chapter is derived from [208].

6.1 Introduction

Grounding affordance from a single image is a fundamental problem in computer vision. It forms the stepping stone to downstream tasks such as understanding human-object interaction [27, 78, 236], visual navigation [139], and object manipulation [6, 100]. Past approaches generally use human demonstrations as supervision to advance this field with tremendous success [196, 177, 176, 147]. While such approaches perform well on objects and actions seen during training, they struggle when generalizing in the wild, i.e. on novel objects unseen during training (Fig. 6.1).

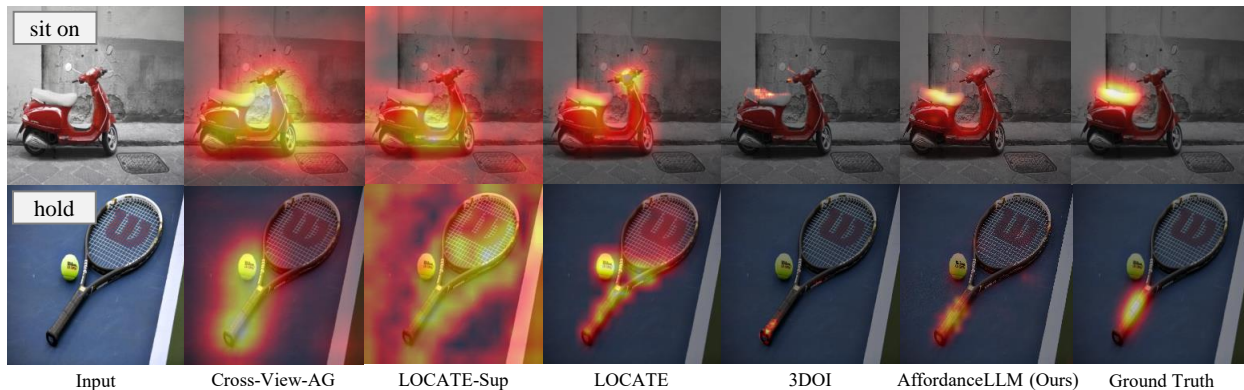


Figure 6.1: **Illustration for the affordance grounding task.** The input is a single image and the corresponding action (e.g., “hold”). The output is a heatmap which highlights regions one can interact. We aim to enhance the generalization capability of affordance grounding to in-the-wild objects that are unseen during training, by developing a new approach, AffordanceLLM, that takes the advantage of the rich knowledge from large-scale vision language models [163] beyond the supervision from the training images.

The difficulties in generalization arise from the fact that affordance grounding is a challenging task that requires comprehensive understanding of an image from multiple aspects. A successful solution requires an understanding of the 3D geometry and functionality of objects and parts, of the actions and intentions of the executing agent, of the potential interaction between object and human, as well as of the spatial configuration of the environment. Much of this knowledge lies beyond the ground-truth localization/recognition of objects and parts provided as heatmaps in a limited training set.

In this chapter we make attempts to improve affordance grounding in the wild by leveraging the rich world, abstract, and human-object-interaction knowledge embedded in large-scale Vision Language Models (VLMs). With large-scale text pretraining, modern VLMs such as GPT-4 [201], LLaVA [163] and Blip-2 [148] have a rich reservoir of world knowledge, as demonstrated by their extraordinary capabilities in answering visually grounded common sense questions [17]. World knowledge is instrumental to affordance reasoning — when presented with an image of a motorcycle and questioned about “How do I ride with this motorcycle?”, LLaVA answers “To ride the motorcycle, you should interact with the handlebars...” (Fig. 6.2), which exhibits commonsensical understanding of affordance. Affordance models equipped with similar world knowledge have a better chance generalizing to the wild than a model that purely learns from limited affordance demonstration data.

Beside world knowledge, another novel factor we introduce to improve affordance reasoning is 3D geometry, as it holds rich information of object functionality. A cylindrical area, for example a handle or a stick, is closely related to the action of grabbing or holding, regardless

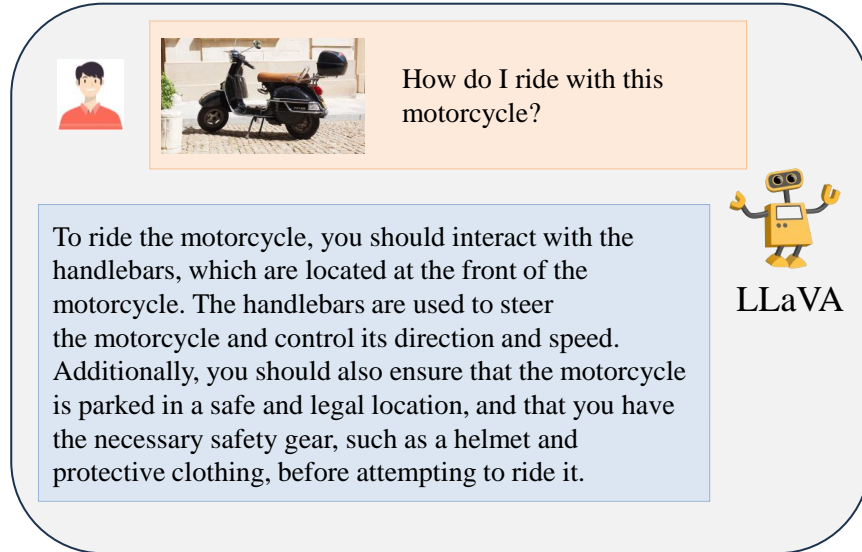


Figure 6.2: State-of-the-art vision language models, such as LLaVA [163], has rich human-object-interaction knowledge, thanks to the large-scale text pretraining. Given a question about how to interact with an object, it typically gives a reasonable solution.

of its color or texture. Similarly, a flat part, for example the surface of a chair or a bench, might indicate areas suitable for sitting or lying. Relating 3D geometries to actions allows us to bypass the difficulties in handling variations in visual appearances, and thus facilitates generalization.

We propose a novel approach, *AffordanceLLM*, that reflects the above intuitions. Our approach builds upon a VLM backbone (LLaVA [163]) to tap into the its world knowledge. We achieve it by extending the backbone with a mask decoder and a special token `<mask_token>`, which are used to predict an affordance map. The whole model can be trained end-to-end. Additionally, we introduce depth maps as 3D information in parallel to RGB images as input to our network, with the goal of eliciting geometric reasoning capability from the network. We found both designs significantly improve performance.

We evaluate our method on the AGD20K [177] benchmark, as this is the only large-scale affordance grounding dataset with accurate action and object labels. We re-split the benchmark to test capability of models to generalize to object categories unseen during training. We show that our approach outperforms all state-of-the-art baselines by a large margin. We take a further step to validate the generalization ability by testing our approach on random Internet images. It produces reasonable affordance maps on object categories very different from the ones in training set. Moreover, it even possesses some capability of generalizing to completely novel actions.

In summary, our contributions are as follows:

1. We introduce the first-ever affordance grounding approach that leverages the rich world knowledge embedded in pretrained VLMs, enabling the model to generalize beyond training data;
2. We demonstrate the importance of 3D information in affordance grounding;
3. Our proposed approach generalizes to novel objects and outperforms all state-of-the-art approaches on AGD20K. It even shows evidence that it could generalize to novel actions.

6.2 Related Work

Eliciting World Knowledge from Vision Language Models. Foundational Vision Language Models that bridge images and language have a rich reservoir of world knowledge, and recent researches have been tapping into it to make advancement in vision tasks. The joint visual-language embedding space learnt from simple image-text pair [214, 54] has made it possible to improve open-world detection [191, 167, 181, 239], and segmentation [271, 175, 146, 290]. The world knowledge here is the correspondence between visual and language concepts.

Large language models (LLMs) trained on trillions of tokens contain even richer world knowledge and are capable of answering common-sense questions. Coupled with vision inputs, the resulting multi-modal LLMs are brought in to solve complex vision problems. For example, Kosmos-2 and Groundhog incorporate the reasoning skill of LLMs to generate bounding boxes and segmentation masks [202, 313, 143]. 3D LLM further extend LLMs to reason about 3D scenes, including visual grounding and navigation [295, 319, 44, 99, 102, 154, 101]. For robotics, PaLM-E and RT2 transfer the knowledge from visual-language domains into motion planning and manipulation [59, 14]. Our approach embodies the same idea to transfer the world knowledge from VLMs, but applies it on a novel setting – solving visual affordance grounding.

Affordance Grounding. Understanding object affordance from a single image is an important step towards embodied visual intelligence, and researchers have built many different approaches to endow machines to have this ability. Nagarajan *et al.* first proposes to ground object affordance from Internet videos [196]. Fang *et al.* constructs an object affordance dataset based on product review videos [68]. Luo *et al.* annotates the first large-scale affordance dataset and call it AGD20K [177]. LOCATE [147] is the state-of-the-art approach on AGD20K. More recently, researchers further extends the scope of the affordance grounding

problem, including extending it to scene understanding [209, 136, 28], 3D models [296], ego-centric videos [195], hand pose generation [116, 299], or associating it with human parts [178]. We use AGD20K as our primary benchmark, and compare our approach with state-of-the-art methods [209, 196, 176, 88, 147].

Incorporating 3D Information for Vision Tasks. 3D information has been shown to be critical in certain vision and robotics tasks. For example, Zhou *et al.* [317] found that visual navigation in mobile sensorimotor systems can benefit from 3D input. Kerr *et al.* [128] found the NeRF-rendered depth map can help grasping in robotics. Similarly, grounding affordance could benefit from 3D information as well, as 3D shapes of objects and their parts hold a lot of hints on their utility and the proper ways to interact with them. While 3D information is not usually available for an arbitrary image, fortunately, researchers have built a series of robust approaches to estimate the 3D of an image, ranging from surface normal estimation [274, 63], depth estimation [33, 152, 217, 303, 302], to 3D reconstruction [79, 159, 198] and few-image NeRF [165, 212, 20, 305]. In our chapter, we mainly use DPT [217] to generate pseudo depth maps to help VLMs to build 3D understanding.

Robotics Manipulation. Manipulation of in-the-wild objects is an important but challenging task in robotics due to the difficulty of data collection. Researchers have developed many methods for different objects in different scenes, such as tabletop objects [113, 81, 242] and mobile manipulation [301]. While manipulation is not our goal, learning affordance can be a solution for manipulation [7, 100, 282].

6.3 Approach

We now introduce our approach, AffordanceLLM, which takes a single image I and an affordance text query T , and generates an affordance map M . We use a template of “What part of the <object_name> should we interact with in order to <action_name> it?” as the text query T . We then train LLM to generate a special token <mask.token> and use its hidden state to decode a dense affordance map M . A brief overview is shown in Fig 6.3.

6.3.1 Overview

Large language model. We choose LLaVA-7B as our backbone multimodal large language model. We refer the reader for a fuller explanation in [163], but briefly, LLaVA contains an image encoder, a text tokenizer, and a large language model LLM. The image encoder is typically a CLIP pretrained ViT, with a linear layer to project the hidden dimension. It encodes the image I into image features F_I . At the same time, the tokenizer encodes the

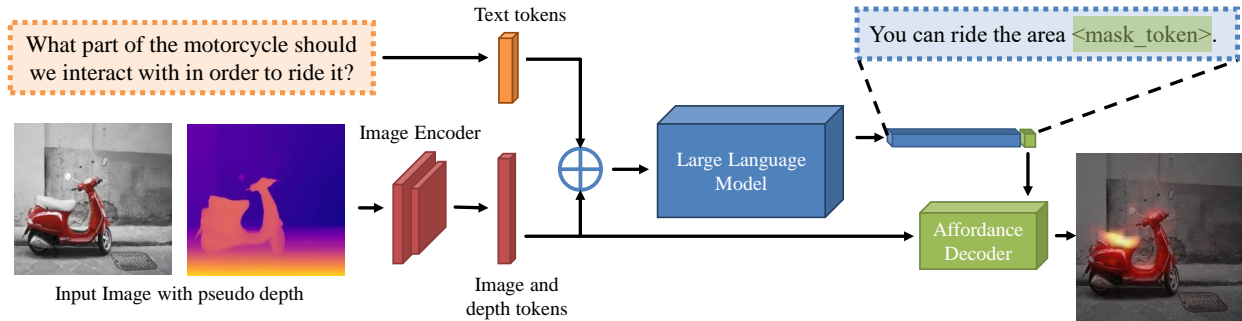


Figure 6.3: **Overview of AffordanceLLM.** The inputs of our model includes a single image and a text prompt related to interaction. We use OWL-ViT [191] as the image encoder to generate image features and project it into the same hidden dimension as the large language model. As well, we use a tokenizer to encode the text prompt. The text features and image features are concatenated together and feed into the LLM. The LLM is fine-tuned to predict a special token, which is used as a query to the mask decoder to generate the final affordance map.

text T into text features F_T . They are concatenated and feed into the language model. The LLM produces text output A as:

$$A = \text{LLM}(F_I, F_T). \quad (6.1)$$

Predicting affordance. How do we perform affordance reasoning while leveraging the world knowledge embedded in LLM? We propose to treat affordance as an implicit text token predicted from the LLM, which could be further decoded into a 2D map. Specifically, we train the LLM to predict a special token `<mask_token>`, the hidden state of which is first projected into a query embedding q and then fed into a Decoder to generate a dense affordance map. Decoder shares a similar architecture as the ones in SAM [131] and 3DOI [209]. It takes in q and image features F_I to produce an affordance map M , i.e.,

$$M = \text{Decoder}(F_I, q). \quad (6.2)$$

Pseudodepth as additional inputs. Besides images, the affordance reasoning task could benefit from 3D information (we will validate the benefits in Sec 6.4). However, modern VLMs are typically only trained with text and 2D images [163, 148]. Therefore, we also include a pseudo depth map as additional inputs to the large language model. For each image, we use the state-of-the-art depth estimation model DPT [217] to generate a pseudo depth map D . We use the same image encoder to encode the depth map D to produce depth

features F_D , and concatenate it with image features. Our final model is thus

$$A, M = \text{AffordanceLLM}(F_I, F_D, F_T). \tag{6.3}$$

Training objectives. Following the same practice as [209], we train the affordance map using a binary focal loss [156], and call it affordance loss L_{aff} . We set the weight of positive examples to be 0.95 and that of negative ones to be 0.05 to balance positives and negatives, as there are more negatives than positives in ground truth affordance map. We follow the standard cross entropy loss for the text output of language models. Our final loss function is a linear combination of affordance loss and text loss, given by

$$L = L_{\text{aff}} + \lambda \cdot L_{\text{text}}. \tag{6.4}$$

In practice, we set $\lambda = 0.01$ to balance two losses, as the affordance loss can be quite small due to the imbalance of positive and negative values.

6.3.2 Network Architecture

Next, we discuss the network architecture, and the training details of our model.

Image encoder. The standard LLaVA uses CLIP image encoder and a linear projection layer [290, 163]. In practice, we find that the CLIP image encoder has low resolution (224x224) and does not capture sufficient information about grounding. Therefore, we use OWL-ViT [191] to replace the standard CLIP-ViT [214]. OWL-ViT has an input resolution of 768x768, which is significantly higher than CLIP. At the same time, OWL-ViT is pre-trained to extract features that include precise location information of objects. As we will empirically show in experiments, using OWL-ViT is significantly better than CLIP. However, we note that our approach is general, and will benefit from any future improvements in pretrained VLM backbones.

Projection. Another problem of using OWL-ViT is about its embedding space. With a much higher input resolution, OWL-ViT produces 576 tokens with a hidden dimension of 768 for each image. In comparison, CLIP only produces 256 tokens for each image. Projecting each individual token into the hidden dimension of LLM (4096) consumes a lot of GPU memory. In practice, we project each token of OWL-ViT to 1024, and concatenating four neighboring tokens into a single token.

Language model. We follow LLaVA [163] and LLama [264] to use the standard text tokenizer to encode our text query. We use LLama-7B [264] as the large language model.

Affordance decoder. We aim to keep a lightweight decoder, as it has been proved to produce good segmentation masks and affordance maps [131, 209, 21, 36]. However, we find the vanilla mask decoder is too lightweight in our case and does not produce high-resolution affordance map. Therefore, we add an additional transposed convolution layer to increase its output resolution.

Implementation. We implement our model using PyTorch and HuggingFace. We initialize our model with LLama-7B pretrained weights. Following LLaVA [163], we freeze the image encoder, pretrain the image projection layer to align OWL-ViT and LLama features, and then use GPT instructions to tune the language model. Finally, we add the mask encoder [131, 209] and tune the whole model on AGD20K [177], which has annotations of object affordance. We use eight NVIDIA A100 (40GB) to train our model, with Fully Sharded Data Parallel. We use a batch size of 4 and set the learning rate as 2e-5.

6.4 Experiments

In experiments, we aim to systematically evaluate the performance of our approach. In particular, we are interested in answering these questions: (1) How well does it generalize, compared with state-of-the-art methods? (2) How does each design choice contribute to the final performance, including prompts, visual encoders, and depth?

6.4.1 Experimental Setup

Metrics. We evaluate primarily on AGD20K [177] and follow its metrics to evaluate our model, which is KLD, SIM and NSS [196, 177, 209, 147]. For KLD, the lower the better. And for SIM and NSS, the higher, the better. A full explanation is available in the supplemental.

Baselines. We compare our approach against state-of-the-art baselines. In general, affordance grounding methods belong to two categories: weakly supervised and fully supervised methods. We report performance of both categories.

(Weakly supervised methods): They do not train on explicit labels of the affordance map. Instead, they are trained on a human demonstration of the same object. These approaches include InteractionHotspots [196], Cross-View-AG [177], Cross-View-AG+ [176], AffCorrs [88], and LOCATE [147]. Among them, LOCATE is the most recent model and has the best results on AGD20K. We use the reported number in LOCATE for the easy split and retrain them for the hard split. Among them, we cannot run AffCorrs, as it focuses on one-shot affordance learning. The reported model on the easy split is adapted by [147] and not publicly available. We also do not run InteractionHotspots because the pretrained model only sup-

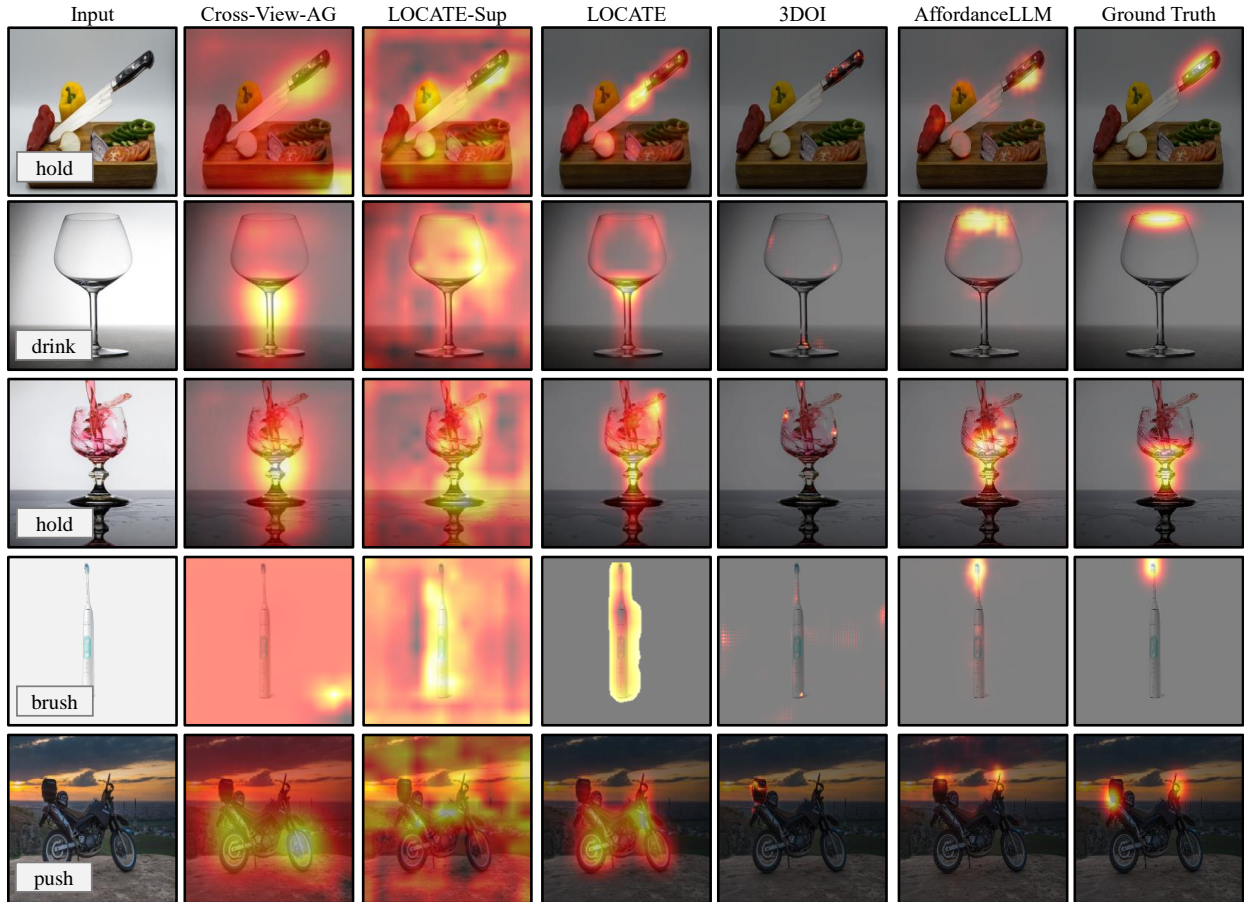


Figure 6.4: Qualitative results on the test set of the hard split. LOCATE-Sup fails to learn a reasonable affordance map due to limited training data. LOCATE [147] typically predicts an affordance map which covers the whole object. 3DOI [209] focuses on a small area of the object. Overall, our approach produces the best-quality affordance predictions.

ports 7 actions. The reported model is retrained by [177] but lacks sufficient implementation details to be reproduced. Therefore, we retrain Cross-View-AG [177], Cross-View-AG+ [176], and LOCATE [147] on the hard split. We maintain the object/action split, but allow them to use more images for weak supervision. Therefore, they have 11,889 images for training.

(Fully supervised methods): Affordance map can also be learned from explicit labels, and we call it supervised methods. This includes 3DOI [209] and ours. We also adapt LOCATE to a fully supervised version for fair comparison.

- *3DOI [209]:* 3DOI is a SAM-based model [131], which takes a single image and a query point and predicts the segmentation mask and affordance map. Therefore, We randomly sample a pixel with score > 0.9 as the query point from the affordance map. We use the 3DOI pretrained model, which has never seen any images in AGD20K, including the training set.

Table 6.1: Difficulty score of different splits. The lower the score, the more similar are the object categories in the train and test set.

Splits	Same	Easy	Hard	Random
Difficulty Score \uparrow	0.000	0.356	0.412	0.491

Table 6.2: **Quantitative results on the *Easy* split of AGD20K [177]**. InteractionHotspots, Cross-View-AG(+), AffCorrs and LOCATE are trained on AGD20K images with weak supervision (13,323 images). LOCATE-Sup and LOCATE-Sup-OWL, and AffordanceLLM are trained on AGD20K images with dense annotation (1,135 images). 3DOI is trained on their own dataset with dense annotation (10,000 images) [209]. AffordanceLLM is comparable to LOCATE [147] on the easy split, where test objects have similar counterparts in the training set. The **best** and second-best results are highlighted in bold and underlined, respectively.

Methods	KLD \downarrow	SIM \uparrow	NSS \uparrow
InteractionHotspots [196]	1.994	0.237	0.577
Cross-View-AG [177]	1.787	0.285	0.829
Cross-View-AG+ [176]	1.765	0.279	0.882
AffCorrs [88]	1.618	0.348	1.021
LOCATE [147]	1.405	<u>0.372</u>	1.157
LOCATE-Sup [147]	1.907	0.236	0.641
LOCATE-Sup-OWL [147, 191]	1.927	0.234	0.624
3DOI [209]	3.565	0.227	0.657
AffordanceLLM (Ours)	<u>1.463</u>	0.377	<u>1.070</u>

- *LOCATE-Sup [147]*: To ensure fair comparison, we also adopt LOCATE and train it using the same binary focal loss as our model. We call it LOCATE-Sup. LOCATE uses a Dino-ViT as its visual encoder [23]. To eliminate the effect of different pretrained visual encoders, we also report the performance of LOCATE-Sup-OWL, which uses the exact same pretrained visual encoder as ours.

6.4.2 Dataset

We follow LOCATE [147] to evaluate primarily on AGD20K [177], as it is the only large-scale dataset for affordance with action and object labels. Since our approach is not weakly supervised and requires dense annotations, we only use AGD20K images with dense annotations.

In this chapter, we primarily evaluate the ability of an affordance model to generalize to unseen object categories, and thus evaluate on the *Unseen* split of the AGD20K benchmark. This split ensures that there is no overlap between the object categories in the train and test

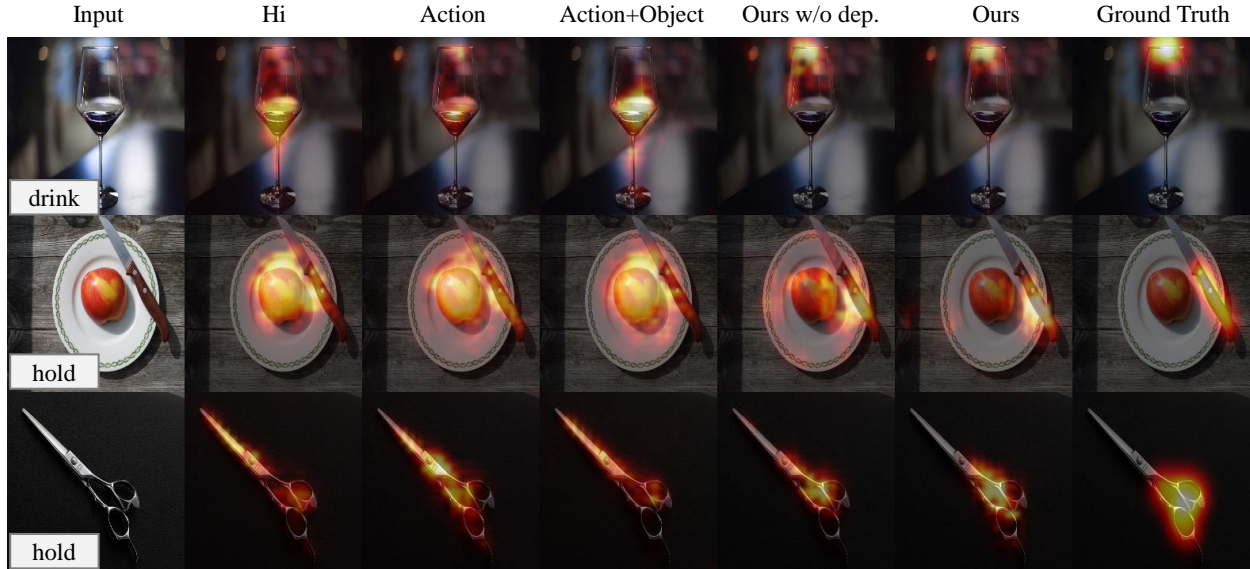


Figure 6.5: Ablation of different text prompts and depth. Ours w/o depth is our approach without pseudodepth as additional inputs. Ours is our full approach. We find constructing the correct text prompt typically helps our model to focus on the correct area. We believe it is because the correct text prompt would activate the world knowledge related to affordance embedded in the VLM.

set.

However, the *Unseen* split has one major issue: there are still a lot of similarities between the objects in the train and test set. Objects in the test set typically have similar counterparts in the training set, leaving models room for memorizing. For example, “skis” in the test set maps to “snowboards” and “skateboards” in training set, “basketball” maps to “baseball”, “knife” maps to “fork”, etc. To make the benchmark more reflective of the generalization ability of a model, we provide a more challenging split. We thus have the following two splits.

Easy split. This is the original *Unseen* split of AGD20K. We have 1135/540 images for train and test with dense annotations for the fully supervised setting, or 13,323/540 images for the weakly supervised setting. The test set remains the same for both settings.

Hard split. We *randomly* put 50% AGD20K object classes into the training set and the remaining classes into the test set to simulate in-the-wild generalization (details in the supplemental). The training and test object are not overlapping, and most objects in the test set do not have a similar counterpart in the training set, and is thus harder to generalize than the *Easy* split. We have 868/807 images for train and test with dense annotations for the fully supervised setting, and 11,889/807 images for weakly supervised setting. The test set is the same for both settings.

Table 6.3: **Quantitative results on the *Hard* split of AGD20K [177]**. Cross-View-AG(+) and LOCATE are trained on AGD20K images with weak supervision (11,889 images). LOCATE-Sup and LOCATE-Sup-OWL, and AffordanceLLM are trained on AGD20K images with dense annotation (868 images). 3DOI is trained on their own dataset with dense annotation (10,000 images) [209]. On the hard split, AffordanceLLM outperforms all baselines by a large margin, which demonstrates the superior generalization ability of our model. We do not run InteractionHotspots [196] and AffCorrs [88], as the reported model has ambiguous implementation details, or is not publicly available. The **best** and second-best results are highlighted in bold and underlined, respectively.

Methods	KLD ↓	SIM ↑	NSS ↑
Cross-View-AG [177]	2.092	0.209	0.138
Cross-View-AG+ [176]	2.034	0.218	0.342
LOCATE [147]	<u>1.829</u>	<u>0.282</u>	0.276
LOCATE-Sup [147]	2.003	0.224	0.435
LOCATE-Sup-OWL [147, 191]	2.127	0.206	0.314
3DOI [209]	4.017	0.200	<u>0.549</u>
AffordanceLLM (Ours)	1.661	0.361	0.947

Measuring split difficulty. We propose a metric to quantify the generalization difficulty of a split. Intuitively, the difficulty to generalize to an object class in the test set is defined by how different it is from the classes in the training set, which could be measured by its semantic distance to the most similar class in the training set [210]. The greater the distance, the harder it is to generalize to this test class. Therefore, for each semantic class c in the test set, we compute its distance d to the most similar class in the training set. We use the CLIP [214] text encoder to obtain an embedding to represent each object class. Assume train classes are C_{train} and test classes are C_{test} , the difficulty of this split is

$$D(C_{\text{train}}, C_{\text{test}}) = 1 - \frac{1}{|C_{\text{test}}|} \sum_{c \in C_{\text{test}}} \max_{c' \in C_{\text{train}}} d(c, c'). \quad (6.5)$$

We compare the difficulty metric among four settings: (1) *Same*: train and test share the same classes; (2) *Easy* split; (3) *Hard* split; (4) *Random*: constructed by randomly even-splitting 50 object classes from LVIS [87], which serves as a lower bound.

We show the difficulty metrics in Tab 6.1. The *Same* split has a similarity metric of 0.0, as all object classes in the test are present during training. The *Easy* split has a similarity metric of 0.356. The *Random* split has a score of 0.491. The *Hard* split has a higher score than *Easy*, meaning that the difference between test and train is more significance in *Hard* than in *Easy*, and is thus harder to generalize.

Table 6.4: Ablation on the hard split. We validate the importance of text prompts, image encoders and pseudo depth to performance.

Depth	Text Prompt	Img Encoder	KLD ↓	SIM ↑	NSS ↑
Yes	Full	OWL-ViT	1.661	0.361	0.947
-	Full	OWL-ViT	1.713	0.352	0.881
-	Full	CLIP-ViT	1.759	0.286	0.776
-	Object, Action	OWL-ViT	1.769	0.329	0.827
-	Action	OWL-ViT	1.843	0.336	0.815
-	Hi	OWL-ViT	1.836	0.325	0.793

6.4.3 Results

Figure 6.4 shows qualitative results on the test set of the hard split. In this split, the objects in the test set bear little to none resemblance to the ones in the training set. We compare our approach, AffordanceLLM, with a set of state-of-the-art baselines. LOCATE [147] tends to predict an affordance map that covers the entire object, indicating poor generalization performance. 3DOI [209] typically focuses on a small area of the object, and sometimes fails to ground the correct region. LOCATE-Sup fails to predict reasonable affordance map, probably due to the small amount of training data. Despite being trained on the same training set as LOCATE-Sup, our approach is able to produce the best affordance map among all methods, showcasing superior generalization capability.

We further compare our model with baselines quantitatively and the results are summarized in Tab. 6.2, 6.3. On the *Hard* split, where the test set objects differ semantically from the training set, our method outperforms all baselines significantly. This improvement can be attributed to the extensive world knowledge and understanding embedded within the large language model. On the *Easy* split, our model is comparable to LOCATE [147] and outperform all other baselines. We hypothesize that the advantage of our approach is less pronounced when the test and train objects exhibit similarity, as the generalization capability becomes less critical. It is also worth noting that unlike LOCATE which is weakly supervised on 10k+ images, our model was fully supervised on some 1k images with dense annotations, which renders a more meaningful comparison with LOCATE-Sup that is trained on the same data. Our method significantly outperforms LOCATE-Sup on both splits, indicating the effectiveness of our approach.

6.4.4 Ablation

We conduct a few ablation studies to understand how different components of the model contribute to the final performance. We test different text prompts, different image encoders, and the effect of pseudo depth as inputs. The results are summarized in Tab 7.2 and Fig 6.5.

Text prompts. Prompt tuning is known to have major effects on large VLMs. We test four different text prompts to understand the effect of text content on model performance:

- *Hi*: We use “Hi” as our text prompt.
- *Action*: We use the action (e.g. “hold”) as the prompt.
- *Object + Action*: We use the object name and action label as our text prompt, for example “hold, knife”.
- *Full*: We use a complete question as the text prompt — “What part of the motorcycle should we interact with in order to push it?”.

We notice that the *Full* prompt yields a higher performance compared with other simple text prompts. It demonstrates that specific question prompt is helpful for extracting the knowledge from pretrained large language models.

Vision encoders. Although LLaVA [163] uses CLIP-ViT, it may not be the optimal vision encoder for our affordance grounding task — CLIP-ViT is trained with an objective to align text-image pairs and is not explicitly optimized to perform localization, and therefore has limited visual grounding capability. We therefore switch to OWL-ViT [191], which is trained on detection datasets with 2M images, and achieves state-of-the-art open vocabulary detection performance. As shown in Tab 6.4, using OWL-ViT as vision backbone far excels using CLIP-ViT. It indicates the importance of grounding capability of visual backbone.

6.4.5 Pseudodepth as Inputs

Our model is trained with pseudo depth map produced by DPT [217]. To verify whether the additional depth inputs are effective, we compare the model trained with and without estimated depth (Tab 6.4 and Fig 6.5). With depth, our model can predict better affordance map, demonstrating the importance of 3D information in affordance reasoning.

6.4.6 Generalization to Internet Images

We further test the generalization of our model on random Internet images in Fig. 6.6. All test objects are novel. To showcase how different these objects are from the ones in train set, for each test object, we retrieve the most similar object in the train set using the metric defined in Eq 6.5. As we can see, these objects are not similar to any objects in the

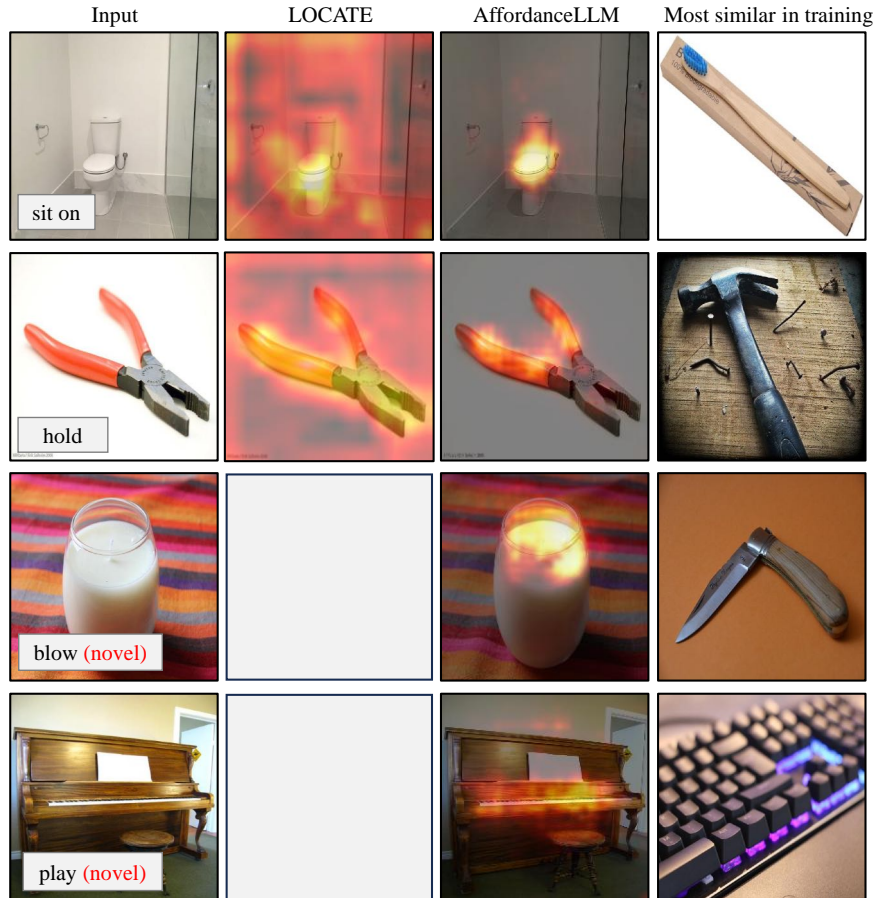


Figure 6.6: Generalization results on random Internet images. We show the most similar objects in the training set to demonstrate how different the objects are from the ones in the training set. **(Row 1, 2)**: AffordanceLLM generalizes to novel objects from random Internet images, while LOCATE [147] fails. **(Row 3, 4)**: AffordanceLLM generalizes to novel actions plus novel objects. LOCATE cannot infer novel actions thus we left it blank.

train set. We go even further to test if our approach is able to generalize to novel actions in addition to novel objects, such as “blow” and “play”. Generalization to novel actions is even more challenging, requiring open vocabulary understanding of actions, which is beyond the capability of LOCATE [147]. Despite these challenges, our approach not only produces very reasonable affordance maps for novel objects, but is also able to handle novel actions plus novel objects, once again demonstrating the extraordinary capability to generalize.

6.4.7 Failure Examples

Finally, we show our failure examples in Fig 6.7. First, we find AffordanceLLM fails on some ambiguous questions. For example, in AGD20K, “cut with” refers to the blade of a knife. However, AffordanceLLM thinks humans should hold the handle of the knife to cut

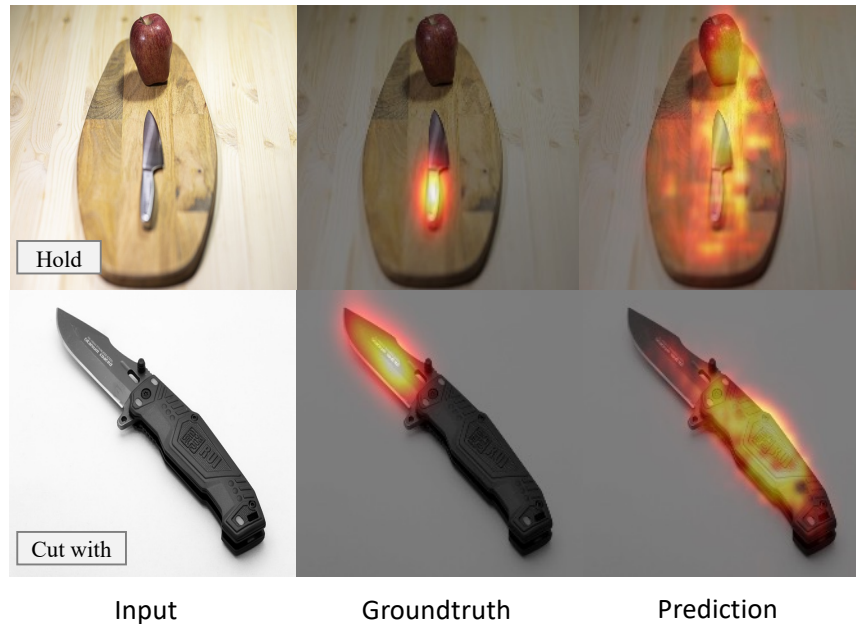


Figure 6.7: Failure examples. **(Row 1:)** AffordanceLLM sometimes fails due to multiple objects present in the scene and it fails to refer to the correct object. **(Row 2:)** AffordanceLLM thinks humans should hold the handle to cut something using the knife, while AGD20K annotators think “cut with” should refer to the blade.

anything. Second, when there are multiple objects in the image, it sometimes cannot refer to the correct object.

6.5 Conclusion

We have presented AffordanceLLM, a novel approach which can ground affordance for in-the-wild objects unseen during training. By tapping into the world knowledge embedded in a Vision Language Model, our proposed approach generalizes much better to in-the-wild objects, compared with state-of-the-art affordance grounding models.

Our approach can have positive impacts by helping build intelligence robots which can manipulate in-the-wild objects. On the other hand, it can be misused to cause physical damage or harm if applied in an adversarial manner.

CHAPTER 7

3D Multiview Pretraining for Robotic Manipulation

Recent works have shown that visual pretraining on egocentric datasets using masked autoencoders (MAE) can improve generalization for downstream robotics tasks [288, 197]. However, these approaches pretrain only on 2D images, while many robotics applications require 3D scene understanding. In this work, we propose 3D-MVP, a novel approach for 3D multi-view pretraining using masked autoencoders. We leverage Robotic View Transformer (RVT), which uses a multi-view transformer to understand the 3D scene and predict gripper pose actions. We split RVT’s multi-view transformer into visual encoder and action decoder, and pretrain its visual encoder using masked autoencoding on large-scale 3D datasets such as Objaverse. We evaluate 3D-MVP on a suite of virtual robot manipulation tasks and demonstrate improved performance over baselines. We also show promising results on a real robot platform with minimal finetuning. Our results suggest that 3D-aware pretraining is a promising approach to improve sample efficiency and generalization of vision-based robotic manipulation policies. The material in this chapter is derived from [213].

7.1 Introduction

Building learning-based manipulation systems is challenging due to the unavailability of diverse large-scale robotics data. To address this, there has been significant interest in using computer vision techniques to learn generalizable visual representations without robotics focused data, for example by self-supervised pre-training on image datasets. In particular, inspired by the success of masked language modeling in NLP, several recent works have explored masked autoencoding (MAE) for visual representation learning [92]. MAE learns to reconstruct randomly masked patches in an input image, encouraging the model to learn high-level semantic features. When applied to egocentric videos from human demonstrations,

MAE has been shown to learn representations that generalize well to downstream robotics tasks such as object manipulation [288, 215, 48].

However, current MAE approaches for robotics pretrain only on 2D images, ignoring the 3D structure of the scene. Prior works in robotics have shown that methods that build an explicit 3D visual representation of the environment are more sample efficient and generalize better than those with only 2D visual representations [81, 242, 207]. Hence, in this work, we explore how we could bring the benefits of visual pretraining to robot manipulation methods that reason with explicit 3D representations.

We propose 3D-MVP, a method for 3D Multi-View Pretraining for robot manipulation. Our approach builds upon recent advances in robot manipulation. Specifically, we use the Robotic View Transformer (RVT) [81], a state-of-the-art 3D manipulation method [81]. RVT takes as input a point cloud of the scene and builds a 3D representation of the scene, using a set of fixed orthogonal “virtual” RGBD images. These RGBD images are fed through a transformer model that fuses information across views and predicts robot actions in the form of future gripper poses.

We choose RVT over other methods for manipulation that build a 3D representation of the scene (e.g. PerAct [242] and Act3D [75]), because other methods use either voxels or point clouds as input a transformer model, while RVT uses orthogonal RGBD images. The view-based representation makes RVT a suitable candidate for MAE pretraining.

We pretrain the multi-view representation in RVT by attaching it to a lightweight MAE decoder. We then randomly mask out a subset of the visual tokens for each view and train the model to reconstruct the multiview RGB-D images. After the pre-training, we discard the decoder. We then fine-tune the visual encoder along with RVT’s action decoder on various manipulation tasks.

In order to learn generalizable and robust visual features, we use the recent works that have led to the creation of large-scale datasets of 3D scenes, such as Objaverse and 3D-FRONT [51, 74, 50, 216]. These datasets contain high-quality 3D scans of indoor environments along with realistic textures and materials. We use these datasets to create sets of orthogonal views that are similar to the 3D representation used in RVT. These sets of orthogonal views are then used for pretraining the visual encoder in RVT. We conduct experiments to ablate various different choices available for pre-training. Specifically, we study how masking strategies, dataset choice and dataset sizes affect the downstream manipulation performance.

Finally, we evaluate 3D-MVP on the RLBench benchmark [113], a suite of manipulation tasks in a simulated environment. We find that pretraining the RVT encoder with 3D-MVP leads to significant improvements over training from scratch or pretraining with 2D MAE.

These results inform how we can advance the state-of-the-art in robotic manipulation with the help of pretraining. We further evaluate 3D-MVP on the Colosseum benchmark [207], which tests a system’s generalization across various unseen variations of manipulation tasks like object size change, color change, and lighting changes. We find that the proposed 3D-MVP method is more robust across various variations than RVT trained from scratch.

In summary, our contributions are three-fold.

- We propose 3D-MVP, a novel approach for 3D multi-view pretraining using masked autoencoding on large-scale 3D datasets.
- We study how various design choices in pretraining, like masking strategy, dataset combination and sizes, affect downstream object manipulation performance.
- We demonstrate that pretraining with 3D-MVP leads to significant improvements on object manipulation tasks. We also show that 3D-MVP enables training policies that are more robust to variations such as size, texture, and lightning, on the COLOSSEUM benchmark.

We hope our work can inform future studies about pretraining for robotic applications.

7.2 Related Work

Our work builds upon several active areas of research, including self-supervised learning, visual pretraining for robotics, and learning robotic manipulation from demonstrations.

Self-supervised learning. Self-supervised learning aims to learn useful representations from unlabeled data by solving pretext tasks that do not require manual annotation. Early work in this area focused on designing pretext tasks for 2D images, such as solving jigsaw puzzles [200], contrastive learning [31, 93] or joint embedding approaches [3, 4, 22, 23, 84, 320]. Most related to our work is the masked autoencoder (MAE) approach proposed by He et al. [92], which learns to reconstruct randomly masked patches in an image. MAE has been shown to learn transferable representations for object detection and segmentation tasks. Furthermore, Bachmann et al demonstrates MAE pretraining can be extended to different modalities such as semantics and depth [5]. In this work, we extend the MAE approach to multi-view 3D scenes, enabling us to learn 3D-aware representations that are useful for robotic manipulation tasks.

Visual pretraining for Robotics. Visual pretraining has demonstrated impressive generalization ability on computer vision tasks. Therefore, prior works have explored whether it works for robotics tasks as well. Specifically, the robotics community has trended towards learning representations using state-of-the-art self-supervised vision algorithms on diverse interaction datasets [83, 236, 46], and finetune the network on robotics tasks [182, 197, 288, 215, 184, 48]. 3D-MVP follows the same procedure. However, existing robotics pretrain-

ing approaches typically learn a 2D visual encoder (e.g. ResNet [95] or ViT [58]), we find they are inferior than manipulation policies which do explicit 3D modeling (e.g. RVT [81], Act3D [75]). Migrating a pretrained ViT to 3D manipulation policies is nontrivial since they do not have a 2D visual encoder. In this chapter, we propose 3D-MVP, which does 3D-aware pretraining on 3D manipulation policies, to fill the gap.

Learning manipulation from demonstrations. Recent work has explored using transformers for multi-task manipulation policies that predict robot actions from visual and language inputs [242, 86, 162, 235, 247]. End-to-end models like RT-1 [15], GATO [220], and InstructRL [162] directly predict 6-DoF end-effector poses but require many demonstrations to learn spatial reasoning and generalize to new scenes. To better handle 3D scenes, PerAct [242] and C2F-ARM [114] voxelize the workspace and detect the 3D voxel containing the next end-effector pose. However, precise pose prediction requires high-resolution voxels which are computationally expensive. Recently, RVT [81] proposes a multi-view transformer that attends over point cloud features from multiple camera views to predict actions. This avoids explicit voxelization and enables faster training and inference than PerAct. Act3D [75] represents the scene as a continuous 3D feature field and samples points to featurize with attention, allowing adaptive resolution. GNFactor [309] jointly optimizes a generalizable neural field for reconstruction and a Perceiver for decision-making, using a shared 3D voxel representation enriched with semantics from a vision-language model. In contrast, our proposed 3D-MVP learns 3D scene representations through masked autoencoding pretraining on a large dataset of 3D object models. This pretraining enables 3D-MVP to build a rich understanding of 3D geometry and semantics prior to finetuning on downstream manipulation tasks. Compared to RVT and Act3D which train from scratch on target tasks, 3D-MVP’s pretraining leads to improved performance, sample efficiency and generalization. Unlike GNFactor which relies on a pretrained vision-language model to inject semantics, 3D-MVP directly learns 3D semantic features from object models.

7.3 Approach

In this section we first provide essential background on RVT, then define our method 3D-MVP that learns 3D-aware representations for robotic manipulation using masked autoencoding on multi-view 3D scenes, and finally describe how we finetune the method on downstream manipulation tasks. Figure 7.1 gives an overview of our approach.

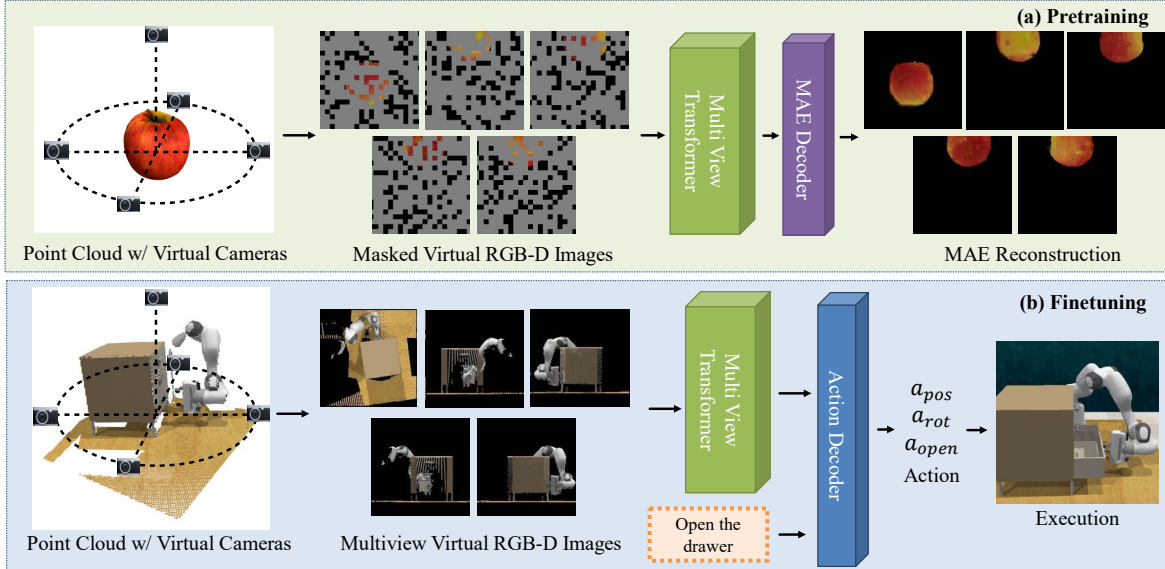


Figure 7.1: Overview of 3D-MVP. (a) We first pretrain a Multiview 3D Transformer using masked autoencoder on multiview RGB-D images. (b) We then finetune the pretrained Multiview 3D Transformer on manipulation tasks. Since the MVT is pretrained, the learned manipulation policy generalizes better. For example, it is more robust to changes of texture, size and lighting.

7.3.1 Background on Robotic View Transformer (RVT).

It is a state-of-the-art object manipulation method [81]. It creates an explicit 3D representation of the scene by using orthogonal virtual views of a scene. Please refer to Goyal et al. [81] for a full explanation. Here, we provide a brief overview and define the notation.

RVT takes a point cloud of the robot workspace as input (Fig. 7.1). RVT is agnostic to the poses of the RGBD cameras used to construct the input point cloud. For example, it can be obtained from a combination of third-person cameras around the workspace, head cameras, or wrist cameras. RVT then renders this point cloud using a set of five “virtual” cameras placed at orthogonal locations around the robot. The virtual cameras are placed at the top, left, right, front, and back of the robot workspace with respect to the robot. Each virtual image has 10 channels: RGB (3 channels), Depth (1 channel), 3D point coordinate in world frame (3 channels), and 3D point coordinate in camera sensor frame (3 channels). We denote the virtual images captured from different virtual camera poses $\{p_1, \dots, p_5\}$ as $\{I_1, \dots, I_5\}$.

These virtual images are then tokenized into N patch embeddings [58], flattened into a sequence of $5N$ tokens spanning all images, and fed to a multi-view transformer. The goal of RVT’s multi-view transformer is to learn a function f_θ that maps the virtual images as well as language instructions L to the 6-DoF end-effector pose and the gripper’s binary open

or close state:

$$a_{\text{pos}}, a_{\text{rot}}, a_{\text{open}} = f_{\theta}(L, I_1, p_1, \dots, I_5, p_5) \quad (7.1)$$

RVT is trained end-to-end from scratch on sampled trajectories from simulator or real robots. While RVT has shown state-of-the-art results on 3D manipulation, it does not generalize to novel objects and scenes. In the next section, we describe our novel approach 3D-MVP, and how we modify and pretrain the RVT encoder using 3D-MVP.

7.3.2 3D Multi-View Pretraining (3D-MVP)

Architecture change to RVT. The key idea is to pretrain the RVT visual encoder f_{θ} using masked autoencoding on large-scale 3D scene datasets. However, RVT’s multiview transformer f_{θ} is an end-to-end model that takes language instructions as input, and produces robot actions. Existing robotics data with language and actions is limited in terms of diversity of 3D scenes, and 3D scene datasets do not typically contain robotics annotations.

To enable pre-training on 3D scene datasets, we first split the multiview transformer f_{θ} into an input renderer \mathcal{R} , an encoder network \mathcal{E} and an action decoder network \mathcal{D} . The renderer \mathcal{R} maps the posed input images into the five virtual images, by constructing a point cloud and rendering it from the five views:

$$\{I_1, \dots, I_5\} = \mathcal{R}(I_1, p_1, \dots, I_5, p_5) \quad (7.2)$$

The encoder \mathcal{E} maps the virtual images into a latent embedding $z \in \mathbb{R}^{5N \times H}$ (where H is the hidden size) and the action decoder \mathcal{D} maps z to the robotic action space, i.e.,

$$a_{\text{pos}}, a_{\text{rot}}, a_{\text{open}} = \mathcal{D}(L, z), \quad z = \mathcal{E}(I_1, \dots, I_5), \quad (7.3)$$

where tokenization of the virtual images into $5N$ patch embeddings is subsumed into \mathcal{E} . Both encoder \mathcal{E} and decoder \mathcal{D} are multiview transformers. We keep the decoder lightweight to focus on pretraining of the encoder.

Pretraining encoder \mathcal{E} . Our visual pretraining focuses on learning a generalizable representation for the encoder \mathcal{E} . We extract point clouds from Objaverse and render the point cloud using the same five “virtual” cameras. Given 5 virtual images $\{I_1, \dots, I_5\}$, we randomly mask out a subset of the visual tokens for each view, and denote the masked inputs as $\{I'_1, \dots, I'_5\}$. We use the encoder to extract the embedding z from the masked inputs,

$$z = \mathcal{E}(\{I'_1, \dots, I'_5\}) \quad (7.4)$$

We use a separate, lightweight MAE decoder \mathcal{D}_{MAE} to reconstruct the original image $\{I_1, \dots, I_5\}$ from the embedding z .

$$\{\tilde{I}_1, \dots, \tilde{I}_5\} = \mathcal{D}_{MAE}(z) \quad (7.5)$$

The encoder \mathcal{E} and decoder \mathcal{D}_{MAE} are trained end-to-end using a pixel-wise reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \frac{1}{5WH} \sum_{i=1}^5 \sum_{p=1}^{W \cdot H} \|[I_i]_{(p)} - [\tilde{I}_i]_{(p)}\|_2^2, \quad (7.6)$$

where $[I]_{(p)}$ indexes the image $I \in \mathbb{R}^{W \times H \times C}$ at pixel p . By jointly learning to reconstruct all five images and varying the masking patterns during training, we hypothesize that the encoder will learn to reason across the multiple views and extract 3D-aware features that are robust to occlusions and viewpoint changes. In order to inform future works, we study how various masking strategies and dataset combinations affect the final downstream performance (See Tab. 7.2).

Implementation details. We implement the pretraining using the PyTorch library and train it on eight NVIDIA V100 GPUs. We use the Objaverse dataset for pretraining [51], which contain a total of 800K+ 3D objects with realistic textures and materials. We sample 200K high-quality 3D models as the training set, and 1000 for validation purpose. We do not construct a large-scale validation set, since the validation is qualitative.

We use a patch size of 10x10 to tokenize images. For the encoder \mathcal{E} , we use a multiview transformer with 8 layers, 8 attention heads and a hidden dimension of 1024. For the decoder, we use a multiview transformer with 2 layers, with the same number of attention heads and hidden dimension. We train the model for 15 epochs using the AdamW optimizer with a learning rate of 0.0001 and a weight decay of 0.01. We use a batch size of 3 and a masking probability of 0.75.

7.3.3 Finetuning on Downstream Manipulation Tasks

To adapt the pretrained encoder \mathcal{E} for a specific manipulation task, we finetune it along with the action decoder \mathcal{D} on a dataset of manipulation demonstrations. Assume a demonstration consists of tuples of virtual images, language goals and actions. During training, we randomly sample a tuple and supervise the network to predict the action given the virtual images and the language goal.

Implementation details. For finetuning on manipulation demonstrations, we follow the standard practice [81, 207]. We use 8 NVIDIA V100 (32GB) for finetuning and a single V100

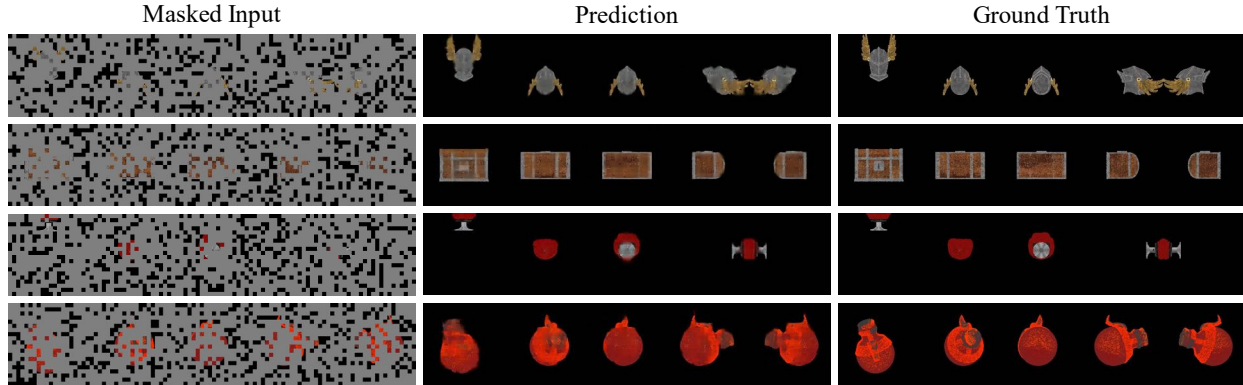


Figure 7.2: MAE Reconstruction results on Objaverse. We find the our pretrained multi-view transformer could generalize to unseen object instances and reconstruct multi-view images from their masked versions.

for evaluation. The learning rate is $1e-4$ with warmup. We use Lamb [304] as the optimizer and the batch size is 3.

7.4 Experiments

In this section, we evaluate the effectiveness of 3D-MVP for robotic manipulation tasks, and aim to answer the following questions: (1) Can masked autoencoding effectively learn useful representations from multi-view 3D data? (2) Does 3D-aware pretraining improve manipulation performance compared to training from scratch or 2D pretraining? (3) Does 3D-aware pretraining improve robustness to environmental variations encountered in manipulation tasks? (4) How do various design choices while pretraining affect the downstream manipulation performance?

To answer these questions, we evaluate 3D-MVP on two benchmarks: RLBench [113] for general manipulation performance and COLOSSEUM [207] for systematic evaluation of robustness to environmental perturbations.

7.4.1 Validating 3D Masked Autoencoding

We validate whether masked autoencoder works in our setup with multi-view images from 3D assets. Specifically, we check whether the pretrained multi-view transformer generalizes to unseen 3D assets from Objaverse [51]. We validate it qualitatively in Figure 7.2. We find that the pretrained 3D-MVP network achieves high-fidelity reconstructions despite 75% of the input points being masked, suggesting that 3D-MVP learns meaningful 3D representations through this pretext task.

7.4.2 Results on RL Bench

We then evaluate whether our proposed pretraining improves manipulation performance. The experiments are conducted on a simulated platform called RL Bench [113]. **Setup.** RL Bench [113] is a popular simulation benchmark for learning manipulation policies. Each task requires the robot to perform a specific action, such as picking up an object, opening a drawer, or stacking blocks. We follow the simulation setup of PerAct [242] and RVT [81] and use CoppelaSim [227] to simulate 18 RL Bench tasks. A Franka Panda robot with a parallel gripper is controlled to complete the tasks. The 18 RL Bench tasks are the same as PerAct and RVT. The visual observations are captured from four noiseless RGB-D cameras positioned at the front, left shoulder, right shoulder, and wrist with a resolution of 128×128 .

Baselines. We compare 3D-MVP with the following baselines on RL Bench: (1) **Image-BC** [115] is an image-to-action behavior cloning approach which takes the visual observation and predict the corresponding action. We compare with two variants which use CNN and ViT as the visual encoders, and call them Image-BC (CNN) and Image-BC (ViT), respectively; (2) **C2F-ARM-BC** [114] is another behavior cloning approach which converts RGB-D observations to multi-resolution voxels and predicts the next key-frame action. (3) **PerAct** [242]: a multi-task a Perceiver transformer for robotic manipulation. The inputs are point clouds with color features and PerAct uses a Perceiver IO network to compress them to a fixed dimension [111]. (4) **RVT** [81]: The same Robotic View Transformer architecture as 3D-MVP but trained from scratch on the downstream tasks. We do not compare with 2D pretraining methods since they do not work well on RL Bench [207].

Metrics. We report the task success rate for each individual tasks, and the average success rate.

Results. We show quantitative results on Table 7.1. For the average success rate, 3D-MVP outperforms existing state-of-the-art methods, PerAct [242] and RVT [81], by a large margin. It demonstrates the effectiveness of bringing visual pretraining for manipulation policy which has explicit 3D modeling. We find the improvement of 3D-MVP mainly comes from tasks which has medium difficulty, such as “insert peg”, “put in cupboard”, and “stack blocks”. If the task is too hard and PerAct/RVT is not able to solve any of them, the pretraining does not help. If the task is too easy and PerAct/RVT has already reached >90 success rate, the pretraining has limited space of improvement.

Models	Average Success	Close Jar	Drag Stick	Insert Peg	Meat off Grill	Open Drawer	Place Cups	Place Wine	Push Buttons
Image-BC (CNN) [115, 242]	1.3	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0
Image-BC (ViT) [115, 242]	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C2F-ARM-BC [114, 242]	20.1	24.0	24.0	4.0	20.0	20.0	0.0	8.0	72.0
PerAct [242]	49.4	55.2	89.6	5.6	70.4	88.0	2.4	44.8	92.8
RVT [81]	62.9	52.0	99.2	11.2	88.0	71.2	4.0	91.0	100
3D-MVP (Ours)	67.5	76.0	100	20.0	96.0	84	4.0	100	96.0

Models	Put in Cupboard	Put in Drawer	Put in Safe	Screw Bulb	Slide Block	Sort Shape	Stack Blocks	Stack Cups	Sweep to Dustpan	Turn Tap
Image-BC (CNN) [115, 242]	0.0	8.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0
Image-BC (ViT) [115, 242]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16.0
C2F-ARM-BC [114, 242]	0.0	4.0	12.0	8.0	16.0	8.0	0.0	0.0	0.0	68.0
PerAct [242]	28.0	51.2	84.0	17.6	74.0	16.8	26.4	2.4	52.0	88.0
RVT [81]	49.6	88.0	91.2	48.0	81.6	36.0	28.8	26.4	72.0	93.6
3D-MVP (Ours)	60.0	100.0	92.0	60	48	28	40	36	80	96

Table 7.1: Results on RL Bench [113]. We report the task completion success rate for 18 RL Bench tasks, as well as the average success rate. 3D-MVP reaches the state-of-the-art performance on the benchmark. The pretraining is mainly helpful for tasks with medium difficulty.

7.4.3 Results on COLOSSEUM

After validating our proposed pretraining is helpful for robotic manipulation, we further evaluate its generalization ability and robustness to environmental variations. Therefore, we evaluate 3D-MVP and the baselines on the COLOSSEUM axes of variation.

Benchmark. COLOSSEUM [207] is a benchmark for evaluating generalization for robotic manipulation. It contains 20 different tasks such as hockey, empty dishwasher. For each task, it generates 12 environmental perturbations, including changes in color, texture, size of objects and backgrounds, and lightnings, distractors and camera poses. The objects which can be changed include Manipulation object (M0), Receiver Object (R0) and the table. Therefore, it is well-suited for evaluating the generalization ability of manipulation approaches with pretraining.

Simulation setup. For simulation, we follow the original COLOSSEUM setup. We use CoppelaSim [227] to simulate all tasks. In training, we do not add any environmental perturbations and generate 100 demonstrations for each task. During test time, we generate 12 environmental perturbations for each task. For each environmental perturbation, we generate 25 demonstrations. For each demonstration, we repeat the generation 20 times if it fails. However, we still experienced a few rendering errors. For example, we find it’s hard to find the right perturbation parameters for the task of “empty dishwasher”. Therefore, we only report results on settings we are able to render. We report baselines on exactly the

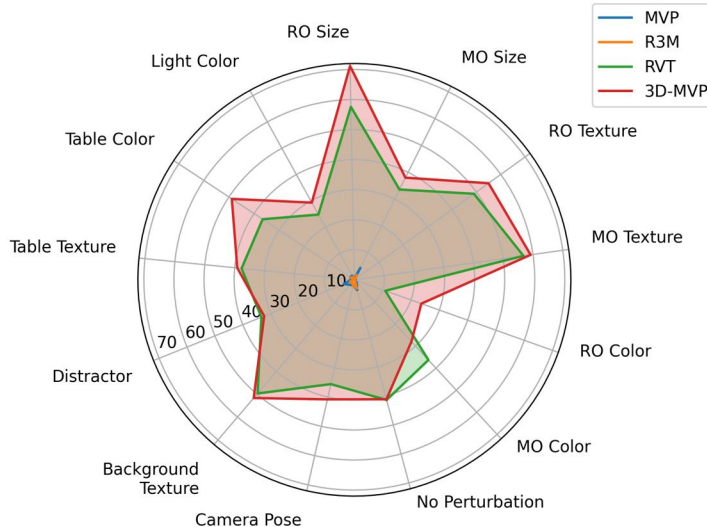


Figure 7.3: Results on COLOSSEUM [207]. We report the average task completion success rate for 12 environmental perturbations and no perturbation. Manipulation policies which do explicit 3D reasoning (RVT [81] works significantly better and 2D pretraining approaches (MVP [288] and R3M [197]). 3D-MVP is more robust than RVT on most perturbations. MO = manipulation object. RO = receiver object.

same setting, to make sure the comparison is fair.

Metrics. We also report the task completion success rate on COLOSSEUM. Instead of reporting the average success rate for each individual tasks, we report the average success rate for each perturbation of the environment, as it will highlight how each method is robust to different perturbations.

Baselines. We compare 3D-MVP with state-of-the-art baselines reported on COLOSSEUM, which includes RVT and two 2D pretraining approaches.

- MVP [288, 215]: A 2D pretraining approach using MAE reconstruction losses. It is pretrained on a collection of interaction datasets, such as Ego4D [83], EpicKitchen [46], and 100DOH [236]. The pretrained encoder is then finetuned and evaluated on COLOSSEUM.
- R3M [197] A 2D pretraining approach using a combination of reconstruction and contrastive losses. It is pretrained on Ego4D videos and languages [83]. The pretrained encoder is then finetuned and evaluated on COLOSSEUM.
- RVT [81]: Trained on COLOSSEUM tasks from scratch.

Results. We show results in Figure 7.3. First, our method outperforms existing 2D pretraining (MVP [288], and R3M) significantly. It indicates existing 2D pretrainig methods are not ready for complicated robotic manipulation. Compared with RVT which is trained from scratch, our method is more robust to most perturbations. It is especially robust to the

Network Architecture	Pretraining Datasets	Masking Strategy	Success Rate
3D-MVP	Objaverse (full) [51]	RGB	67.6
3D-MVP	Objaverse (small) [51]	RGB	65.3
3D-MVP	Objaverse (full) [51]	All	64.4
3D-MVP	3D-FRONT [74]	RGB	63.6
3D-MVP	RLBench [113]	RGB	67.5
3D-MVP	RLBench [113]	All	64.7
3D-MVP	None	None	62.9
RVT [81]	None	None	62.9

Table 7.2: Ablation studies on the RLBench benchmark. We analyze the contribution of our network architecture, pretraining datasets, and the masking strategy. For each variant, we report the average task completion success rate on RLBench [113].

change of texture and size of Receiver Object (RO), size of the manipulation object (MO), Light color and Table color. We believe it is because the pretraining stage enables our approach to see diverse 3D objects.

7.4.4 Ablation Studies

To analyze the impact of different design choices in 3D-MVP, we conduct ablation studies on the RLBench benchmark. Table 7.2 shows the average success rates of 3D-MVP with different network architecture, masking strategies and pretraining datasets. And we discuss results as follows.

Does the improvement come from the change of network architecture? In order to do the pretraining on RVT, we made some architecture changes as described in Sec. 7.3.2. To validate that the performance boost comes from the pretraining instead of the network architecture, we finetune our approach without pretraining and find the performance is 62.9, similar to the original RVT [81].

Should we pretrain on object or room-level data? The choice of pretraining datasets are typically critical for self-supervised learning. In our experiments, we mainly use Objaverse [51], which is a object-centric 3D datasets. Since we are mainly evaluated on tabletop manipulation, we also try room-level 3D datasets such as 3D-FRONT [74]. We conduct the experiments on RLBench, and observe pretraining on 3D-FRONT only boosts the performance mildly from 62.9 to 63.6. In comparison, pretraining on Objaverse boosts the performance to 67.6. This suggests that the diversity and scale of the pretraining data are important for learning generalizable representations.

Does more pretraining data help? We sampled a subset of Objaverse with 18K objects

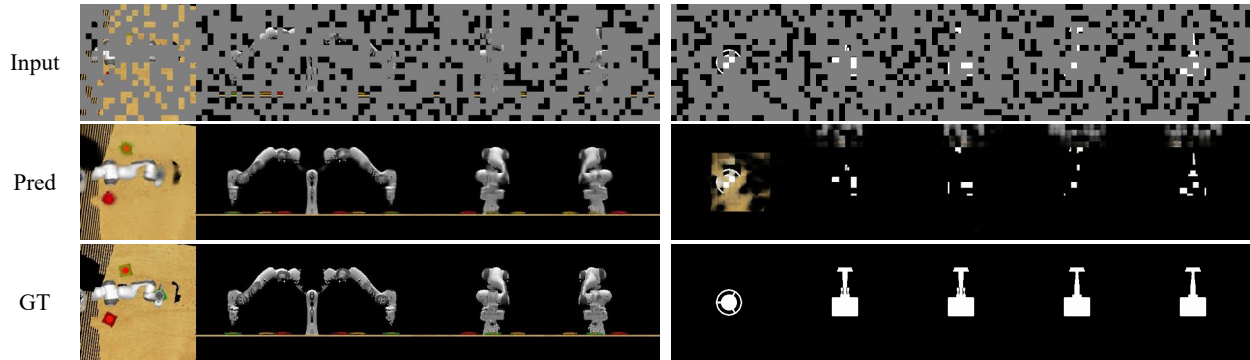


Figure 7.4: Pretraining MAE on only RL Bench scenes leads poor generalization performance. **(Left)**: MAE reconstruction results on unseen RL Bench renderings. **(Right)**: MAE reconstruction results on Objaverse renderings. While the reconstruction is reasonable on RL Bench unseen renderings, it overfits to RL Bench and does not learn a general representation.

and called it Objaverse (small). The full Objaverse dataset we pretrain on has 200K objects. When we pretrain the encoder with Objaverse (small), we get a 65.3 mean success rate, which is worse than using Objaverse (full). This suggests that a larger dataset of pretraining helps performance in the downstream task.

Can we pretrain on RL Bench? The proposed 3D pretraining is self-supervised and only requires the 3D point cloud. Therefore, an interesting question is: can we just pretrain on the manipulation dataset (e.g. RL Bench)? We extract the RL Bench point cloud and build a pretraining dataset. After the pretraining, we finetune the model on training demonstrations as usual. We find it can achieve an average success rate of 67.5 on RL Bench test set, which is comparable to Objaverse pretraining. However, the pretrained model suffers from generalization issues. As is shown in Figure 7.4, the encoder has overfitted to RL Bench, and does not work on other environments such as COLOSSEUM.

Masking strategy. For the masking strategy, we observe that masking RGB channels performs better than masking all channels. We hypothesize that the network finds it very challenging to reconstruct all the channels and is unable to learn better visual representations. We view our findings as similar to He et al. [92], who find that the benefits of pretraining diminish if the pretraining task becomes too difficult, like when the masking ratio becomes very high ($\geq 80\%$).

7.5 Conclusion

In this work, we introduced 3D-MVP, a novel approach for 3D multi-view pretraining using masked autoencoders to improve the performance and generalization of robotic manipulation policies. By pretraining the encoder of Robotic View Transformer (RVT) on the large-scale Objaverse 3D object dataset using masked autoencoding, we demonstrate that the learned 3D representations lead to improved sample efficiency and robustness when finetuned on downstream manipulation tasks. We evaluated our approach on two benchmarks: RLBench, for general manipulation performance and COLOSSEUM, for systematic evaluation of robustness to environmental perturbations. On RLBench, 3D-MVP outperforms state-of-the-art manipulation baselines, achieving higher success rates. On COLOSSEUM, which tests 12 axes of variations such as object color, size, texture, lighting and more, 3D-MVP maintains higher success rates compared to baselines as the magnitude of perturbations increases. These results suggest that scalable 3D-aware pretraining on diverse object datasets is a promising approach to developing general-purpose robotic manipulation systems.

Limitations and future work. While 3D-MVP achieves promising results on the RLBench and COLOSSEUM benchmarks, there are several limitations that we plan to address in future work. First, the current version of 3D-MVP uses a fixed set of camera viewpoints and does not explicitly reason about occlusions and spatial relationships between objects. In future work, we plan to explore more advanced 3D representations, such as neural radiance fields, that can handle arbitrary camera viewpoints and model the 3D structure of the scene more explicitly. Second, the current version of 3D-MVP assumes that the scene, robot, and objects follow quasi-static dynamics, and does not handle dynamic interactions between the robot and the environment. In future, we plan to explore techniques for learning action-conditional representations that can predict the effect of the robot’s actions on the 3D scene. Third, the current version of 3D-MVP requires a small amount of labeled data for each downstream task. In future, we plan to explore how to enable 3D-MVP to generate to novel manipulation tasks which have not been finetuned on.

Social impacts. The development of more generalized and robust robotic manipulation systems enabled by 3D-aware pretraining as proposed in our work has the potential for significant societal impact. On the positive side, such systems could automate many repetitive or dangerous manual labor tasks, improving worker safety and productivity. Assisting robots that can reliably manipulate objects in unstructured environments could also improve quality of life for the elderly and people with disabilities. However, the increased automation from these systems may also displace some jobs, disproportionately impacting workers with lower levels of education and technical skills. It will be important to enact policies to help retrain

and transition affected workers to new roles. Additionally, the datasets used for pretraining these models, like Objaverse, may encode biases that could be reflected in the downstream robotic system's behavior. Care must be taken to audit the data and model behavior to mitigate potential discriminatory impacts.

CHAPTER 8

Future Directions

While significant progress has been made in the field of embodied AI and its interaction with the 3D world, there remain grand challenges and opportunities for future research. This section outlines promising directions in the areas of multimodal LLM agents and dynamic digital twins.

8.1 Multimodal LLM Agent

Multimodal Large Language Models (LLMs) have demonstrated a deep understanding of interactions and show great potential for controlling embodied agents and robots. Two promising research directions in this area are:

Digital agent. By leveraging the ability of multimodal LLMs to understand interactions, we can build powerful embodied AI agents. A prime example is Voyager, the first LLM-powered embodied lifelong learning agent in Minecraft that continuously explores the world, acquires diverse skills, and makes novel discoveries without human intervention [269]. Besides digital agents in Minecraft, we can also build web agents. For example, Webvoyager can complete user instructions end-to-end by interacting with real-world websites [91]. Building upon their success, future research could explore the development of LLM-powered embodied agents in more complex and realistic environments. It potentially leads to AI systems that can intelligently assist humans in various domains, such as scheduling meetings, taking phone calls, and scheduling travel plans.

Robotics foundation model. The world knowledge embedded in multimodal LLMs has shown the potential to build robotics foundation models. Traditionally, each robot requires a separate model for each task, such as a manipulation policy to perform actions like opening a drawer. Despite progress in multitask manipulation policies [81, 242, 75], they are still limited to a single robot setup, like the Franka Panda. However, with multimodal LLMs, it may be possible to learn generalizable robotics skills that can be transferred to individual robot

setups. One potential solution is ask the multimodal LLM to predict the visual observation of each key step, and use a low-level execution policy to reach the key step. For example, VoxPoser extracts affordances and constraints from LLMs to compose 3D value maps, which are used by motion planners to zero-shot synthesize trajectories for everyday manipulation tasks [105]. These advancements in robotics foundation models can potentially bring a “ChatGPT moment” to robotics in the future, revolutionizing how we interact with and utilize robots in various domains.

8.2 Dynamic Digital Twins

Constructing dynamic digital twins remains open challenges with significant potential for future research. Two promising directions in this area are:

Dynamic digital twin from videos. Developing systems that can construct interactive digital replicas of real-world environments in real-time from video input is an exciting frontier. Early attempts, such as consistent video depth estimation [180], have shown the potential for rendering consistent 3D scenes from videos. More recently, Video2Game by Xia et al. demonstrates a novel approach to automatically convert real-world scene videos into realistic, interactive game environments [286]. Their system combines neural radiance fields (NeRF) for capturing scene geometry and appearance, mesh distillation for efficient rendering, and physics modeling for object interactions. Further research in this direction could enable a wide range of applications, from gaming and simulation to robot interaction in dynamic environments.

Dynamic digital twin generation. Significant progress has been made in 3D generative AI in recent years. For instance, Text2Room generates textured 3D meshes from text prompts using 2D text-to-image models [98]. SceneScape generates a long-term video depicting a walkthrough of the scene, according a given camera trajectory and text prompt [73]. However, the generated scenes remain static. Combining interactive 3D scene understanding with 3D generative AI could enable the creation of interactive, AI-generated 3D environments. This has potential applications in automatically generating interactive game worlds or constructing large-scale datasets for AI training. Challenges include ensuring consistency and coherence of the generated scenes under user interaction, and efficiently updating the 3D representations in real-time.

CHAPTER 9

Conclusion

In this dissertation, we have explored various approaches to enable machines to perceive, understand, and interact with the 3D world. Our research has spanned from passive 3D perception to active object manipulation, and we have developed techniques that advance the state-of-the-art in machine perception and interaction.

- Associative3D introduced a novel approach for jointly estimating 3D reconstructions, object associations and camera poses from two views, demonstrating strong performance on synthetic data and generalization to real indoor scenes.
- ViewSeg enabled 3D semantic segmentation from arbitrary viewpoints using only a few annotated images per scene, without requiring explicit 3D supervision. This makes 3D semantic understanding more practical for real-world applications.
- A novel system was developed to produce 3D planar representations of object articulation from ordinary videos. This enables dynamic 3D understanding of how objects move and function.
- A transformer model was introduced to predict crucial interaction-related properties for objects in an image, including movability, rigidity, articulation, affordances and potential actions. This allows an agent to reason about how to manipulate novel objects.
- AffordanceLLM leveraged the knowledge in large vision-language models to ground affordances for arbitrary objects and actions, significantly outperforming prior methods in generalizing to new object categories and novel interactions.
- 3D-MVP utilized multi-view masked autoencoding to learn rich 3D representations that improve the performance and robustness of robotic manipulation policies. This enables active manipulation of objects guided by visual understanding.

One of the key lessons we have learned throughout our research is the importance of leveraging large-scale datasets and pretraining mechanisms to improve performance and generalization. By using diverse video data and vision-language models, we have been able to train models that can predict 3D object interactions and affordances from a single image, and even manipulate objects in the 3D world.

Another important lesson is the need for multimodal learning and dynamic digital twins. As we move towards more capable and adaptable AI agents, it is crucial to develop systems that can learn from multiple modalities and adapt to changing environments. This requires building digital agents and robotics foundation models that can learn from diverse data sources and generalize to new situations.

Finally, our research has highlighted the potential of self-supervised 3D representation learning to bridge the gap between visual understanding and physical interaction. By learning rich 3D representations from diverse object datasets, we have been able to improve the performance and robustness of robotic manipulation policies.

Overall, my PhD research has taught us the value of exploring novel approaches to machine perception and interaction, and the importance of leveraging large-scale datasets and pretraining mechanisms to improve performance and generalization. I hope that our work will contribute to the development of more capable and adaptable AI agents that can navigate and operate in both the physical world and its digital twin.

APPENDIX A

Supplementary: Volumetric Reconstruction from Sparse Views

A.1 Implementation

Detection proposals. We use more advanced object proposals compared to prior works [138, 265], which used edge boxes [324]. We found that edge boxes were often the limiting factor. Instead, we train a class-agnostic Faster-RCNN [222] to generate proposals, treating all objects as the foreground.

Object Branch. For each object, our object branch will predict a 300-dimensional vector, which represents its 3D properties. Linear layers are used to predict its shape embedding, translation, rotation, scale and object embedding. For the object embeddings, we use three linear layers. The size of outputs is 256, 128, 64, respectively. These linear layers predict a 64-dimensional embedding finally.

We train the object branch in two stages. In the first stage, we follow the training of 3D-RelNet [138] with ground truth bounding boxes. The loss of affinity matrix is ignored in this stage. In the second stage, we freeze all layers except the linear layers to predict the object embeddings. We only apply the affinity loss in this stage. For all two stages, we use Adam with learning rate $\varepsilon = 10^{-4}$ to optimize the model, with momentum 0.9. The batch size is 24. Although 3D-RelNet is finetuned on detection proposals, we only use the intermediate model trained with ground truth bounding box because (1) 3D-RelNet is finetuned on edgebox proposals and our Faster-RCNN proposals are good enough; (2) ground truth affinity is only available with ground truth bounding box.

Camera Branch. The object and camera branches are trained independently. The translation is represented as 3D vectors, and the rotation is represented as quaternions. We run k-means clustering on the training set to produce 60 and 30 bins for translation and rotation. For rotation, we use spherical k-means to ensure the centroids are unit vectors.

The input image pairs are resized to 224x224. They are passed through a siamese network with ResNet-50 pre-trained on ImageNet [95] as the backbone. The outputs from each instance of the siamese network are concatenated, and passed through a linear layer, producing a 128-dimensional vector. The vector is then passed through a translation branch and a rotation branch. Each branch is a linear layer which outputs 60 and 30 dimensional vectors for translation and rotation bins.

Our loss function is the cross entropy loss. The loss for the translation prediction and the rotation prediction are weighted equally. We use stochastic gradient descent with learning rate $\varepsilon = 10^{-3}$ and momentum 0.9. The batch size is 32. We also augment the data by reversing the order of image pairs.

Tuning the stitching stage. The search space contains top-3 rotation, top-10 translation and top-128 object correspondence hypotheses. The threshold of affinity is 0.5. λ_P , λ_S , and λ_U are tuned as hyperparameters on the validation set to preclude the trivial solution. We use $\lambda_S = 5$, $\lambda_U = 1$. For λ_P , we use 5 for rotation and 1 for translation.

A.2 Visualization of the Object Embedding Space

We use t-SNE [183] to visualize the object embedding space to check what the affinity matrix learns visually. We show our results in Fig. A.1.

Without using semantic labels as supervision, objects in the same category are closer to each other. We also notice that table and desk have similar embeddings. The object embeddings can distinguish 3D models partially, but not as well as semantic labels.

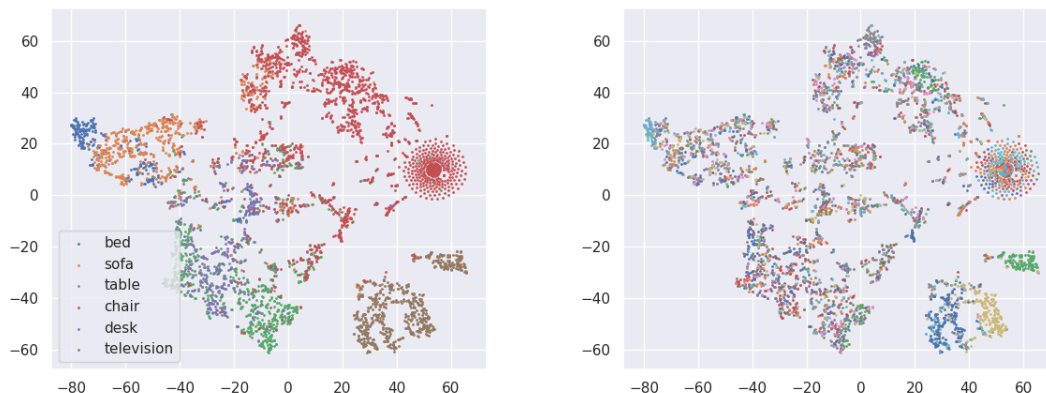


Figure A.1: t-SNE visualization of the object embedding space. **Left:** We assign the same color to embeddings with the same semantic labels. **Right:** We assign the same color to embeddings with the same 3D models.

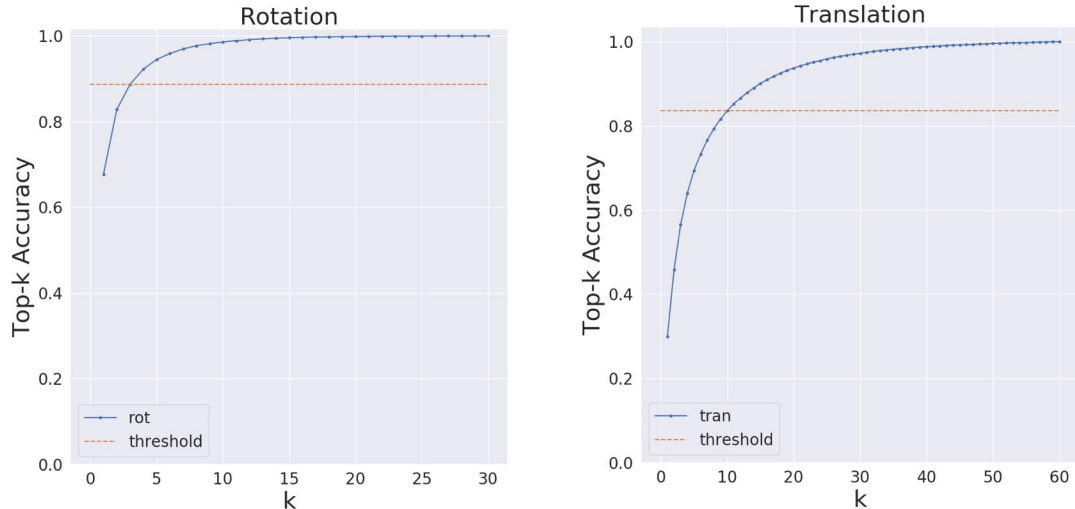


Figure A.2: Top-K accuracy of the camera branch on the validation set. The orange line shows the Top-K accuracy of K we choose for the stitching stage.

A.3 Proposals of Camera Pose Transformation

In the stitching stage, we select the top 3 most likely bins for rotation and top 10 most likely bins for translation. We demonstrate our motivation for choosing the number. On the validation set, we evaluate the top-K classification accuracy. We show the top-K accuracy for translation and rotation in Fig. A.2. We notice that the top-1 accuracy is not high. However, the top 3 most likely bins for rotation and top 10 most likely bins for translation can ensure a high accuracy in a relatively small searching space. Therefore, we select these bins in the stitching stage. During test time, the top-3 rotation accuracy is 88.7% and top-10 translation accuracy is 83.6%.

A.4 Comparion with Single-view Baselines

In full scene evaluation, we are also interested in whether the multi-view setting helps, since we only add another sparse view. We address the question by comparing with single-view baselines. We take a prediction from 3D-RelNet [138] on one view of the pair, selected randomly. On the whole test set, the AP is 13.7, which significantly underperforms all baselines. It shows our proposed approach has significant improvement built upon single-view baselines, and multi-view helps reconstruct the scene.

A.5 Merging Corresponding Objects

When our approach finds corresponding objects in two views, we average the translation and scale, but pick up the shape and rotation at random. Here we study alternative options. We use top 50% examples in the test set ranked by the performance of single-view predictions, so that the difference is more obvious.

First, we empirically show the rotation cannot be averaged, since there are typically multiple rotation modes. In Table A.1, we compare the performance between averaging rotation and pick up one at random. Averaging rotation dramatically hurts the performance.

Table A.1: Comparison between averaging the rotation and picking up one at random.

	All	Shape	Trans	Rot	Scale
random rot	38.8	27.3	39.6	33.2	35.1
average rot	31.4	27.3	40.2	29.2	35.1
improvement	-7.4	+0.0	+0.6	-4.0	+0.0

Moreover, a reasonable way to average the shape is to average the vector representation of the object [77]. In Table A.2, we compare their performance and they are almost the same. Therefore, we choose the simplest approach, picking up one shape at random.

Table A.2: Comparison between averaging the shape and picking up one at random.

	All	Shape	Trans	Rot	Scale
average shape	38.8	27.3	39.6	33.2	35.1
random shape	38.8	27.2	39.6	33.2	35.1
improvement	+0.0	-0.1	+0.0	+0.0	+0.0

A.6 Additional Qualitative Results

We show additional qualitative examples in Fig. A.3, A.4, and A.5 which follows the same format as Fig. 3, 4 and 5 of the main paper.

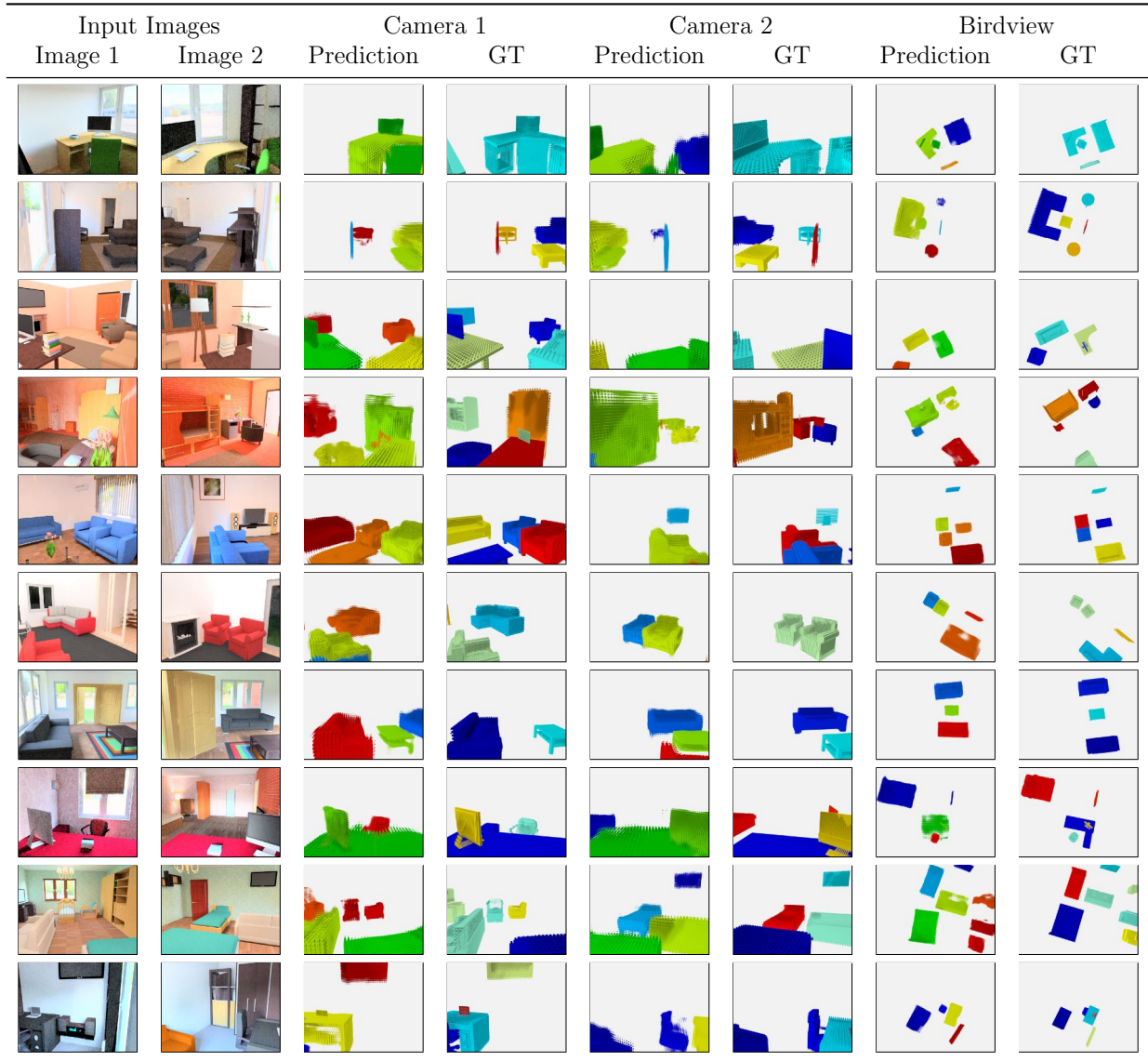


Figure A.3: Additional Qualitative results on the SUNCG test set. The final 3D predictions are shown in three different camera poses (1) the same camera as image 1; (2) the same camera as image 2; (3) a bird view to see all the objects in the whole scene. In the prediction, red/orange objects are from the left image, blue objects are from the right image, green/yellow objects are stitched from both the images.

Image 1	Image 2	Feedforward	NMS	Raw Affinity	Associative3D	GT

Figure A.4: Comparison between Associative3D and alternative approaches. All outputs are shown from bird's eye view.

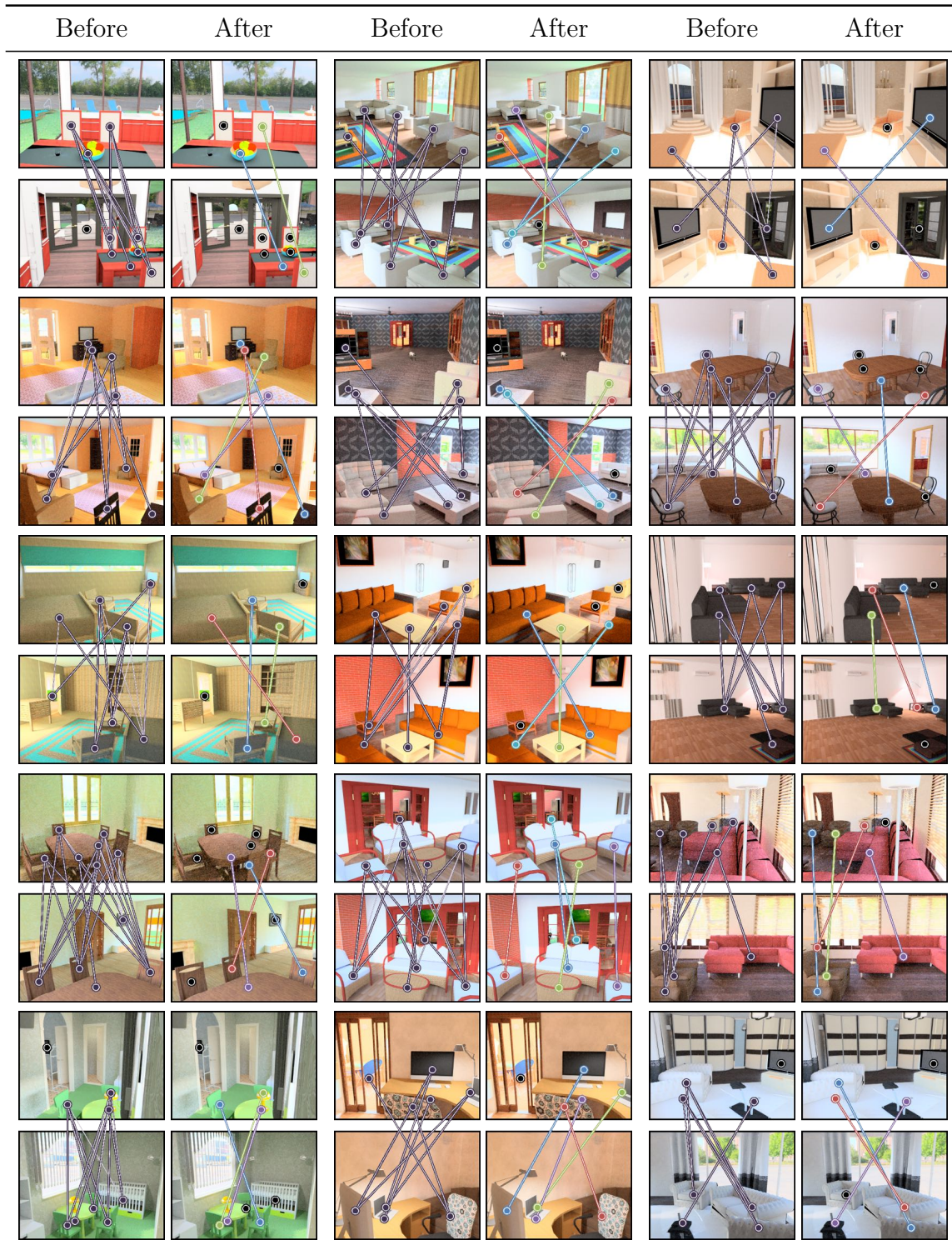


Figure A.5: Visualization of the stitching stage. The affinity matrix generates proposals of corresponding objects, and then the stitching stage removes outliers by inferring the most likely explanation of the scene. Before stitching, the thickness and darkness of the line represent the value of affinity score. The thicker / darker, the higher the value in the affinity matrix.

APPENDIX B

Supplementary: 3D Semantic Segmentation from Novel Viewpoints

B.1 Network Architecture

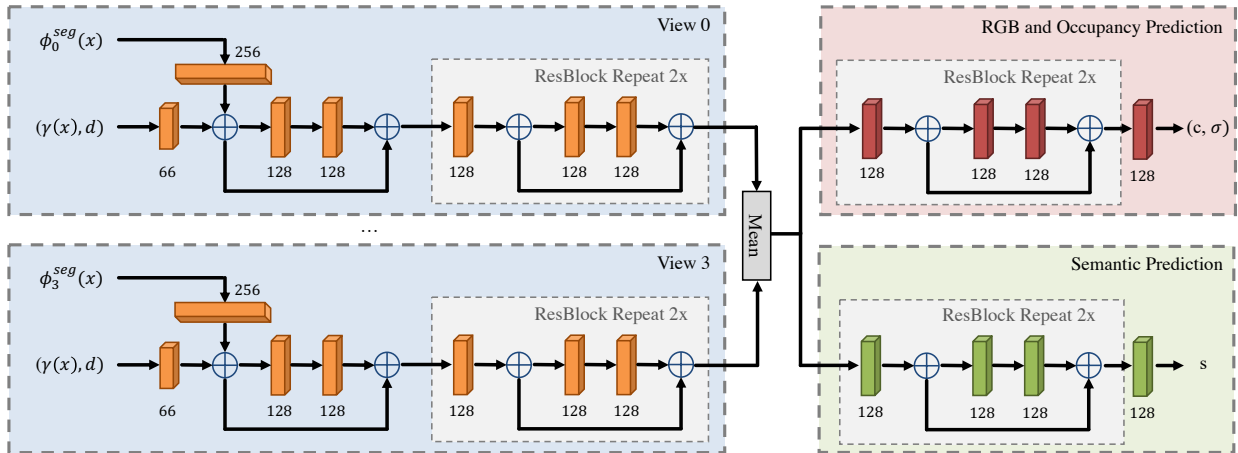


Figure B.1: Detailed architecture of our semantic 3D representation f . Each cuboid is a linear layer where the number around it represents the input dimension. The output dimension is always 128 except the last layer of RGB and semantic prediction. Before the mean aggregation, the network takes inputs from each source view but weights are shared.

The detailed architecture of f is illustrated in Fig. B.1. It takes as input a positional embedding of the 3D coordinates of \mathbf{x} , $\gamma(\mathbf{x})$, the viewing direction \mathbf{d} and the semantic embeddings $\{\phi_j^{\text{seg}}\}_{j=1}^N$. As output, f produces

$$(\mathbf{c}, \sigma, \mathbf{s}) = f(\gamma(\mathbf{x}), \mathbf{d}, \phi_0^{\text{seg}}(\mathbf{x}), \dots, \phi_{N-1}^{\text{seg}}(\mathbf{x})) \quad (\text{B.1})$$

where $\mathbf{c} \in \mathbb{R}^3$ is the RGB color, $\sigma \in \mathbb{R}$ is the occupancy, and $\mathbf{s} \in \mathbb{R}^{|\mathcal{C}|}$ is the distribution over semantic classes \mathcal{C} . Hypersim [225] provides annotations for 40 semantic classes. We discard

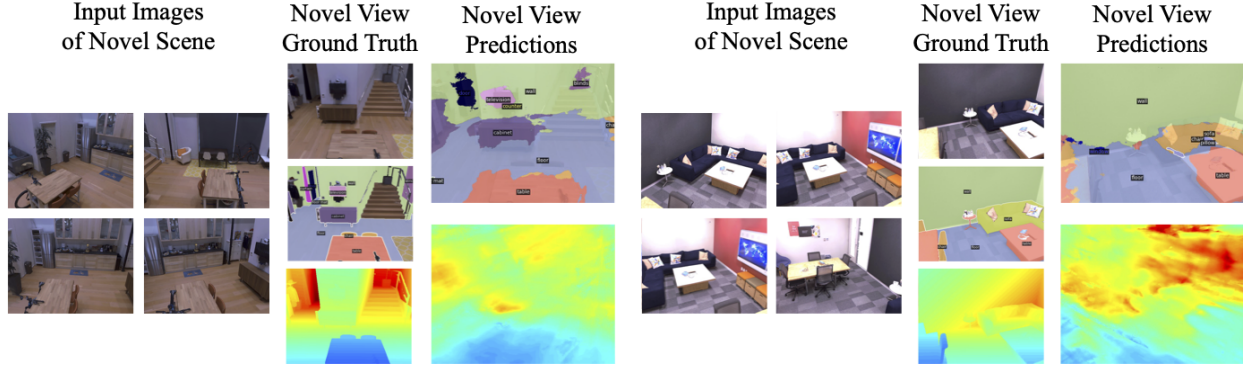


Figure B.2: Predictions on Replica with noisy cameras. For each example, we show the 4 input RGB views (left), the ground truth RGB, semantic and depth maps for the novel target view (middle) and ViewSeg’s predictions (right). Our model does not have access to the true observations from the target view at test time. Depth colormap (scaled per example): min max.

Model	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta < 1.25^2$
ViewSeg noft	8.57	35.2	61.3	40.3	57.9	18.3	1.10	0.253	0.240	0.268	0.579	0.840
ViewSeg	14.1	40.1	62.8	42.8	60.6	24.6	0.887	0.208	0.232	0.180	0.631	0.888

Table B.1: Performance for semantic segmentation (blue) and depth (green) on the Replica [253] test set with noisy cameras.

$\{otherstructure, otherfurniture, otherprop\}$ and thus $|\mathcal{C}| = 37$ for both Hypersim [225] and Replica [253].

We largely follow PixelNeRF [305] for the design of our network. We deviate from PixelNeRF and use 128 instead of 512 for the dimension of hidden layers (as in NeRF [188]) for a more compact network. The dimension of the linear layer which inputs $\phi_j^{seg}(\mathbf{x})$ is set to 256 to match the dimension of semantic features from DeepLabv3+ [30].

B.2 Training Details

Pretraining the Semantic Segmentation Module. We first pretrain our semantic segmentation backbone network DeepLabv3+ [30] on ADE20k [318], which has 20,210 images for training and 2,000 images for validation. We implement DeepLabv3+ in PyTorch with Detectron2 [284]. We train on the ADE20k training set for 160k iterations with a batch size of 16 across 8 Tesla V100 GPUs. The model is initialized using ImageNet weights [52]. We optimize with SGD and a learning rate of 1e-2. During training, we crop each input image to 512x512. We remove the final layer, which predicts class probabilities, and use the output from the penultimate layer as our semantic encoder. We do not freeze the model when training ViewSeg, allowing finetuning on Hypersim semantic categories.

Training Details on Hypersim. We implement ViewSeg in PyTorch3D [219] and Detectron2 [284]. We initialize the semantic segmentation module with ADE20k pretrained weights. We train on the training set for 13 epochs with a batch size of 32 across 32 Tesla V100 GPUs. The input and render resolution are set to 1024x768. In practice, we find the semantic encoder needs relatively high resolution to segment small objects. We optimize with Adam and a learning rate of 5e-4. We follow the PixelNeRF [305] strategy for ray sampling: We sample 64 points per ray in the coarse pass and 128 points per ray in the fine pass; we sample 1024 rays per image. We set $\lambda = 25$ in Eq. 6 of the main paper to balance the semantic and photometric loss, which is consistent with Semantic-NeRF [316].

Training Details on Replica. We finetune our model on the Replica training set [253]. Replica has 18 scenes. In practice, we find Replica does not have room-level annotations and our sampled source and target views can be at different rooms within the Replica apartments. Hence, we exclude them from our data. We split the rest 15 scenes into a train/val split of 12/3 scenes. We use the same hyperparameters as Hypersim to finetune our model on Replica.

B.3 Noisy Camera Experiment

In our experiments, we assume camera poses both during training and evaluation. We perform an additional ablation assuming *noisy* cameras for both during training and testing. During evaluation, source view cameras are noisy but not the target camera, as we wish to compare to the target view ground truth. We insert noise to the cameras by perturbing the rotation matrix with random angles sampled from $[-10^\circ, 10^\circ]$ in all three axis (X , Y & Z). This results in a significant camera noise and stretch tests our method under such conditions.

Table B.1 shows results on the noisy Replica test set. The 1st row shows the performance of our ViewSeg pre-trained on Hypersim and without any finetuning. The 2nd row shows the performance of our model finetuned on the noisy Replica training set. We observe that performance for both model variants is worse compared to having perfect cameras, as is expected. Performance improves after training with camera noise, suggesting that our ViewSeg is able to generalize better when trained with noise in cameras. Fig. B.2 shows qualitative results on the noisy Replica test set. The predicted RGB targets are significantly worse compared to the perfect camera scenario, which suggests that RGB prediction with imperfect cameras is very challenging. However, the semantic predictions are much better computer to their RGB counterparts. This shows that our approach is able to capture scene

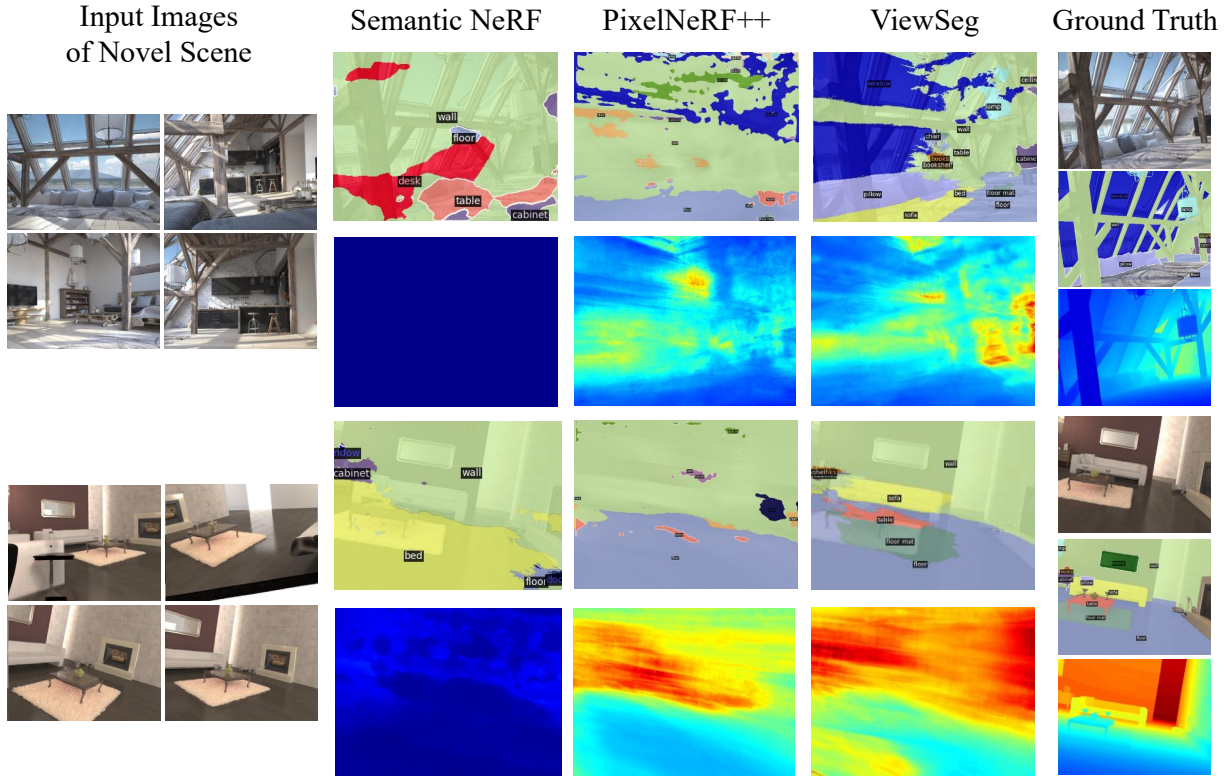


Figure B.3: More predictions on Hypersim. Depth colormap (scaled per example): min max.

priors and generalize to new scenes even under imperfect conditions.

B.4 Additional Results

Fig. B.3 shows more qualitative results on Hypersim. Fig. B.4 shows more qualitative results on Replica. We also provide video animations of our predictions in the supplementary folder.

B.5 Evaluation

We report the complete set of depth metrics for all the tables in the main submission. The comparison with PixelNeRF and CloudSeg is in Table B.2. The ablation of different loss terms is in Table B.3. The comparison of different backbones is in Table B.4. The input study is in Table B.5. Our experiments on the Replica dataset are in Table B.6.

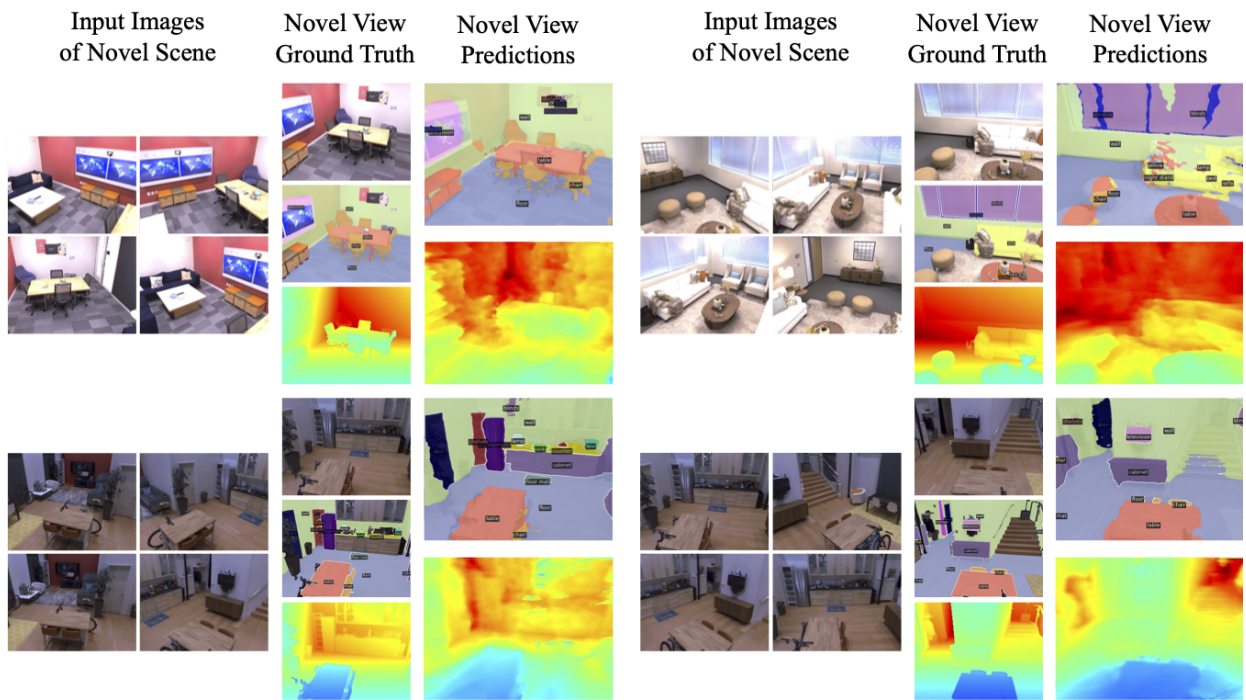


Figure B.4: More predictions on Replica. Depth colormap (scaled per example): min  max.

Model	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	L ₁ ^T (↓)	L ₁ ^S (↓)	Rel(↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$	
PlexNerf+S	1.58	14.9	47.9	17.9	35.9	3.63	2.80	2.69	2.90	0.746	0.856	0.653	0.300	0.276	0.319	0.531	0.500	0.557	0.689	0.663	0.712	0.712
CloudSeg	0.46	29.6	4.42	1.80	3.31	3.25	3.81	3.83	3.77	0.856	0.907	0.737	0.145	0.165	0.178	0.277	0.211	0.332	0.389	0.314	0.451	0.451
ViewSeg	17.1	33.2	58.9	44.8	62.2	23.9	2.29	2.18	2.38	0.646	0.721	0.584	0.409	0.393	0.421	0.656	0.633	0.676	0.794	0.772	0.812	0.812
Oracle	40.0	58.1	71.3	66.6	79.1	52.1	0.96	1.10	0.83	0.235	0.317	0.163	0.731	0.651	0.800	0.808	0.848	0.942	0.954	0.925	0.978	0.978

Table B.2: Extended version of Table 1 in the main paper.

ViewSeg loss	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	L ₁ ^T (↓)	L ₁ ^S (↓)	Rel(↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$	
w/o photometric loss	16.9	30.8	58.7	44.8	62.5	22.7	2.49	2.35	2.61	0.677	0.750	0.615	0.359	0.347	0.363	0.611	0.594	0.625	0.764	0.744	0.780	0.780
w/o semantic loss	-	-	-	-	-	-	2.58	2.52	2.63	0.787	0.919	0.678	0.345	0.317	0.369	0.587	0.548	0.621	0.740	0.704	0.770	0.770
w/o source view loss	14.3	28.2	57.9	28.2	61.1	19.3	2.37	2.27	2.45	0.683	0.764	0.615	0.397	0.378	0.413	0.649	0.623	0.670	0.785	0.761	0.806	0.806
w/o viewing dir	16.0	33.1	59.2	44.9	62.1	21.5	2.53	2.38	2.65	0.708	0.783	0.646	0.354	0.351	0.356	0.662	0.593	0.610	0.759	0.754	0.772	0.772
final	17.1	33.2	58.9	44.8	62.2	23.9	2.29	2.18	2.38	0.646	0.721	0.584	0.409	0.393	0.421	0.656	0.633	0.676	0.794	0.772	0.812	0.812

Table B.3: Extended version of Table 2 in the main paper.

ViewSeg backbone	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	L ₁ ^T (↓)	L ₁ ^S (↓)	Rel(↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$	
DLv3+ [6] + ADE20k [318]	17.1	33.2	58.9	44.8	62.2	23.9	2.29	2.18	2.38	0.645	0.721	0.584	0.409	0.393	0.421	0.656	0.633	0.676	0.794	0.772	0.812	0.812
DLv3+ [6] + IN [52]	16.3	33.2	59.2	45.2	62.5	22.0	2.28	2.17	2.36	0.614	0.682	0.559	0.415	0.400	0.427	0.663	0.640	0.682	0.799	0.776	0.818	0.818
ResNet34 [65] + IN [52]	7.45	21.7	55.9	37.1	56.1	11.2	2.67	2.51	2.81	0.712	0.815	0.626	0.320	0.304	0.333	0.562	0.541	0.580	0.720	0.702	0.736	0.736

Table B.4: Extended version of Table 3 in the main paper.

ViewSeg	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	L ₁ ^T (↓)	L ₁ ^S (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
w/ 4 views	17.1	33.2	58.9	44.8	62.2	23.9	2.29	2.18	2.38	0.646	0.721	0.584	0.408	0.393	0.421	0.656	0.633	0.676	0.794	0.772	0.812
w/ 3 views	15.5	31.3	58.7	43.9	61.5	20.8	2.39	2.25	2.49	0.652	0.730	0.587	0.387	0.376	0.395	0.634	0.617	0.648	0.777	0.759	0.793
w/ 2 views	13.6	27.4	57.7	41.9	60.2	18.2	2.57	2.49	2.64	0.765	0.878	0.672	0.363	0.339	0.383	0.605	0.574	0.633	0.751	0.721	0.776
w/ 1 view	11.6	24.9	56.5	39.7	57.9	15.8	2.62	2.52	2.70	0.734	0.828	0.657	0.332	0.322	0.339	0.562	0.541	0.580	0.710	0.686	0.730

Table B.5: Extended version of Table 4 in the main paper.

ViewSeg	mIoU	IoU ^T	IoU ^S	fwIoU	pACC	mACC	L ₁ (↓)	L ₁ ^T (↓)	L ₁ ^S (↓)	Rel (↓)	Rel ^T (↓)	Rel ^S (↓)	$\delta < 1.25$	$\delta^T < 1.25$	$\delta^S < 1.25$	$\delta < 1.25^2$	$\delta^T < 1.25^2$	$\delta^S < 1.25^2$	$\delta < 1.25^3$	$\delta^T < 1.25^3$	$\delta^S < 1.25^3$
ViewSeg notI	13.2	44.8	56.0	51.4	66.8	27.1	0.982	0.851	1.138	0.222	0.194	0.254	0.623	0.687	0.546	0.880	0.905	0.850	0.968	0.974	0.961
ViewSeg	30.2	56.2	62.8	62.3	75.6	48.4	0.550	0.510	0.597	0.130	0.130	0.130	0.851	0.857	0.844	0.961	0.953	0.972	0.986	0.980	0.992
Oracle	56.2	76.8	78.0	90.1	93.8	79.4	0.226	0.230	0.220	0.058	0.065	0.050	0.976	0.965	0.991	0.998	0.996	0.999	1.000	1.000	1.000

Table B.6: Extended version of Table 5 in the main paper.

APPENDIX C

Supplementary: Understanding 3D Articulation in Internet videos

C.1 Implementation Details

Table C.1: Overall architecture for our proposed network. The backbone, RPN and plane branches are identical to [120]. The RPN predicts a bounding box for each of A anchors in the input feature map. C is the number of categories (here = 2 for rotation and translation). We use class agnostic mask because the mask head is trained on ScanNet. TConv is a transpose convolution with stride 2. ReLU is used between all Linear, Conv and TConv operations. Depth branch uses Conv and Deconv layers to generate a depthmap with the same resolution as the input image.

Index	Inputs	Operation	Output shape
(1)	Inputs	Input Image	$H \times W \times 3$
(2)	(1)	Backbone: ResNet50-FPN	$h \times w \times 256$
(3)	(2)	RPN	$h \times w \times A \times 4$
(4)	(2),(3)	RoIAlign	$14 \times 14 \times 256$
(5)	(4)	Box: $2 \times$ downsample, Flatten, Linear($7 \times 7 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 5C$)	$C \times 5$
(6)	(4)	Mask: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), TConv($256 \rightarrow 256, 2 \times 2, 2$), Conv($256 \rightarrow 1 \times 1$)	$1 \times 28 \times 28$
(7)	(4)	Normal: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Linear($14 \times 14 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 3$)	1×3
(8)	(4)	Rotation Axis: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Linear($14 \times 14 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 3$)	$C \times 3$
(9)	(4)	Translation Axis: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Linear($14 \times 14 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 2$)	$C \times 2$
(10)	(2)	Depth	$H \times W \times 1$

Detector. Our network architecture is shown in Table C.1. We use Detectron2 [284] to implement our articulation detection and take the codebase from SparsePlanes [120]. Our articulation head predicts rotation and translation axis separately. Each of the branch takes the RoI feature from the backbone and uses four convolutional layers with 256 channels and two linear layers to regress the axes. The rotation branch predicts a three dimensional vector and the translation branch predicts a two dimensional vector. While they train the model on Matterport3D [26], we start from COCO pretraining and train the model on our Internet videos. The training is run on a single RTX 2080Ti.

Temporal Optimization. For tracking, we use 0.5 as our IoU threshold. For articulation model fitting, we use the ScanNet camera intrinsics as our assumed camera intrinsics, since the plane and depth heads of the model is trained on ScanNet [42]. We use PyTorch to implement the temporal optimization. For 3D transformations we use PyTorch3D [219] so that it is compatible. The optimization is parallel and runs on 8 GTX 1080Ti gpus of an internal cluster to save inference time and it can be run on a single gpu.

C.2 Data Collection Pipeline

Our semi-automatic data collection consists of an automatic pipeline to download and filter Internet videos to remove clear negatives, and a manual annotation to label articulated objects. We also discuss how we filter Charades dataset [243] since it involves slightly different steps.

C.2.1 Filtering Internet videos

Youtube queries. We start with 10 initial common articulated objects in our daily life: door, laptop, oven, refrigerator, washing machine, dishwasher, microwave oven, drawer, cabinet and box. Using the combination of words, we make a list of queries for each initial category, e.g. “best laundry tips”, following 100DOH [236]. To improve the number of videos we can find on the Internet and the diversity of the dataset, we also translate the queries into Chinese, Japanese, Hindi and Spanish. We search these queries on Youtube and download related Creative Commons videos.

Converting videos to shots. Within these videos, we find stationary continuous shots by fitting homographies [90] on ORB [229] features. In practice, we find ORB [229] are much faster and slightly more robust than SIFT [173] features, so it saves a lot of computing time.

Filtering videos based on interaction. A lot of stationary shots does not contain any people or objects of interest. We further filter out video shots based on a hand interaction detector trained on 100K+ frames of Internet data [236]. In practice, we find it works well and we believe it is because [236] is also trained on Internet data. For each video shot, we run the hand interaction detector on frames evenly sampled every 1 second. We remove video shots which do not have hand interactions at all.

Filtering categories of interests. We further filter object of interests by an object detector trained on COCO [158] and LVIS [87]. We use the pretrained model of Faster R-CNN (X101-FPN, 3x) from Detectron2 [284]. For categories that COCO does not have annotation (e.g.

washing machines), we use LVIS since it has much more categories. Especially, LVIS does not have annotations for doors, and we use doorknob instead.

C.2.2 Annotating Articulated Objects

Finally, we use to annotate articulated objects using crowdsourcing. We split video shots into 3s clips, where the fps is 30. Therefore, there are 90 frames per clip.

We use Hive¹ as our data annotation platform. We include the screenshot and estimated hourly pay for each step, since these steps are separate from each other. The hourly pay is a rough estimation since we only have limited worker statistics provided by Hive and we do not manage it ourselves.

Recognizing articulated clips. The first step is to judge if the video clips have objects which are being articulated. This is a binary classification question. We show 9 key frames of the video shot (sample every 10 frames) and ask workers to classify. The screenshot is shown in Figure C.1. We pay \$0.015 for each clip, each clip is annotated by about 2 workers (which is managed by Hive based on consensus and out of our control), and we estimate they can annotate 8 clips per minute. Therefore, the estimated hourly pay is $0.015 \cdot 8 \cdot 60/2 = \3.6 .

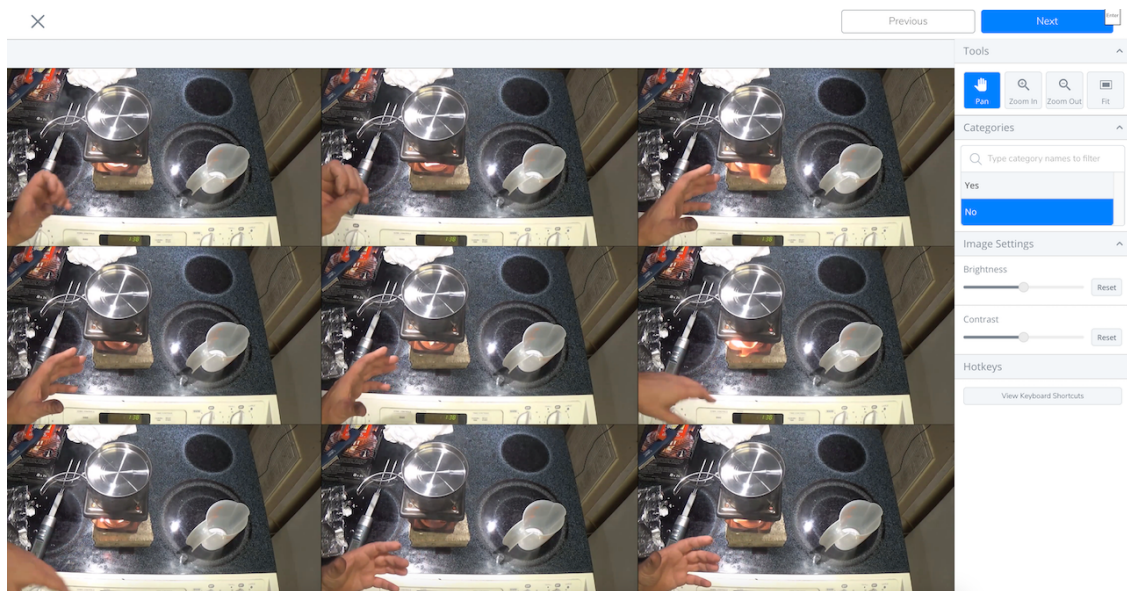


Figure C.1: The screenshot of recognize articulated clips.

Annotating bounding boxes of articulated objects. After labelling positive video clips, we annotate bounding boxes of objects which are being articulated. We also ask workers

¹<https://thehive.ai/>

to specify the object is being rotated or translated. We only annotate on 9 key frames of the video clip, since consecutive frames tend to be similar. The screenshot is shown in Figure C.2. We pay \$0.14 for each clip, each clip is annotated by about 2 workers, and we estimate they can annotate 1.5 clips per minute. Therefore, the estimated hourly pay is $0.14 \cdot 1.5 \cdot 60/2 = \6.3 .

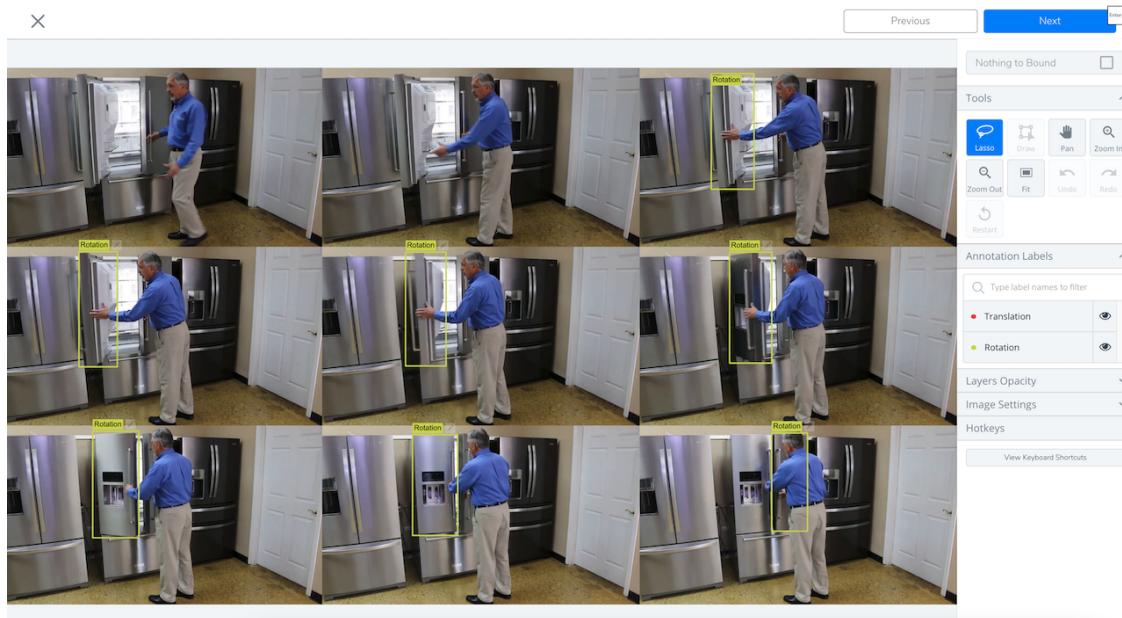


Figure C.2: The screenshot of annotating bounding boxes.

Annotating rotation axis. For objects which are annotated to be rotated, we annotate their rotation axes. We ask workers to draw a line to represent the 2D rotation axis. The screenshot is shown in Figure C.3. We pay \$0.04 for each clip, each clip is annotated by about 2 workers, and we estimate they can annotate 4 axes per minute. Therefore, the estimated hourly pay is $0.04 \cdot 4 \cdot 60/2 = \4.8 .

Annotating translation axis. For objects which are annotated to be translated, we annotate their translation direction. We also ask workers to draw a line to represent the 2D rotation direction. However, since translation is only related to the angle of the line and does not need the line offset, we draw a circle at the center of the bounding box and ask workers to start there. The screenshot is shown in Figure C.4. The estimated hourly pay is \$4.8, which is the same as annotating rotation axis since it is defined as the same task “line segment” on Hive.

Annotating surface normals. All bounding boxes and articulation axes are annotated in 2D. However, in this chapter, we are interested in 3D object articulation. Thus, for the



Figure C.3: The screenshot of annotating rotation axes.

test set, we also annotate the σ to annotate the surface normal of the plane following [34], so we can evaluate how well our model can learn 3D properties. Since the annotation of surface normals are not available on Hive and we only need surface normals on the test set for evaluation purpose, we do all surface normals annotations ourselves. The screenshot is shown in Figure C.5.

Finally, we postprocess the dataset to make sure the dataset does not have offensive content, cartoons, and any videos depicting children. Since the distribution is unbalanced and negative examples are much more than positive ones, we only sample a negative clip with hand interaction from the same video shot of positive clips.

C.2.3 Annotating Charades Dataset

We test generalization of our method on Charades without additional training. Data is prepared and annotated using the following process on the original Charades test set.

The Charades test set contains 1863 videos of people acting out designated scripts. Videos are typically short at around 30 seconds. The dataset contains script information and additional annotation, which can be used to filter videos that are highly likely to contain articulation. After initial filtering, we split filtered videos into clips of a pre-defined length. These clips are then annotated as to whether they contain articulation, for articulation bounding box location (if applicable), and articulation angle (if applicable). Annotation is performed using a similar setup to YouTube via Hive. Because Charades is small, we use all

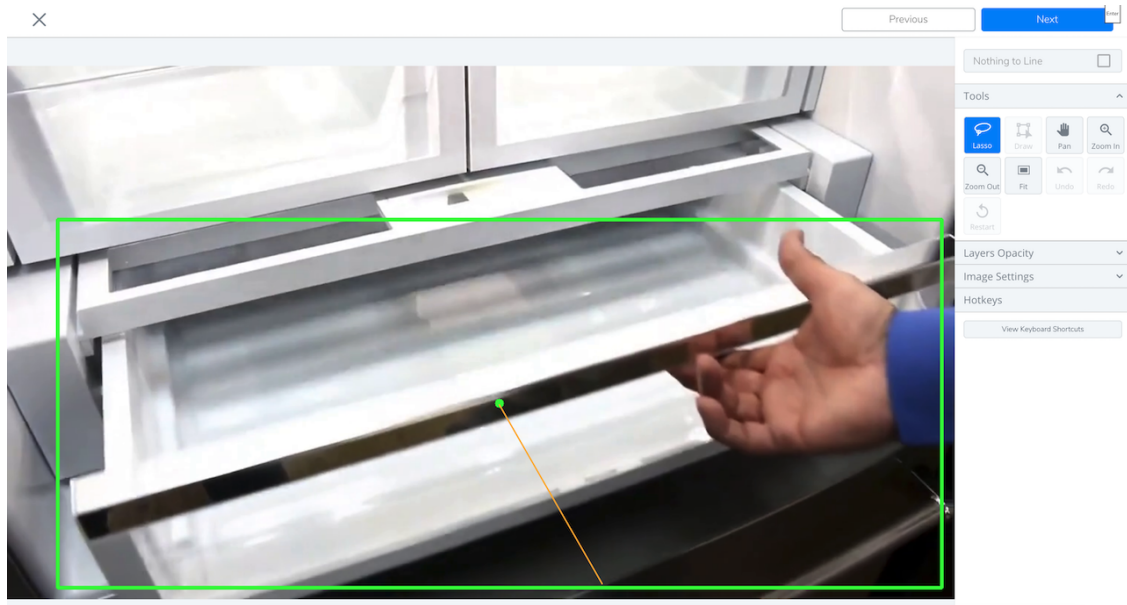


Figure C.4: The screenshot of annotating translation axes.

clips which Hive workers have labeled as containing articulation.

Filtering criteria. Charades contains videos with acting information, which we use to perform filtering. Each video has corresponding categorical actions that can be used to find dense articulation instances. These categorical actions fall into 157 categories such as "Holding some clothes" or "Opening a bag", and are annotated on a one-tenth of a second basis. For example, one given video may have two corresponding actions "Holding some clothes" and "Opening a bag", which correspond to seconds 1.2 - 11.7 and 12.3 - 18.3, correspondingly. We select video clips corresponding to eight categorical actions for our dataset: "Opening a door", "Closing a door", "Opening a laptop", "Closing a laptop", "Opening a closet/cabinet", "Closing a closet/cabinet", "Opening a refrigerator", and "Closing a refrigerator".

Gathering filtered clips. To find corresponding video clips, we first calculate the middle of each action – e.g. for the 1.2 - 11.7 interval this would be 6.45. Next, we select a 7.5 seconds both greater than and less than the middle of the action. This is subject to beginning and ending of video, and we remove overlapping clips. In our example, "Holding some clothes" has a middle of 6.45 seconds, so the clip would begin at 0 seconds and end at 13.95 seconds. The second clip has a middle of 15.3, and cannot overlap with the first, so would start at 13.95 seconds and end at 22.8. This totals 649 filtered clips of ≤ 15 seconds.

Splitting video clips. Given a set of video clips which correspond to the selected categorical actions, we next split clips into 3 second clips to be used for a standardized articulation framework. Any clip less than 3 seconds is truncated. This results in 2232 3-second mini-

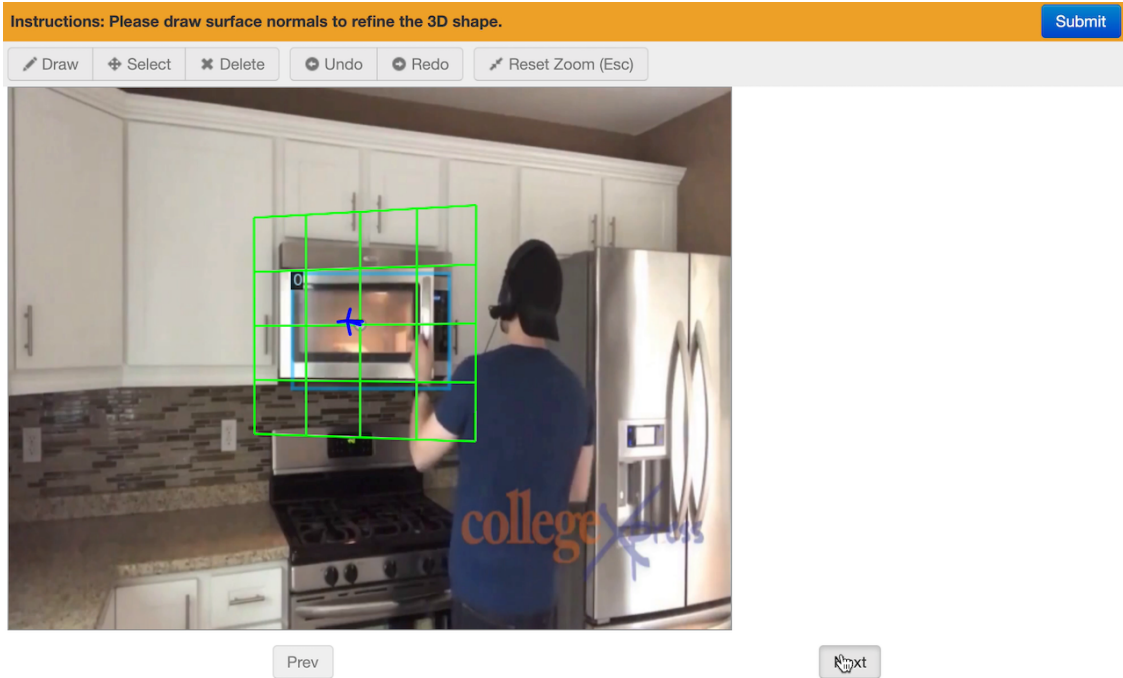


Figure C.5: The screenshot of annotating surface normals.

clips.

C.3 Rendering Sapien data

In our experiments, we test whether the model trained on synthetic data transfer to Internet videos using Sapien renderings [287]. Here we provide additional details about rendering and example images. Examples of our renderings are shown in Figure C.6.

To generate these results, we first randomly sample 3D objects with articulation. We filtered 1053 objects of 18 categories with movable planes from PartNet-Mobility Dataset [287]. 18 categories are: “Box”, “Dishwasher”, “Display”, “Door”, “FoldingChair”, “Laptop”, “Microwave”, “Oven”, “Phone”, “Refrigerator”, “Safe”, “StorageFurniture”, “Suitcase”, “Table”, “Toilet”, “TrashCan”, “WashingMachine”, and “Window”. They have significant overlapping with our queries to generate Internet videos, since these objects are common objects which can be articulated. We control its rotation or translation and render ground truth depth, surface normal, mask, motion type and 3D rotation axis. The outputs are object articulation videos without backgrounds.

To mimic real 3D scenes, we blend random backgrounds. Otherwise the detection problem becomes trivial. We use ScanNet [42] images with synthetic humans [267] used to train our approach, to ensure the fair comparison.



Figure C.6: Random examples from Sapien renderings.

APPENDIX D

Supplementary: Understanding 3D Object Interaction from a Single Image

D.1 Implementation

Transformer Decoder. The transformer decoder D takes the memory m from encoder and a set of queries, including N point queries k_p and one depth query k_d . It predicts a set of point pooled features h_1, \dots, h_N and depth pooled features h_d , *i.e.*

$$h_1, h_2, \dots, h_N, h_d = D(m; k_p^{(1)}, k_p^{(2)}, \dots, k_p^{(N)}, k_d) \quad (\text{D.1})$$

We set $N = 15$, as all images have lower than 15 query points. For images without 15 query points, we pad the input to 15 and do not train on these padding examples. The depth query k_d is a learnable embedding, similar to object queries in DETR [21]. All queries are feed into the decoder in parallel, as they are independent of each other.

Prediction heads. DETR [21] uses a linear layer to predict the object classes and a three-layer MLP to regress the bounding boxes, based on h . Motivated by DETR, we use a linear layer for the prediction of movable, rigidity, articulation class and action. We use a three-layer MLP to predict the bounding boxes and rotation axes, as they require localization. We add a gaussian bump [145] for affordance ground truth, where the radius is 5.

Balance of loss functions. Since we use multiple loss functions for each prediction and each loss has a different range, they need to be balanced. We treat the weights of losses as hyperparameters and tune them accordingly. The weights of movable, rigidity, articulation class, and action losses are 0.5. The weights of mask losses (both focal loss [156] and DICE [190]) are 2.0. The weights of box L1 loss is 5.0 and generalized IoU loss is 2.0. The weights of axis angle loss is 1.0 and axis offset loss is 10.0. The weights of affordance loss is 100.0. The weights of depth losses are 1.0. For both focal losses of segmentation masks and

affordance map, we use $\gamma = 2$. For the focal loss of segmentation mask, we use $\alpha = 0.25$ to balance positive and negative examples. In affordance we use the standard $\alpha = 0.95$ since there are much more negatives than positives.

Training details. The image encoder, prompt encoder and the mask decoder are pretrained on Segment-Anything [131]. To save gpu memory, we use SAM-ViT-b as the image encoder, which is the lightest pretrained model. The other heads (e.g. affordance) are trained from scratch. We use an AdamW optimizer [172] of the learning rate 10^{-4} and train the model for 200 epochs. The input and output resolution is 768×1024 . The batch size is 2. We train the model on four NVIDIA A40 gpu, with distributed data parallel.

Rendering 3D Interaction. Given all these predictions, we are able to predict the potential 3D object interaction of articulated objects from a single image. For articulated objects with a rotation axis, we first backproject the predicted 2D axis to 3D, based on the predicted depth [211]. We then rotate the object point cloud along the 3D axis and project it back to 2D. We fit a homography between the rotated object points and the original one, using RANSAC [69]. Finally, we warp the homography on the original object mask. There is a similar procedure for articulated objects with a translation axis. Instead, we estimate an average surface normal of the object, and use it as the direction of translation axis [161, 159, 211]. Moreover, the interaction of deformable objects is high dependent of its material, which is difficult to predict from pure visual cues [293]. On the other hand, freeform objects can be moved without any constraints. Therefore, in this chapter, we only render 3D interaction for articulated objects. We use pytorch3D [219] and opencv to implement the projection and homography fitting. Final results are shown in the animation video.

D.2 Data Collection

In this section, we introduce steps of the data annotation. We show the statistics of our dataset in Figure D.1. We also show additional annotations in Figure D.2.

Selecting query points. We first ask workers to select approximately five query points for each image. The query point should be on an interactive object. Some query point should be on large objects, while others should be on small objects. We annotate more query points of fixtures later, as fixtures do not need additional annotations.

Bounding boxes. According to the query point, we ask workers to draw a bounding box. The bounding box should only cover the movable part of an object. For example, if the query point is on the door of a refrigerator, the bounding box should only cover the door, instead of the whole refrigerator. It is because we are asking “what can I do here”.

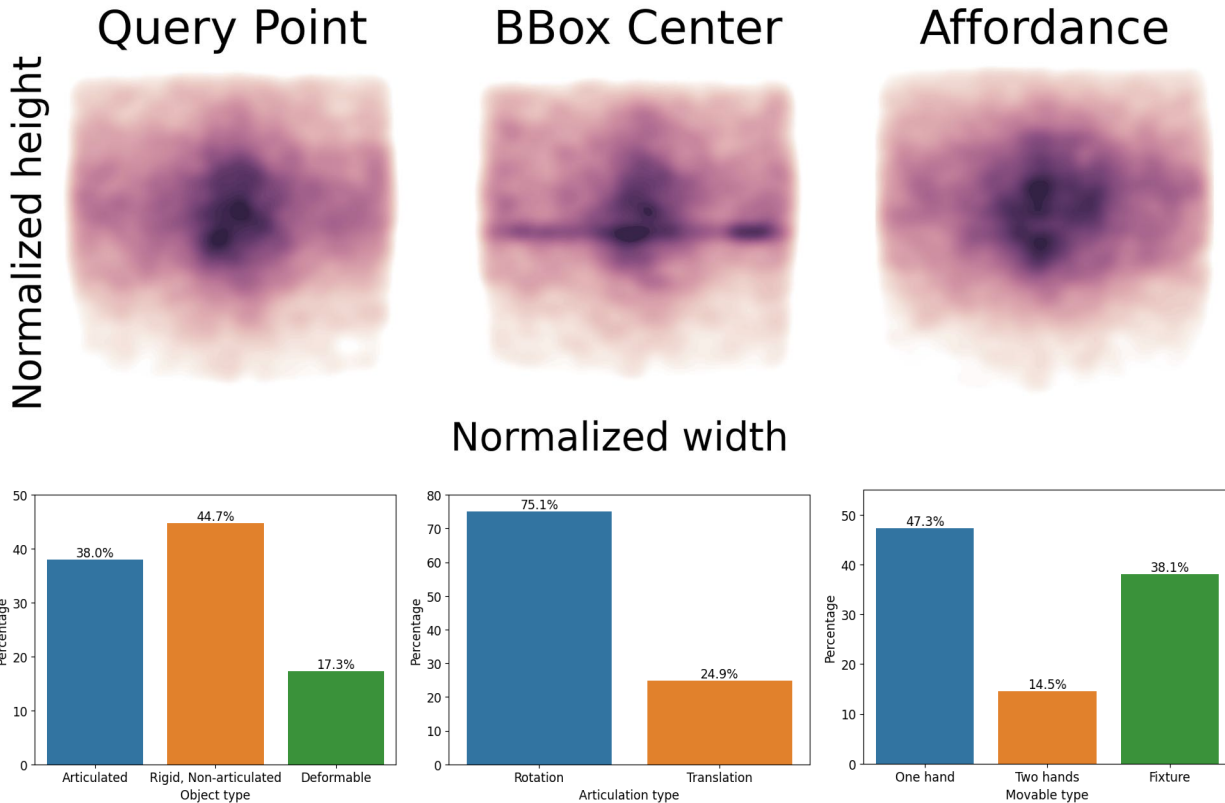


Figure D.1: Statistics of our 3DOI dataset. **(Row 1)** We show the distribution of query points, box centers, and affordance in normalized image coordinates, similar to LVIS [87] and Omni3D [13]. **(Row 2)** We show the distribution of object types, articulation types and movable types.

Properties of the object. We then annotate properties of the object. It is a series of multiple choice questions: (1) can the object be moved by one hand, or two hands? (2) is the object rigid or not? (3) if it is rigid, is it articulated or freeform? (4) if it is articulated, is the motion rotation or translation? (5) if we want to interact with the articulated object, should I push or pull?

Rotation Axes. For objects which can be rotated, we ask workers to draw a 2D line to represent the rotation axis.

Segmentation Masks. For all objects, we further ask workers to draw the segmentation mask of the movable part.

Fixtures. Finally, we collect another 10K images and randomly sample 5 query points for each image. We ask workers to annotate whether they are fixtures or not. We mix the dataset with these annotations.

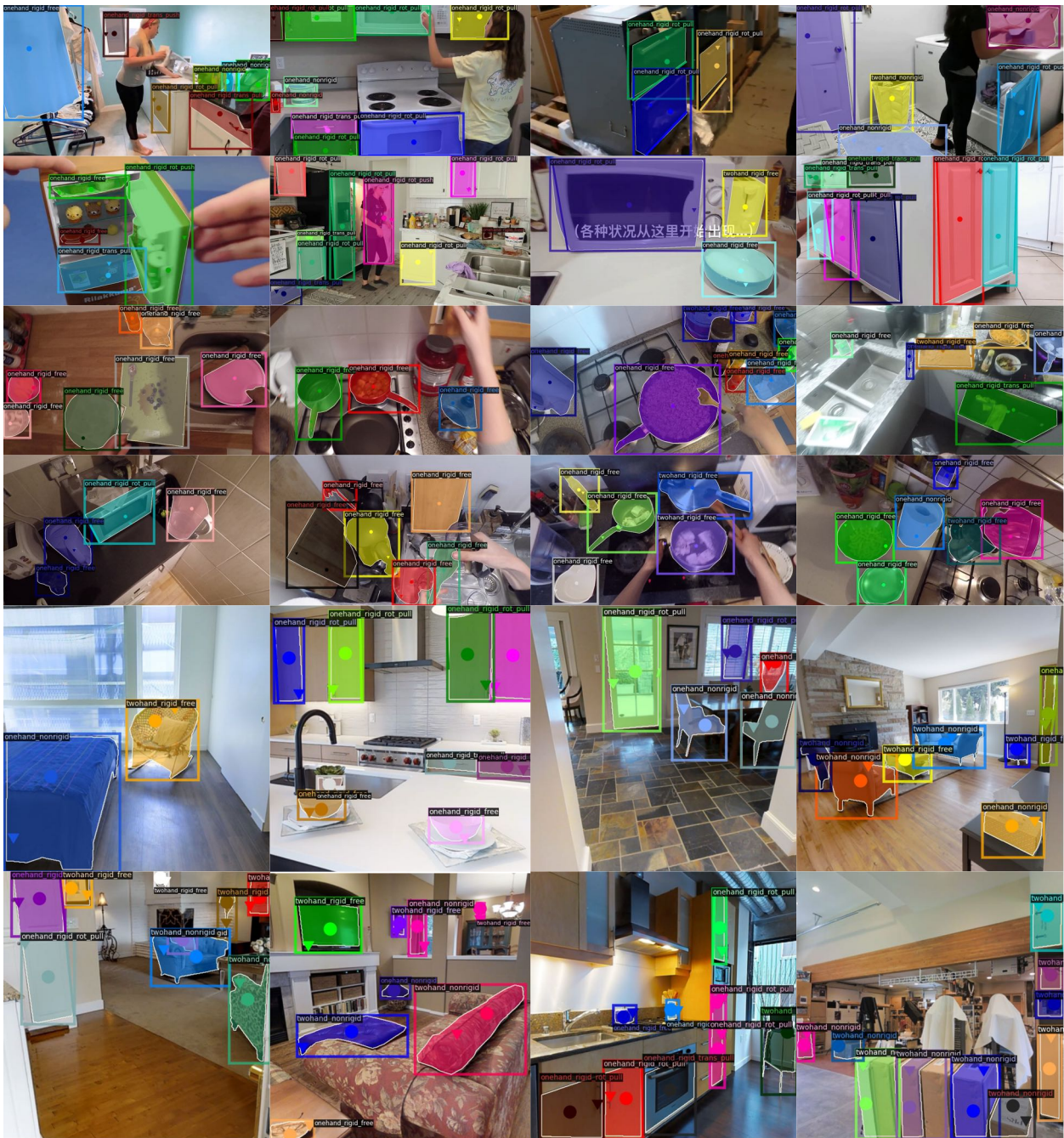


Figure D.2: Example annotations of our 3DOI dataset. Row 1-2 come from Internet videos [211]. Row 3-4 come from egocentric videos [47]. Row 5-6 come from renderings of 3D dataset [62]. The dot is the query point, and ▼ is the affordance.

APPENDIX E

Supplementary: Grounding Affordance from Vision Language Models

E.1 Metrics

In the section, we explain the metrics (KLD, SIM, and NSS) to evaluate our model.

- **Kullback-Leibler Divergence (KLD)** measures distribution difference between the predicted affordance map (M) and the ground truth (M'), which is

$$\text{KLD}(M, M') = \sum_i M'_i \log \left(\epsilon + \frac{M'_i}{\epsilon + M_i} \right), \quad (\text{E.1})$$

- **Similiary (SIM)** is also called histogram intersection, which measures the intersection between the predicted affordance map (M) and the ground truth (M'). The final range is from 0 to 1. It is given by

$$\text{SIM}(M, M') = \sum_i \min(M_i, M'_i), \quad (\text{E.2})$$

where $\sum_i M_i = \sum_i M'_i = 1$.

- **Normalized Scanpath Saliency (NSS)** measures the correspondence between the prediction map (M) and the ground truth (M'). It is given by

$$\text{NSS}(M, M') = \frac{1}{N} \sum_i \hat{M} \times M'_i, \quad (\text{E.3})$$

where $N = \sum_i M'_i$, $\hat{M} = \frac{M - \mu(M)}{\sigma(M)}$. $\mu(M)$ and $\sigma(M)$ are the mean and standard deviation, respectively.

E.2 Details of the Data Splits

In the easy split, we follow the object split of the original AGD20K *Unseen* setting [177]. The easy split has 33 object classes for training and 14 for testing. We have 1135/540 images for train and test with dense annotations for the fully supervised setting, or 13,323/540 images for the weakly supervised setting.

- Train classes: scissors, badminton racket, surfboard, frisbee, hot dog, tennis racket, hammer, microwave, oven, punching bag, carrot, snowboard, book, suitcase, skateboard, wine glass, keyboard, javelin, motorcycle, discus, bench, toothbrush, bottle, cell phone, chair, orange, rugby ball, couch, baseball, fork, bowl, apple, baseball bat.
- Test classes: camera, bed, bicycle, golf clubs, soccer ball, cup, laptop, banana, skis, knife, axe, broccoli, basketball, refrigerator.

In the hard split, we randomly put around 50% AGD20K object classes into the training set and the remaining classes into the test set to simulate in-the-wild generalization. The hard split has 28 object classes for training and 22 for testing. We have 868/807 images for train and test with dense annotations for the fully supervised setting, and 11,889/807 images for the weakly supervised setting.

- Training objects include carrot, cup, bowl, discus, book, camera, golf clubs, bottle, broccoli, binoculars, drum, baseball, apple, frisbee, cell phone, baseball bat, couch, hammer, bicycle, bench, fork, badminton racket, banana, hot dog, axe, bed, chair, basketball.
- Test objects include soccer ball, laptop, punching bag, oven, suitcase, javelin, wine glass, motorcycle, scissors, snowboard, keyboard, rugby ball, orange, surfboard, knife, skateboard, pen, microwave, skis, tennis racket, refrigerator, toothbrush.

BIBLIOGRAPHY

- [1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. 2009.
- [2] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.
- [3] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pages 456–473. Springer, 2022.
- [4] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.
- [5] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multima: Multi-modal multi-task masked autoencoders. In *ECCV*, 2022.
- [6] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. 2022.
- [7] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *CVPR*, 2023.
- [8] Sid Yingze Bao, Mohit Bagra, Yu-Wei Chao, and Silvio Savarese. Semantic structure from motion with points, regions, and objects. In *CVPR*, 2012.
- [9] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In *CVPR*, 2013.
- [10] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.
- [11] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.
- [12] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *ICRA*, 2017.
- [13] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3d: A large benchmark and model for 3d object detection in the wild. In *CVPR*, 2023.

- [14] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- [15] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [16] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.
- [17] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [18] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, 41(3):740–757, 2018.
- [19] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018.
- [20] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. In *CVPR*, 2022.
- [21] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- [22] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.
- [23] 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.
- [24] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012.
- [25] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *TPAMI*, 2011.
- [26] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.
- [27] Yu-Wei Chao, Zhan Wang, Yugeng He, Jiakuan Wang, and Jia Deng. Hico: A benchmark for recognizing human-object interactions in images. In *ICCV*, 2015.

- [28] Joya Chen, Difei Gao, Kevin Qinghong Lin, and Mike Zheng Shou. Affordance grounding from demonstration video to target image. In *CVPR*, 2023.
- [29] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [30] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.
- [31] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [32] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016.
- [33] Weifeng Chen, Shengyi Qian, and Jia Deng. Learning single-image depth from videos using quality assessment networks. In *CVPR*, 2019.
- [34] Weifeng Chen, Shengyi Qian, David Fan, Noriyuki Kojima, Max Hamilton, and Jia Deng. Oasis: A large-scale dataset for single image 3d in the wild. In *CVPR*, 2020.
- [35] Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. Holistic++ scene understanding: Single-view 3D holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [36] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021.
- [37] Cheng Chi and Dmitry Berenson. Occlusion-robust deformable object tracking without physics simulation. In *IROS*, 2019.
- [38] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *NeurIPS*, pages 2414–2422, 2016.
- [39] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016.
- [40] Cristina Garcia Cifuentes, Jan Issac, Manuel Wüthrich, Stefan Schaal, and Jeanette Bohg. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters*, 2(2):577–584, 2016.
- [41] David Crandall, Andrew Owens, Noah Snavely, and Daniel Huttenlocher. SfM with MRFs: Discrete-continuous optimization for large-scale structure from motion. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.
- [42] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

- [43] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015.
- [44] Yinpei Dai, Run Peng, Sikai Li, and Joyce Chai. Think, act, and ask: Open-world interactive personalized robot navigation. *arXiv preprint arXiv:2310.07968*, 2023.
- [45] Amaury Dame, Victor A. Prisacariu, Carl Y. Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *CVPR*, 2013.
- [46] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.
- [47] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55, 2022.
- [48] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, 2023.
- [49] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *ICML*, 2006.
- [50] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.
- [51] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.
- [52] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [53] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021.
- [54] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *CVPR*, 2021.
- [55] Karthik Desingh, Shiyang Lu, Anthony Pipari, and Odest Chadwicke Jenkins. Factored pose estimation of articulated objects using efficient nonparametric belief propagation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7221–7227. IEEE, 2019.
- [56] Tien Do, Khiem Vuong, and Hyun Soo Park. Egocentric scene understanding via multimodal spatial rectifier. In *CVPR*, 2022.
- [57] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold,

- Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [58] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [59] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [60] Yilun Du, Zhijian Liu, Hector Basevi, Ales Leonardis, Bill Freeman, Josh Tenenbaum, and Jiajun Wu. Learning to exploit stability for 3D scene parsing. In *Advances in Neural Information Processing Systems*, pages 1726–1736, 2018.
- [61] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deep-Pruner: Learning efficient stereo matching via differentiable patchmatch. In *ICCV*, 2019.
- [62] Ainaz Eftekhari, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, 2021.
- [63] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.
- [64] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.
- [65] Sovann En, Alexis Lechervy, and Frédéric Jurie. RPNet: an end-to-end network for relative camera pose estimation. In *ECCV*, 2018.
- [66] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [67] Rui Fan, Hengli Wang, Bohuan Xue, Huaiyang Huang, Yuan Wang, Ming Liu, and Ioannis Pitas. Three-filters-to-normal: An accurate and ultrafast surface normal estimator. *IEEE Robotics and Automation Letters*, 6(3):5405–5412, 2021.
- [68] Kuan Fang, Te-Lin Wu, Daniel Yang, Silvio Savarese, and Joseph J Lim. Demo2vec: Reasoning object affordances from online videos. In *CVPR*, 2018.
- [69] 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.
- [70] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, 2019.
- [71] David F. Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A. Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single-view geometry. In *ECCV*, 2012.

- [72] David F. Fouhey, Weicheng Kuo, Alexei A. Efros, and Jitendra Malik. From lifestyle VLOGs to everyday interactions. In *CVPR*, 2018.
- [73] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. 2023.
- [74] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.
- [75] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: Infinite resolution action detection transformer for robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.
- [76] J. Gibson. *The ecological approach to visual perception*. Boston: Houghton Mifflin, 1979.
- [77] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.
- [78] Georgia Gkioxari, Ross Girshick, Piotr Dollar, and Kaiming He. Detecting and recognizing human-object interactions. In *CVPR*, 2018.
- [79] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019.
- [80] Georgia Gkioxari, Nikhila Ravi, and Justin Johnson. Learning 3d object shape and layout without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1695–1704, 2022.
- [81] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.
- [82] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022.
- [83] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.
- [84] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent—a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.
- [85] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018.
- [86] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia Pinel, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. In *Conference on Robot Learning*, pages 175–187. PMLR, 2023.

- [87] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.
- [88] Denis Hadjivelichkov, Sicelukwanda Zwane, Lourdes Agapito, Marc Peter Deisenroth, and Dimitrios Kanoulas. One-shot transfer of affordance regions? affcorr! In *Conference on Robot Learning*, pages 550–560. PMLR, 2023.
- [89] Sanjay Haresh, Xiaohao Sun, Hanxiao Jiang, Angel X Chang, and Manolis Savva. Articulated 3d human-object interactions from rgb videos: An empirical analysis of approaches and challenges. *arXiv preprint arXiv:2209.05612*, 2022.
- [90] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [91] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.
- [92] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
- [93] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [94] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.
- [95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [96] Yang He, Wei-Chen Chiu, Margret Keuper, and Mario Fritz. Std2p: Rgbd semantic segmentation using spatio-temporal data-driven pooling. In *CVPR*, 2017.
- [97] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3d object categories from videos in the wild. In *CVPR*, 2021.
- [98] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *ICCV*, 2023.
- [99] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *arXiv preprint arXiv:2307.12981*, 2023.
- [100] Cheng-Chun Hsu, Zhenyu Jiang, and Yuke Zhu. Ditto in the house: Building articulation models of indoor scenes through interactive perception. In *ICRA*, 2023.
- [101] Haifeng Huang, Zehan Wang, Rongjie Huang, Luping Liu, Xize Cheng, Yang Zhao, Tao Jin, and Zhou Zhao. Chat-3d v2: Bridging 3d scene and large language models with object identifiers. *arXiv preprint arXiv:2312.08168*, 2023.

- [102] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.
- [103] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018.
- [104] Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. Holistic 3D scene parsing and reconstruction from a single rgb image. In *ECCV*, 2018.
- [105] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.
- [106] Zeng Huang, Tianye Li, Weikai Chen, Yajie Zhao, Jun Xing, Chloe LeGendre, Linjie Luo, Chongyang Ma, and Hao Li. Deep volumetric video from very sparse multi-view performance capture. In *ECCV*, 2018.
- [107] Christian Häne, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In *CVPR*, 2014.
- [108] Vladimir Iglovikov. Binary segmentation of people. https://github.com/ternaus/people_segmentation, 2020.
- [109] Vladimir Iglovikov and Alexey Shvets. Ternaunet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *arXiv preprint arXiv:1801.05746*, 2018.
- [110] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015.
- [111] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.
- [112] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. *arXiv preprint arXiv:2008.10518*, 2020.
- [113] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.
- [114] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *CVPR*, 2022.
- [115] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.
- [116] Juntao Jian, Xiuping Liu, Manyi Li, Ruizhen Hu, and Jian Liu. Affordpose: A large-scale dataset of hand-object interactions with affordance-driven hand pose. In *ICCV*, 2023.

- [117] Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel X Chang. Opd: Single-view 3d openable part detection. In *ECCV*, 2022.
- [118] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *CVPR*, 2022.
- [119] Ziyu Jiang, Buyu Liu, Samuel Schulter, Zhangyang Wang, and Manmohan Chandraker. Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations. In *CVPR*, 2020.
- [120] Linyi Jin, Shengyi Qian, Andrew Owens, and David F. Fouhey. Planar surface reconstruction from sparse views. In *ICCV*, 2021.
- [121] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human model fitting towards in-the-wild 3d human pose estimation. *arXiv preprint arXiv:2004.03686*, 2020.
- [122] Ken-ichi Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV*, 2001.
- [123] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.
- [124] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017.
- [125] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, 2018.
- [126] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.
- [127] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017.
- [128] 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 *6th Annual Conference on Robot Learning*, 2022.
- [129] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.
- [130] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019.
- [131] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *ICCV*, 2023.
- [132] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020.
- [133] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. Pare: Part attention regressor for 3d human body estimation. In *ICCV*, 2021.

- [134] Scott Konishi and Alan L Yuille. Statistical cues for domain specific image segmentation with performance analysis. In *CVPR*, 2000.
- [135] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.
- [136] Sumith Kulal, Tim Brooks, Alex Aiken, Jiajun Wu, Jimei Yang, Jingwan Lu, Alexei A Efros, and Krishna Kumar Singh. Putting people in their place: Affordance-aware human insertion into scenes. In *CVPR*, 2023.
- [137] Nilesh Kulkarni, Abhinav Gupta, David Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020.
- [138] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3D-RelNet: Joint object and relational network for 3D prediction. In *ICCV*, 2019.
- [139] Ashish Kumar, Saurabh Gupta, David F. Fouhey, Sergey Levine, and Jitendra Malik. Visual memory for robust path following. In *NIPS*, 2018.
- [140] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018.
- [141] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016.
- [142] Lubor Ladický, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.
- [143] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Ji-aya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.
- [144] Katrin Lasinger, René Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.
- [145] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018.
- [146] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022.
- [147] Gen Li, Varun Jampani, Deqing Sun, and Laura Sevilla-Lara. Locate: Localize and transfer object parts for weakly supervised affordance grounding. In *CVPR*, 2023.
- [148] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.
- [149] Lin Li, Salman Khan, and Nick Barnes. Silhouette-assisted 3D object instance reconstruction from a cluttered scene. In *ICCV Workshops*, 2019.
- [150] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *TOG*, 2018.
- [151] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *CVPR*, 2020.

- [152] 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.
- [153] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.
- [154] Zeju Li, Chao Zhang, Xiaoyan Wang, Ruilong Ren, Yifan Xu, Ruifei Ma, and Xiangde Liu. 3dmit: 3d multi-modal instruction tuning for scene understanding. *arXiv preprint arXiv:2401.03201*, 2024.
- [155] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
- [156] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
- [157] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.
- [158] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [159] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *CVPR*, 2019.
- [160] Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floornet: A unified framework for floorplan reconstruction from 3d scans. In *ECCV*, 2018.
- [161] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *CVPR*, 2018.
- [162] Hao Liu, Lisa Lee, Kimin Lee, and Pieter Abbeel. Instruction-following agents with jointly pre-trained vision-language models. 2022.
- [163] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.
- [164] Liu Liu, Han Xue, Wenqiang Xu, Haoyuan Fu, and Cewu Lu. Towards real-world category-level articulation pose estimation. *IEEE Transactions on Image Processing*, 2022.
- [165] 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 *ICCV*, 2023.
- [166] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *ICCV*, 2019.
- [167] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

- [168] Yizhou Liu, Fusheng Zha, Lining Sun, Jingxuan Li, Mantian Li, and Xin Wang. Learning articulated constraints from a one-shot demonstration for robot manipulation planning. *IEEE Access*, 7:172584–172596, 2019.
- [169] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.
- [170] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [171] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *ECCV*, 2014.
- [172] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [173] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [174] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *ICCV*, 2017.
- [175] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *CVPR*, 2022.
- [176] Hongchen Luo, Wei Zhai, Jing Zhang, Yang Cao, and Dacheng Tao. Grounded affordance from exocentric view. *arXiv preprint arXiv:2208.13196*, 2022.
- [177] Hongchen Luo, Wei Zhai, Jing Zhang, Yang Cao, and Dacheng Tao. Learning affordance grounding from exocentric images. In *CVPR*, 2022.
- [178] Hongchen Luo, Wei Zhai, Jing Zhang, Yang Cao, and Dacheng Tao. Leverage interactive affinity for affordance learning. In *CVPR*, 2023.
- [179] Tiange Luo, Honglak Lee, and Justin Johnson. Neural shape compiler: A unified framework for transforming between text, point cloud, and program. 2022.
- [180] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (TOG)*, 39(4):71–1, 2020.
- [181] Chuofan Ma, Yi Jiang, Xin Wen, Zehuan Yuan, and Xiaojuan Qi. Codet: Co-occurrence guided region-word alignment for open-vocabulary object detection. In *Advances in Neural Information Processing Systems*, 2023.
- [182] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.
- [183] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [184] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36, 2024.

- [185] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *ICRA*, 2017.
- [186] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 675–687. Springer, 2017.
- [187] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, 2015.
- [188] 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.
- [189] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022.
- [190] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.
- [191] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weisenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *ECCV*, 2022.
- [192] D. Mishkin, M. Perdoch, and J. Matas. Mods: Fast and robust method for two-view matching. *CVIU*, 1(141):81–93, 2015.
- [193] Kaichun Mo, Leonidas Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *ICCV*, 2021.
- [194] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *ICCV*, 2021.
- [195] Lorenzo Mur-Labadia, Jose J Guerrero, and Ruben Martinez-Cantin. Multi-label affordance mapping from egocentric vision. In *ICCV*, 2023.
- [196] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019.
- [197] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.
- [198] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020.
- [199] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: a retargetable forward and inverse renderer. *TOG*, 2019.

- [200] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.
- [201] OpenAI. Gpt-4 technical report, 2023.
- [202] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. In *ICLR*, 2023.
- [203] Claudia Pérez-D’Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *ICRA*, 2017.
- [204] Sudeep Pillai, Matthew R Walter, and Seth Teller. Learning articulated motions from visual demonstration. In *RSS*, 2014.
- [205] Andrew Price, Linyi Jin, and Dmitry Berenson. Inferring occluded geometry improves performance when retrieving an object from dense clutter. *International Symposium on Robotics Research (ISRR)*, 2019.
- [206] Philip Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *ICCV*, 1998.
- [207] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.
- [208] Shengyi Qian, Weifeng Chen, Min Bai, Xiong Zhou, Zhuowen Tu, and Li Erran Li. Understanding 3d object interaction from a single image. In *CVPRW*, 2024.
- [209] Shengyi Qian and David F Fouhey. Understanding 3d object interaction from a single image. In *ICCV*, 2023.
- [210] Shengyi Qian, Linyi Jin, and David F. Fouhey. Associative3d: Volumetric reconstruction from sparse views. In *ECCV*, 2020.
- [211] Shengyi Qian, Linyi Jin, Chris Rockwell, Siyi Chen, and David F Fouhey. Understanding 3d object articulation in internet videos. In *CVPR*, 2022.
- [212] Shengyi Qian, Alexander Kirillov, Nikhila Ravi, Devendra Singh Chaplot, Justin Johnson, David F Fouhey, and Georgia Gkioxari. Recognizing scenes from novel viewpoints. *arXiv preprint arXiv:2112.01520*, 2021.
- [213] Shengyi Qian, Kaichun Mo, Valts Blukis, David F Fouhey, Dieter Fox, and Ankit Goyal. 3d-mvp: 3d multiview pretraining for robotic manipulation. *arXiv preprint arXiv:2406.18158*, 2024.
- [214] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [215] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

- [216] Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.
- [217] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.
- [218] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.
- [219] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.
- [220] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.
- [221] 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 *International Conference on Computer Vision*, 2021.
- [222] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [223] Hamid Rezaatofghi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019.
- [224] Stephan R Richter and Stefan Roth. Matryoshka networks: Predicting 3D geometry via nested shape layers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1936–1944, 2018.
- [225] Mike Roberts and Nathan Paczan. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. *arXiv preprint arXiv:2011.02523*, 2020.
- [226] Chris Rockwell and David F Fouhey. Full-body awareness from partial observations. In *ECCV*, 2020.
- [227] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 1321–1326. IEEE, 2013.
- [228] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.
- [229] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.

- [230] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013.
- [231] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019.
- [232] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.
- [233] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. In *IEEE Robotics and Automation Letters*, 2017.
- [234] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.
- [235] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.
- [236] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020.
- [237] Dandan Shan, Richard E.L. Higgins, and David F. Fouhey. COHESIV: Contrastive object and hand embedding segmentation in video. In *NeurIPS*, 2021.
- [238] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *CVPR*, 2018.
- [239] Cheng Shi and Sibe Yang. Edadet: Open-vocabulary object detection using early dense alignment. In *ICCV*, October 2023.
- [240] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.
- [241] Daeyun Shin, Zhile Ren, Erik B Sudderth, and Charless C Fowlkes. 3d scene reconstruction with multi-layer depth and epipolar transformers. In *ICCV*, 2019.
- [242] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.
- [243] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.
- [244] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.
- [245] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.
- [246] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

- [247] Anthony Simeonov, Ankit Goyal, Lucas Manuelli, Lin Yen-Chen, Alina Sarmiento, Alberto Rodriguez, Pulkit Agrawal, and Dieter Fox. Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement. *arXiv preprint arXiv:2307.04751*, 2023.
- [248] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019.
- [249] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.
- [250] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.
- [251] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019.
- [252] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *ICCV*, 2017.
- [253] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.
- [254] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011.
- [255] Zhiqiang Sui, Haonan Chang, Ning Xu, and Odest Chadwicke Jenkins. Geofusion: Geometric consistency informed scene estimation in dense clutter. *arXiv:2003.12610*, 2020.
- [256] Xiaohao Sun, Hanxiao Jiang, Manolis Savva, and Angel Xuan Chang. Opdmulti: Openable part detection for multiple objects. *arXiv preprint arXiv:2303.14087*, 2023.
- [257] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018.
- [258] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021.
- [259] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.
- [260] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, 2019.

- [261] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.
- [262] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, 9(2):137–154, 1992.
- [263] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):878–892, 2008.
- [264] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [265] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2D image of a 3D scene. In *CVPR*, 2018.
- [266] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018.
- [267] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017.
- [268] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021.
- [269] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- [270] John YA Wang and Edward H Adelson. Representing moving images with layers. *IEEE transactions on image processing*, 3(5):625–638, 1994.
- [271] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *ICML*, 2022.
- [272] Qianqian Wang, Xiaowei Zhou, and Kostas Daniilidis. Multi-image semantic matching by mining consistent features. In *CVPR*, 2018.
- [273] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions ~ transformations. In *CVPR*, 2016.
- [274] Xiaolong Wang, David F. Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015.
- [275] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinqing Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *CVPR*, 2019.

- [276] Yixuan Wang, Dale McConachie, and Dmitry Berenson. Tracking partially-occluded deformable objects while enforcing geometric constraints. In *ICRA*, 2021.
- [277] Yian Wang, Ruihai Wu, Kaichun Mo, Jiaqi Ke, Qingnan Fan, Leonidas J Guibas, and Hao Dong. Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions. In *ECCV*, 2022.
- [278] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based object detection. *arXiv preprint arXiv:2109.07107*, 3(6), 2021.
- [279] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-supervised neural articulated shape and appearance models. In *CVPR*, 2022.
- [280] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020.
- [281] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3D shape reconstruction via 2.5D sketches. In *Advances in neural information processing systems*, pages 540–550, 2017.
- [282] Ruihai Wu, Chuanruo Ning, and Hao Dong. Learning foresightful dense visual affordance for deformable object manipulation. In *ICCV*, 2023.
- [283] Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects. *arXiv preprint arXiv:2106.14440*, 2021.
- [284] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [285] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018.
- [286] Hongchi Xia, Zhi-Hao Lin, Wei-Chiu Ma, and Shenlong Wang. Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video. *arXiv preprint arXiv:2404.09833*, 2024.
- [287] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *CVPR*, 2020.
- [288] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.
- [289] Xiang Xu, Hanbyul Joo, Greg Mori, and Manolis Savva. D3d-hoi: Dynamic 3d human-object interactions from videos. *arXiv preprint arXiv:2108.08420*, 2021.
- [290] Xin Xu, Tianyi Xiong, Zheng Ding, and Zhuowen Tu. Masqclip for open-vocabulary universal image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 887–898, 2023.
- [291] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *TOG*, 2019.

- [292] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze. *arXiv preprint arXiv:2203.01197*, 2022.
- [293] Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. Touch and go: Learning from human-collected vision and touch. In *NeurIPS*, 2022.
- [294] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *ECCV*, 2018.
- [295] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. In *ICRA*, 2024.
- [296] Yuhang Yang, Wei Zhai, Hongchen Luo, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Grounding 3d object affordance from 2d interactions in images. 2023.
- [297] Zhenpei Yang, Jeffrey Z. Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for rgb-d scans via scene completion. In *CVPR*, 2019.
- [298] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *CVPR*, 2020.
- [299] Yufei Ye, Xueting Li, Abhinav Gupta, Shalini De Mello, Stan Birchfield, Jiaming Song, Shubham Tulsiani, and Sifei Liu. Affordance diffusion: Synthesizing hand-object interactions. In *CVPR*, 2023.
- [300] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. Shelf-supervised mesh prediction in the wild. In *CVPR*, 2021.
- [301] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.
- [302] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *ICCV*, 2019.
- [303] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021.
- [304] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.
- [305] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021.
- [306] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task

- and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.
- [307] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *CVPR*, 2019.
- [308] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.
- [309] Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on Robot Learning*, pages 284–301. PMLR, 2023.
- [310] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *CVPR*, 2019.
- [311] Jason Y. Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. Predicting 3d human dynamics from video. In *ICCV*, 2019.
- [312] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. In *Advances in Neural Information Processing Systems*, pages 2257–2268, 2018.
- [313] Yichi Zhang, Ziqiao Ma, Xiaofeng Gao, Suhaila Shakiah, Qiaozi Gao, and Joyce Chai. Groundhog: Grounding large language models to holistic segmentation. In *CVPR*, 2024.
- [314] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017.
- [315] Kai Zhao, Qi Han, Chang-Bin Zhang, Jun Xu, and Ming-Ming Cheng. Deep hough transform for semantic line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [316] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021.
- [317] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action? *Science Robotics*, 4(30):eaaw6661, 2019.
- [318] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019.
- [319] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. *arXiv preprint arXiv:2305.16986*, 2023.
- [320] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.
- [321] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

- [322] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.
- [323] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019.
- [324] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.
- [325] Daniel Zwillinger and Stephen Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.