UNIVERSITY OF CALIFORNIA,
IRVINE

Leveraging Depth for 3D Scene Perception

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Yunhan Zhao

2024

DEDICATION

To the family.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vi**

LIST OF TABLES **xii**

ACKNOWLEDGMENTS **xvi**

VITA **xvii**

ABSTRACT OF THE DISSERTATION **xix**

1 Introduction **1**
1.1 3D Scene Perception Applications 2
1.1.1 Classic Vision Tasks 2
1.1.2 Emerging Tasks 3
1.2 Depth Estimation 4
1.2.1 Sensing Techniques 4
1.2.2 Learning-based Depth Estimators 5
1.3 Contributions 7
1.4 Thesis Organization 8

2 Bridging high-level domain gaps in monocular depth predictions **11**
2.1 Background 12
2.2 Related Work 15
2.3 Attend, Remove, Complete (ARC) 17
2.3.1 Attention Module $\mathcal{A}$ 18
2.3.2 Inpainting Module $\mathcal{I}$ 19
2.3.3 Style Translator Module $\mathcal{T}$ 21
2.3.4 Depth Predictor $\mathcal{D}$ 22
2.3.5 Training by Modular Coordinate Descent 22
2.4 Experiments 23
2.4.1 Implementation Details 24
2.4.2 Indoor Scene Depth with NYUv2 & PBRS 25
2.4.3 Outdoor Scene Depth with Kitti & vKitti 26
2.4.4 Ablation Study and Qualitative Visualization 28
2.5 Conclusion and Future Work 33

3 Mitigating camera pose distribution shift in depth predictions 36
3.1 Background . . . 37
3.2 Related Work . . . 39
3.3 Perspective-aware Data Augmentation . . . 41
3.4 Depth Prediction with Camera Pose Prior . . . 43
3.5 Experiments . . . 46
3.5.1 Within & Cross-Distribution Evaluation . . . 50
3.5.2 Out-of-Distribution Evaluation . . . 51
3.5.3 Applicability to Other Predictor Networks . . . 51
3.5.4 Synthetic-to-Real Generalization . . . 53
3.5.5 Further Discussion and Ablation Study . . . 54
3.6 Conclusion . . . 61

4 Reference-based image inpainting leveraging depth maps 66
4.1 Background . . . 67
4.2 Related Work . . . 69
4.3 Method . . . 71
4.3.1 Initialization Stage . . . 72
4.3.2 Joint Optimization Stage . . . 73
4.3.3 Rendering and Postprocessing Stage . . . 76
4.4 Experiments . . . 78
4.4.1 Quantitative Results . . . 81
4.4.2 Qualitative Results . . . 82
4.4.3 Ablation Study . . . 82
4.5 Discussion, Limitations, and Conclusion . . . 92

5 Instance tracking in 3D scenes with RGBD egocentric videos 97
5.1 Background . . . 98
5.2 Related Work . . . 101
5.3 IT3DEgo: Protocol and Dataset . . . 102
5.3.1 Benchmarking Protocol . . . 103
5.3.2 Dataset . . . 105
5.4 Methodology . . . 108
5.4.1 Baseline: Re-purposed SOT Trackers . . . 108
5.4.2 Improved Baseline . . . 109
5.5 Experiments . . . 111
5.5.1 Benchmark Results . . . 113
5.5.2 Further Analysis and Ablation Study . . . 114
5.6 Discussion . . . 119
5.7 Conclusion . . . 120

6 Concluding Remarks 122
6.1 Summary of Contributions . . . 122
6.2 Future Directions . . . 124

Bibliography **126**

Appendix A Chapter 2 Supplemental Material **150**

Appendix B Chapter 3 Supplemental Material **152**

Appendix C Chapter 4 Supplemental Material **155**

Appendix D Chapter 5 Supplemental Material **157**

LIST OF FIGURES

Page

2.1 (a) The presence of novel objects and clutter can drastically degrade the output of a well-trained depth predictor. (b) Standard domain adaptation (*e.g.*, a style translator trained with CycleGAN) only changes low-level image statistics and fails to solve the problem (even trained with depth data from both synthetic and real domains), while removing the clutter entirely (c) yields a remarkably better prediction. Similarly, the insertion of a poster in (d,e) negatively affects the depth estimate and low-level domain adaptation (f) only serves to hurt overall performance. 12

2.2 Flowchart of our whole ARC model in predicting the depth given a real-world image. The ARC framework performs real-to-synthetic translation of an input image to account for low-level domain shift and simultaneously detects the "hard" out-of-domain regions using a trained attention module $\mathcal{A}$. These regions are removed by multiplicative gating with the binary mask from $\mathcal{A}$ and the masked regions inpainted by module $\mathcal{I}$. The translated result is fed to final depth predictor module $\mathcal{D}$ which is trained to estimate depth from a mix of synthetic and (translated) real data. . 16

2.3 Ablation study of the sparsity factor ρ of the ARC model on NYUv2 dataset. We use "Abs-Rel" and "δ^1" to measure the performance (see Table 2.1 for their definition). Note that the sparsity level ρ cannot be exact 1.0 due to KL loss during training, so we present an approximate value with $\rho = 0.99999$. 28

2.4 Sorted per-sample error reduction of ARC over the mix training baseline on NYUv2 dataset w.r.t. the RMS-log metric. The error reduction is computed as RMS-log(ARC) − RMS-log(mix training). The blue vertical line represents the index separating negative and positive error reduction. 29

2.5 Qualitative results list some images over which our ARC model improves the depth prediction remarkably, as well as failure cases. (Best viewed in color and zoomed in.) 30

2.6 Study of the sparsity factor ρ of the ARC model on Kitti dataset. 30

2.7 Per-sample improvement of ARC and the mix training baseline on NYUv2 testing set w.r.t RMS-log. 31

2.8 Qualitative visualizations of our ARC on Kitti dataset including improvements as well as failure cases. White arrows in the last column are used to highlight the regions over which the model improves or degrades visibly w.r.t depth prediction. We use the same color bar for the visualizing depth in each row. (Best view in color and zoomed in.) 33

2.9 Additional qualitative visualizations of our ARC on NYUv2 dataset. (Best viewed in color and zoomed in.) 35

3.1 Contemporary monocular depth predictors, *e.g.*, DORN [71], rely on large-scale training data which is naturally *biased* w.r.t. the distribution of camera poses (*e.g.*, pitch angle distribution shown in gray). As a result, DORN makes unreliable predictions on test images captured with uncommon poses (red bars), *e.g.*, pitch angles >120°. To address this issue, we propose two novel techniques that drastically reduce prediction errors (*cf*. black bars) by leveraging perspective-aware data augmentation during training and known camera pose at test time. Qualitative examples with more extreme camera pitch angles (top) show that incorporating our techniques leads to notable improvements. 37

3.2 Visual comparison of PDA and CDA. From the original example (left), conventional data augmentation (CDA) synthesizes a new example (middle) by randomly cropping a sub-region. It ignores camera pose information and will simply copy the depth values w.r.t. the corresponding pixels. In contrast, perspective-aware augmentation (PDA) simulates a rotation of the camera and synthesizes a new training example with geometrically consistent depth values corresponding to the new camera pose (right). 40

3.3 **Left:** We illustrate the proposed CPP which encodes camera pose (ω, θ, h) as a 2D image. Intuitively, for a spatial coordinate $\mathbf{q}$ on the image plane (*i.e.*, the encoding map), we find its physical point $\mathbf{p}_g$ on the ground plane, along the ray cast from the camera. Then we compute the pseudo depth value as the length from the camera to $\mathbf{p}_g^z$ which is the projection of $\mathbf{p}_g$ onto the depth direction $\mathbf{z}$ (red line) using Eqn. 3.7. This results in an encoded CPP map for a given camera pose. **Right:** We visualize some encoded CPP maps by varying the ω, θ and h independently. 41

3.4 Distribution of camera pitch and heights for three subsets of images from InteriorNet. From the *Natural* subsect, we observe the dataset of InteriorNet does have a naturally biased distribution (esp. pitch). Please refer to the text on how we construct the three subsets. 48

3.5 Breakdown analysis of depth prediction w.r.t. pitch. The background shade denotes the camera pose distribution w.r.t. pitch. Clearly, both PDA and CPP improve depth prediction in underrepresented camera poses. Surprisingly, CPP remarkably boosts depth prediction, while applying both CPP and PDA achieves the best performance "everywhere". 48

3.6 Qualitative comparison between Vanilla and our method (using both CPP and PDA) on random testing images from InteriorNet (left) and ScanNet (right). Depth maps for each example are shown with the same colorbar range. Notably, the images are captured under some uncommon camera angles relative to the training pose distribution (Fig. 3.5). The Vanilla model seems to make erroneous predictions w.r.t. the overall scale of the depth. In contrast, by applying CPP and PDA, the new model ("ours") produces visually improved results. 50

3.7 **Left:** We plot depth prediction error (Abs^r) w.r.t. different levels of noise in camera height and pitch. We apply CPP to train/test a depth predictor (based on Vanilla) on InteriorNet *Natural* train/test-set. For a given noise level δ, the trained model makes depth predictions using a CPP map computed with a perturbed camera pose, *e.g.*, the pitch is sampled from $\theta_{gt}+\mathbf{U}[-\delta, \delta]$. The black dot at the origin stands for the (best) performance using the true camera pose (*i.e.*, no noises are presented in pitch and height). The dashed lines represent the average performance levels for the Vanilla and CPP with predictive poses. **Right**: We visualize depth prediction by CPP model when encoding perturbed camera pitch angles $\theta_{gt}\pm 18°$ and heights $h_{gt}\pm 0.1$(m). CPP model predicts shallower depth when both pitch and camera height decrease (*i.e.*, camera is tilted down or translated closer to the floor). This qualitatively confirms that the camera pose prior induces a meaningful shift in the estimator. The corresponding RGB image and ground-truth depth appear in Fig. 3.8. 54

3.8 Camera pose alone provides a strong depth prior even for "blind" depth prediction. Specifically, over the InteriorNet *Natural* train-set, we train a depth predictor solely on the CPP encoded maps $\mathbf{M}$ *without* RGB as input. For visual comparison, we compute the average depth map (shown left). We visualize depth predictions on two random examples. All the depth maps are visualized with the same colormap range. Perhaps not surprisingly, $\mathbf{M}$ presents nearly the true depth in floor areas, suggesting that camera pose alone does provide strong prior depth information for these scenes. 56

3.9 During training on InteriorNet *Natural* train-set, we randomly perturb camera pose to generate new training examples. We specify the scale of the perturbation $s = \{0, 2, 4, 8, 16\}$, meaning that, when $s = 2$, we randomly perturb pitch/roll/yaw angles by adding a perturbation within $[-s * 5°, s * 5°]$. **Left**: Applying more larger scale PDA "flattens" camera pose distribution of the whole training set. **Right**: We test each of the trained models on the InteriorNet *Uniform* test-set. We find applying more intense PDA consistently improves depth prediction until $s = 16$ (*i.e.*, rotating at most 80°), presumably when very large void regions are introduced in the synthesized training examples (Fig. 3.2). 57

3.10 Visual comparisons of encoded maps of CPP and CPP-Clip with different pitch θ and camera height h. We set the threshold τ=20 for CPP-Clip. Encoded maps computed by CPP-Clip have the "red stripe" when the pitch is around 90° while CPP encoded maps have more smooth transitions when capturing the horizon. . . 59

3.11 **Uppeer:** visualizations of CPP encoding with different hyper-parameter C (top); **bottom:** depth prediction performance as a function of hyper-parameter C. We train depth predictors on InteriorNet *Natural* train-set and test on its *Natural* test-set. From visual inspection, changing the parameter C only affects the part of CPP encoded maps where pixels are above the horizon. As shown by the performance curve, our proposed CPP encoding is very robust w.r.t different values of C. . . . 60

3.12 Illustration of how camera pose provides a strong depth prior through "blind depth prediction". Specifically, over the InteriorNet *Natural* train-set, we train a depth predictor solely on the CPP encoded maps **M** *without* RGB as input. For visual comparison, we compute an averaged depth map (shown left). We visualize depth predictions on two random examples. All the depth maps are visualized with the same colormap range. Perhaps not surprisingly, **M** presents nearly the true depth in floor areas, suggesting that camera pose alone does provide strong prior depth information for these scenes. 61
3.13 **Upper row:** visualizations of augmented examples using PDA with different scales. **Bottom row:** performance curves of depth predictor trained with PDA with different scales of pitch θ (left) and roll ω (right), respectively. Please refer to Fig. 3.9 for detailed descriptions. All models are trained on InteriorNet *Natural* train-set and evaluated on both *Natural* (dotted line) and *Uniform* (solid line) test-sets. As we increase the augmentation scale in pitch, the performance of depth predictors improves until scale s=16, when large void regions are introduced in the generated examples. On the other hand, increasing augmentation scales in roll lead to steady performance increments. In general, PDA consistently improves depth prediction over a Vanilla model trained without PDA. 62
3.14 **Top:** CPP encoding with ground-truth camera height and fixed pitch. **Bottom:** CPP encoding with ground-truth pitch and fixed camera height. All models are trained/evaluated on InteriorNet *Natural* train/test-set. Comparing two blue or red curves across two plots, we find that encoding ground-truth height achieves better performance, suggesting height is "more important" than pitch. Moreover, encoding either ground-truth height or pitch outperforms Vanilla model. 63
3.15 Depth predictions of Vanilla and our model (jointly applied CPP and PDA) on InteriorNet test-set. From these images captured under various camera poses, our model predicts better depth than Vanilla model in terms of the overall scale. . . . 64
3.16 Depth predictions of Vanilla and our model (jointly applied CPP and PDA) on ScanNet test-set. From these images captured under various camera poses, our model predicts better depth than Vanilla model in terms of the overall scale. 65

4.1 Given a reference image and a target image with hole, GeoFill utilizes predicted correspondence matching and depth maps to estimate a 3D mesh and relative camera pose and intrinsics. Compared to TransFill, the previous state-of-the-art approach, GeoFill handles complex scenes better by iteratively refining predicted depth maps and relative pose. 67
4.2 Overview of our system pipeline. We are given a target image with a hole and a source image. We aim at warping the single-source image to the target to fill the hole. We first estimate the relative pose as well as predict the monocular depth of the source, and then adjust the scale and offset of the depth map. After that, to mitigate the potential errors caused by deep models, we jointly optimize camera relative pose and depth map scale and offset to make the depth map and image contents well-align near the hole region. Finally, we render the reprojected source and refine it using post-processing. 68

4.3 Qualitative comparison of GeoFill against other baselines on user-provided images (top 2 rows), RealEstate10K (mid 2 rows), and MannequinChallenge dataset (last two rows). 78
4.4 Performance gain of our method compared to TransFill w.r.t the average hole size. GeoFill has a greater advantage when the hole is larger. 86
4.5 Qualitative comparisons of our approach against TransFill with different hole sizes. **Please zoom in** to see that ours looks good but there are broken structures, ghosting, and distortion artifacts in TransFill. 87
4.6 Visual comparisons of GeoFill and TransFill without the CST module. 89
4.7 Visual plots showing the performance of GeoFill with different focal lengths. PSNR diff is computed by using GeoFill with new focal length subtract GeoFill with focal length equals to 750. 90
4.8 Qualitative results of GeoFill handling some common appearance changes such as in white balance and exposure due to camera movement. 91
4.9 Per-sample performance gain of ours compared to TransFill. The blue vertical line separates positive and negative PSNR gain. 92
4.10 Visual examples of failure cases of GeoFill. 94
4.11 Qualitatively comparison of GeoFill against other baselines on user-provided images (top 3 rows), RealEstate10K (mid 3 rows), and MannequinChallenge dataset (last 3 rows). 95
4.12 Visual illustration of inpaiting performance of GeoFill on user-provided images, RealEstate10K, and MannequinChallenge dataset. 96

5.1 **Motivation for the proposed IT3DEgo benchmark task**. We envision the real-world application of an assistive agent that continuously tracks enrolled object instances in 3D and can provide navigation guidance to users to retrieve object instances at any time. Tracked objects are either enrolled online (first row in the library) where objects of interest are identified automatically based on user interactions or pre-enrolled (bottom four rows in the library), where task-relevant objects are modeled from a collection of photos taken from different views. The former setup comes with additional in-context sensor information, such as camera pose and depth while the latter features richer visual information. 98
5.2 **Illustration of input and output of our benchmark task**. Given a raw RGB-D video sequence with camera poses and object instances of interest, i.e., either by online enrollment (SVOE) or pre-enrollment (MVPE), the goal of our benchmark task is to output the object instance 3D centers in a predefined world coordinate at each timestamp. Please check Section 5.3.1 for more details. 103
5.3 **Illustration of our benchmark dataset.** It is collected with HoloLens2 which captures RGB, depth, and four grayscale side views at 30 fps. Additionally, the device also captures per-frame camera poses allowing coarse reconstruction of the surroundings. 106
5.4 **Visualization of raw and preprocessed multi-view images**. Raw images represent the images directly output from the capture device, i.e., iPhone 13 Pro. We process raw images with segmentation and cropping before feeding them into the models. 107

5.5 **Illustration of our multi-view capture setup.** The left panel shows our camera positions when taking 25 images to support the pre-enrollment study. Specifically, we take 12 object-centric photos evenly from 360° while keeping the camera 30° elevation. Another 12 images are taken in a similar fashion while keeping the camera 60° elevation. Lastly, we take one top-down view. An example of the top-down view with the QR code is shown on the right. 108

5.6 **Qualitative visualizations of tracking with SVOE in both 3D space (left) and projected 2D view (right).** We visualize three top-performing trackers from different categories, i.e., EgoSTARK, VITKT_M, and SAM+DINOv2. For projected 2D visualization, we compare the projected 3D points of each model w.r.t to the ground-truth annotated 2D bounding boxes. In the 3D view, we show 3 concentric circles at each ground-truth position representing 0.25, 0.5 and 0.75 meter thresholds. In both 2D and 3D visualizations, we find SAM+DINOv2 outperforms others as the predictions are closer to the center of object instances. 110

5.7 **Performance comparisons of SAM+DINOv2 with different cosine thresholds.** By increasing the threshold, we find the model performance first improves and then gradually decreases. Intuitively, increasing the threshold will initially filter noisy predictions but when the threshold is too large the model will miss correct object 3D location updates. 115

5.8 **Performance w.r.t number views in MVPE.** We run SAM+DINOv2 with different numbers of views while keeping everything else the same for a fair comparison. We find the performance saturates after using 5 views. This suggests that simply encode and average features benefit from a higher number of views (i.e., number of views from 1 to 5) but still cannot fully exploit the visual information from different views (i.e., after using 5 views). 116

5.9 **Performance improvement by updating on visible only frames.** We control the memory update of SAM+DINOv2 by updating the memory only when the object instance is visible. We find the performance is significantly improved, indicating one of the major challenges of the baseline is to correctly update the memory with high quality predictions. 117

5.10 **Performance comparisons of different encoders at various cosine thresholds**. From the results, we find: (1) *Stronger encoder improve the performance.* The best performance of SAM+DINOv2 is stronger than SAM+DINO where both models have the peak performance when the cosine threshold equals 0.6. (2) *Similar performance trend w.r.t cosine similarity changes.* The performance of both models first improves and then gradually decreases when increasing the cosine threshold from 0.3 to 0.8. 118

5.11 **2D visualizations of frames from raw video sequences (upper panel) and 3D visualizations of the capture environments (lower panel)**. The benchmark videos record camera wearers perform naturalistic tasks in real-world scenarios, such as cooking and repairing. Please refer to Figure 5.3 for the layout of each sensor on the HoloLens2. 121

LIST OF TABLES

Page

2.1 A list of metrics used for evaluation in experiments, with their calculations, denoting by y and y^* the predicted and ground-truth depth in the validation set. 23

2.2 **Quantitative comparison** over NYUv2 testing set [214]. We train the state-of-the-art domain adaptation methods with the small amount of annotated real data in addition to the large-scale synthetic data. We design three baselines that only train a single depth predictor directly over synthetic or real images. Besides report *full* ARC model, we ablate each module or their combinations. We set ρ=0.85 in the attention module $\mathcal{A}$ if any, with more ablation study in Fig. 2.3. Finally, as reference, we also list a few top-performing methods that have been trained over several orders more annotated real-world frames. 26

2.3 **Quantitative comparison** over Kitti testing set [79]. The methods we compare are the same as described in Table 2.2, including three baselines, our ARC model and ablation studies, the state-of-the-art domain adaptation methods trained on both synthetic and real-world annotated data, as well as some top-performing methods on this dataset, which have been trained over three orders more annotated real-world frames from kitti videos. 27

2.4 **Quantitative comparison** between ARC and mix training baseline inside and outside of the mask region on NYUv2 testing set [214], where Δ represents the performance gain of ARC over mix training baseline under each metric. 29

3.1 **Within & cross-distribution evaluation**. In each dataset, we train depth predictors on their *Natural* train-sets and evaluate on both *Natural* and *Uniform* test-sets. We apply different methods to a Vanilla model. All models use the same network architecture. Vanilla performs poorly in cross-distribution evaluation (*cf.Natural*-test vs. *Uniform*-test), demonstrating that the biased camera pose distribution affects the training of depth predictors. As expected, RS hurts depth prediction compared to Vanilla. In contrast, CPP and PDA show better performance; jointly applying them performs the best (*i.e.*, “Both”). Finally, comparing alternative methods to our CPP (vs. Native) and PDA (vs. RS and CDA) shows the merits of our methods. 49

3.2 **Out-of-distribution evaluation**. We train depth predictors on the *Restricted* train-sets and test on both *Restricted* and *Natural* test-sets of each datasets. Vanilla model performs poorly on *Natural* test-sets, clearly showing the challenge of depth prediction on images captured under novel/never-before-seen camera poses. CPP slightly improves performance, but PDA helps more. CDA also improves performance presumably because it synthesizes more training examples, but underperforms PDA. As expected, jointly applying both CPP and PDA achieves the best performance on both *Restricted* and *Natural* test-sets. 52

3.3 **Applicability to other predictor architectures**. In each dataset, we train state-of-the-art depth predictors (DORN [71] and VNL [267]) by optionally applying our CPP and PDA approaches. All models are trained/tested on the *Natural* train/test-sets per dataset. Clearly, both CPP and PDA boost their performance. 53

3.4 **Mitigating distribution bias of camera poses improves synthetic-to-real domain adaptation**. We train depth predictors synthetically on InteriorNet (*Natural* train-set) and test them on real-world images from ScanNet *Natural* and *Uniform* test-sets. This is a typical setup for synthetic-to-real domain adaptation in the context of depth prediction. Interestingly, we find that CDA hurts the performance, presumably because the generated training examples by CDA do not obey the relations among camera model, scene geometry and camera pose, and hence do not necessarily help training a generalizable depth predictor. In contrast, our PDA helps synthetic-to-real generalization and applying CPP improves further. Importantly, applying CPP with predictive poses (CPP$_{pred}$) achieves a remarkable performance boost, whereas using the true camera pose in CPP performs the best. 55

3.5 Comparisons of different encoding methods evaluated on InteriorNet test-sets. CPP applies an inverse tangent transform $\tan^{-1}$ in encoding the camera poses. In contrast, CPP-Clip replaces the $\tan^{-1}$ function with a clipping operation while keeping every other step the same as CPP encoding. Both CPP and CPP-Clip perform better than Vanilla model, demonstrating the effectiveness of our CPP method. Clearly, using the inverse tangent operator is better than clipping. 58

3.6 Comparison between using the average depth map (Avg) computed on the InteriorNet *Natural* train-set and the "blind predictor", which estimates depth solely from per-image CPP encoded maps *without* RGB images. We report results on InteriorNet *Natural* test-set. We find that "blind predictor" performs better than "avg depth map", implying the benefit of exploiting camera poses. We also report on two specific images on which "blind predictor" performs well compared to the average performance of Avg or Blind, as shown in Fig. 3.12. This further confirms that camera poses contain useful prior knowledge about scene depth. 58

3.7 **CPP Encoding with Predicted Poses**. We train depth predictors with CPP using true poses on *Natural* train-sets of the two datasets. We test models on *Natural* and *Uniform* test-sets, respectively. Note that in testing we encode predicted poses given by a pose predictor. Clearly, CPP with predicted poses still outperforms Vanilla model; when jointly trained with PDA, CPP with predicted poses performs even better. Nevertheless, encoding predicted poses underperforms encoding true poses. 62

4.1 Quantitative comparisons of GeoFill against other baselines on the RealEstate10K dataset. . . . 80
4.2 Quantitative comparisons of GeoFill against other baselines on the MannequinChallenge dataset. . . . 81
4.3 Ablations on the objective functions in the joint optimization stage of GeoFill. . . 83
4.4 Ablations on the pixel importance weight map $\mathbf{W}$. . . . 83
4.5 Quantitative comparison of our method with initially estimated parameters and optimized parameters. . . . 85
4.6 Relative camera pose evaluation of initial guess and our optimized results, where R and t represent the rotation and translation, respectively. . . . 85
4.7 Depth evaluation of our initial guess and optimized results. The evaluation metrics include absolute relative difference (Abs^r), squared relative difference (Sq^r), root mean squared log error (RMS-log), and accuracy with a relative error threshold of $\delta^k < 1.25^k$, k = 1, 2. . . . 86
4.8 Ablations on Initial alignment comparisons of our method compared to TransFill without the CST Module. . . . 88
4.9 Initial alignment comparisons of our method compared to TransFill without the CST Module on MannequinChallenge dataset (FD=10). . . . 88
4.10 User study results of GeoFill against ProFill, OPN, and TransFill. . . . 90

5.1 **Comparisons of egocentric datasets that explore tracking-related problem.** Existing egocentric datasets only explore the tracking problem in 2D or predicting discrete 3D locations. Some mention the tracking problem in 3D but only consider limited sensor data (RGB) or synthetic environments. Our benchmark dataset supports the study of instance tracking in 3D real-world scenarios (RWS in the table) from egocentric videos. . . . 101
5.2 **Benchmark results of tracking with SVOE.** From the results, we draw three salient conclusions: (1) The ability of re-identifying object instances after they disappear is important, as long-term and egocentric specific trackers outperform short-term trackers, i.e., RGB-ST and RGB-D. (2) Currently, encoding depth maps as auxiliary information cannot improve performance since depth maps are sparse and not always perfectly aligned with RGB frames due to distortions. (3) The Kalman filter smoothing yields marginal improvements over the simple memory heuristic. The method with KF subscript indicates it applies the Kalman filter. . 112
5.3 **Benchmark results of tracking with MVPE.** We evaluate top-performing trackers in each category in Table 5.2 for MVPE setup. From the results, we have the following summaries: (1) *SOT trackers cannot fully exploit pre-enrollment information.* Detection-initialized versions perform less well compared to SVOE due to the inaccurate estimated initial bounding boxes. VITKT_M, which uses many modules that rely heavily on the initialization, degrades more significantly. (2) *Encoding rich visual information generally helps.* SAM+DINOv2 shows an even larger performance boost because it is more robust to the inaccurate initialization. The D and T superscripts indicate the detection- and template-based initializations, respectively. . . . 114

5.4 **Quantitative comparisons of 2D tracking results w/ and w/o 3D guidance.** With 3D guidance means the 2D results are computed by finding the bounding box proposal with the smallest L2 distance from projected 3D trajectories. Without 3D guidance means proposals are selected purely based on the visual feature cosine similarity. Please refer to Section 5.5.2 for more details. From the results, we find the tracking results are significantly improved with the 3D guidance, indicating that tracking in 3D in egocentric videos is much easier than in 2D by leveraging camera pose and depth sensors. 115
5.5 **Quantitative comparisons of different proposal generators.** We compare the performance of SAM+DINOv2 and YOLOv7+DINOv2. To keep the comparison fair, the only differences between these models are the proposal generators. From the results, we find adopting YOLOv7 makes the performance slightly worse. The proposal quality from YOLOv7 is lower but runs faster. 118

ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents for their unconditional support. I would not have the opportunity and determination to complete this degree without them. I am forever grateful from the bottom of my heart.

Next, I would like to thank my Ph.D. advisor Charless Fowlkes. His thorough guidance and genuine support are invaluable to me. It is my great honor and pleasure to learn alongside him, and gradually become an independent researcher. I also want to special thank Shu Kong. His research philosophy and suggestions have benefited me significantly during my entire Ph.D. study.

I would like to express my sincere gratitude to all the senior fellows who helped me in the past, including my committee members for the advancement and final defense exams: Erik Sudderth, Alexander Berg, Xiaohui Xie, Shuang Zhao, and Zyg Pizlo; my mentors and external collaborators through multiple internships: Deva Ramanan, Connelly Barnes, Yuqian Zhou, Eli Shechtman, Sohrab Amirghodsi, Taesung Park, Neil Goeckner-Wald, Harshit Gupta, and Cengiz Oztireli.

I would also like to acknowledge my dear lab mates: Daeyun Shin, Samia Shafique, Zhe Wang, Minhaeng Lee, and Xinxuan Lu. Thank you for the insightful discussions and generous help. Additionally, I also want to thank all my beloved friends who accompanied me through this long and challenging journey. Together, we have built countless precious and unforgettable memories.

Lastly, I would like to acknowledge the funding and resources that supported me during my Ph.D. degree, including NSF grants IIS-1813785, IIS-1618806, DARPA Perceptually enabled Task Guidance (PTG) Program under contract number HR00112220005, a research gift from Qualcomm, and a hardware donation from NVIDIA.

VITA

Yunhan Zhao

EDUCATION

Doctor of Philosophy in Computer Science	**Jun 2024**
University of California, Irvine	*Irvine, CA*
Master of Science in Applied Mathematics and Statistics	**May 2019**
Johns Hopkins University	*Baltimore, MD*
Master of Science in Robotics	**May 2018**
Johns Hopkins University	*Baltimore, MD*
Bachelor of Science in Mechanical Engineering	**May 2016**
Binghamton University, State University of New York	*Binghamton, NY*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2019–2024**
University of California, Irvine	*Irvine, CA*
Software Engineering Intern	**Jun 2023–Sep 2023**
Google LLC	*Mountain View, CA*
Research Intern	**Jun 2022–Sep 2022**
Adobe Inc.	*San Jose, CA*
Research Intern	**Jun 2021–Dec 2021**
Adobe Inc.	*San Jose, CA*
Research Assistant	**Jun 2020–Sep 2020**
Carnegie Mellon University, The Robotics Institute	*Pittsburgh, PA*
Research Assistant	**Feb 2018–May 2019**
Johns Hopkins University	*Baltimore, MD*
Research Assistant	**Jun 2018–Sep 2018**
Massachusetts Institute of Technology	*Cambridge, MA*

TEACHING EXPERIENCE

Teaching Assistant – University of California, Irvine	**Irvine, CA**
Machine Learning and Data Mining (CompSci 178)	*Fall 2019*
Computational Photography and Vision (CompSci 116)	*Winter 2020*
Image Understanding (CompSci 216)	*Spring 2020*

REFEREED CONFERENCE PUBLICATIONS

Y. Zhao, H. Ma, S. Kong, C. Fowlkes, "Instance Tracking in 3D Scenes from Egocentric Videos". *The IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2024

Q. Shen*, **Y. Zhao***, N. Kwon, J. Kim, Y. Li, and S. Kong. "A high-resolution dataset for instance detection with multi-view instance capture". *Thirty-seventh Conference on Neural Information Processing Systems, Datasets and Benchmarks Track,* 2023

Y. Zhao, C. Barnes, Y. Zhou, E. Shechtman, S. Amirghodsi, and C. Fowlkes. "Geofill: Reference-based image inpainting with better geometric understanding". *The IEEE Winter Conference on Applications of Computer Vision,* 2023

Y. Zhao, S. Kong, and C. Fowlkes. "Camera pose matters: Improving depth prediction by mitigating pose distribution bias". *The IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pages 15759–15768, 2021

Y. Zhao, S. Kong, D. Shin, and C. Fowlkes. "Domain decluttering: Simplifying images to mitigate synthetic-real domain shift and improve depth estimation". *The IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pages 3330–3340, 2020

Y. Zhao, Y. Tian, C. Fowlkes, W. Shen, and A. Yuille. "Resisting large data variations via introspective transformation network". *The IEEE Winter Conference on Applications of Computer Vision,* pages 3080–3089, 2020

PATENT

Y. Zhao, C. Barnes, Y. Zhou, S. Amirghodsi and E. Shechtman. "Image reprojection and multi-image inpainting based on geometric depth parameters". US20230145498A1

ABSTRACT OF THE DISSERTATION

Leveraging Depth for 3D Scene Perception

By

Yunhan Zhao

Doctor of Philosophy in Computer Science

University of California, Irvine, 2024

Professor Charless Fowlkes, Chair

3D scene perception aims to understand the geometric and semantic information of the surrounding environment. It is crucial in many downstream applications, such as autonomous driving, robotics, AR/VR, and human-computer interaction. Despite its significance, understanding the 3D scene has been a challenging task, due to the complex interactions between objects, heavy occlusions, cluttered indoor environments, major appearance, viewpoint and scale changes, *etc*. The study of 3D scene perception has been significantly reshaped by the powerful deep learning models. These models are capable of leveraging large-scale training data to achieve outstanding performance. Learning-based models unlock new challenges and opportunities in the field.

In this dissertation, we first present learning-based approaches to estimate depth maps, one of the crucial information in many 3D scene perception models. We describe two overlooked challenges in learning monocular depth estimators and present our proposed solutions. Specifically, we address the high-level domain gap between real and synthetic training data and the shift in camera pose distribution between training and testing data. Following that we present two application-driven works that leverage depth maps to achieve better 3D scene perception. We explore in detail the tasks of reference-based image inpainting and 3D object instance tracking in scenes from egocentric videos.

Chapter 1

Introduction

3D scene perception aims to comprehend the layout and geometric information of each object instance and the surrounding environment. Furthermore, it is also required to comprehend both the apparent and hidden relationships present between scene elements. These capabilities are fundamental to the way humans perceive and interpret images, thus imparting these astounding abilities in machines has been a long-standing goal in the computer vision discipline. It has wide applications in many areas, such as autonomous driving, robotics, and human-computer interaction [80, 28, 65]. 3D scene understanding is also remarkably challenging due to the complex interactions between objects, heavy occlusions, cluttered indoor environments, major appearance, viewpoint and scale changes across different scenes and inherent ambiguity, such as inferring the object size from a monocular camera.

Recent developments in large-scale data-driven deep learning [125, 98, 238, 54] have sparked a renewed interest in addressing these challenges. These models are capable of leveraging large-scale training data and extracting highly informative features from sensor data. Numerous learning-based methods have been proposed to tackle 3D scene perception problems and they significantly outperform traditional approaches [22, 63, 219, 218]. At the same time, the learning-based approach also opens new opportunities and challenges for better addressing the 3D scene perception problem.

In the remaining part of this chapter, we will first introduce some widely studied applications of 3D scene perception to highlight its importance. Following that we will present the literature of estimating depth maps, an important type of representation in understanding the 3D scene geometry. Lastly, we will discuss our contributions in the area and outline the following chapters.

1.1 3D Scene Perception Applications

Effective 3D perception of an observed scene greatly enriches the knowledge about the surrounding environment and is crucial to effectively develop high-level applications for various purposes.

1.1.1 Classic Vision Tasks

3D object detection focuses on locating and recognizing objects of the surrounding environment, which is important in the area of autonomous driving [136, 272]. The detection model generally takes multi-modal data (*e.g.*, images from cameras, point clouds from LiDAR scanners, high definition maps *etc.*) as input, and predicts the geometric (*e.g.*, locations, sizes, orientations) and semantic information (*e.g.*, pedestrians, cyclists, and vehicles) of critical elements on a road. State-of-the-art 3D detection algorithms are built on top of the LiDAR data, *i.e.*, either LiDAR only [165, 36] or fusion LiDAR and camera information [10, 153].

Motion planning is well-studied in the area of robotics. Given a task in the unstructured real-world environment, the robot needs to actively sense its surroundings, make decisions, and execute them [25, 131]. In a daily scenario of carrying an object to another room, the robot is required to precisely predict the 3D pose of the object, grasp it and plan the path to the target position. A thorough understanding of the surrounding 3D scene enables robots to execute the task accurately and smoothly.

1.1.2 Emerging Tasks

Recently, novel benchmark tasks emerged as the development of the societal need and the promising performance brought by the deep learning models. Additionally, many works begin exploring enhancing traditional pixel-level tasks with 3D geometric information.

Egocentric-related tasks. In the literature, egocentric tasks focus on activity recognition [199, 187, 113, 186], hand-object interaction [46, 150, 151]. Compared to the third-person perspective, egocentric data is captured by mounting sensors on people or robots. More egocentric-related tasks have been proposed recently due to the increased interest in AR/VR applications, such as privacy protection, activity forecasting, and social interaction predictions [88, 229, 68, 139, 173]. Many egocentric tasks are beneficial from geometric information of the surrounding environment, commonly from depth maps or SLAM models.

Image and video generation focus on generating plausible textures of one single image or a sequence of images. In the literature, learning-based models commonly leverage the large-scale training data and hallucinate realistic textures [270, 256, 271]. However, the generated texture is not guaranteed to be geometrically consistent, especially from different views. Recently, an increasing amount of models have explored the 3D representation to add geometric constraints on generated textures, such as depth or implicit 3D representations [216, 176, 172].

1.2 Depth Estimation

Pivotal to 3D scene perception is the acquisition/estimation of reliable depth information—a task for which several technologies exist, ranging from active sensors to passive methods. Additionally, recent advances in deep learning have been rapidly established and are gaining momentum.

1.2.1 Sensing Techniques

Current depth sensors are categorized into two classes: active and passive methods. Active methods, such as structured light or time of flight sensors [93, 207], emit a controlled energy signal and then receive and process the reflected energy to compute the depth information. Active methods have higher accuracy but also have the disadvantages of complicated technical implementation and higher costs. Passive methods only require the RGB images from cameras to extract the depth information. Compared with the active methods, the passive methods are more convenient and flexible, with lower software and hardware requirements, but lower measurement accuracy.

Structured light. A sequence of known patterns is sequentially projected onto an object, which gets deformed by the geometric shape of the object. The object is then observed from a camera in a different direction. By analyzing the distortion of the observed pattern, *i.e.*,, the disparity from the original projected pattern, depth information can be extracted [83, 93]. The method is independent of the color and texture of the objects and has high accuracy, especially close to the object. However, it also has its shortcomings, such as being difficult to use in strong light environments, short measurement distances, and being susceptible to reflections from smooth flat surfaces.

Time of flight (ToF). ToF by transmitting a pulsed or modulated optical signal, and then measuring the time difference of the return wavefront, since the frequency and wavelength of the optical signal are known, the depth information can be inferred [41, 129]. This process is largely independent of the scene texture and can be used for real-time depth estimation. Since the principle of ToF method is similar to the structured light method above, it is usually regarded as the twin brother of the structured light method. Compared to the structured light method, TOF is usually faster in response speed, more robust to the environment lighting conditions, and has better sensing distance.

Stereo vision. A system of stereo vision system consists of a stereo camera, i.e., two cameras placed horizontally. The two images captured simultaneously by these cameras are then processed for the recovery of visual depth information. Traditionally, stereo-based depth estimation methods relied typically on matching pixels across multiple images captured using accurately calibrated cameras [204, 228]. However, it is not suitable when dealing with occlusions, featureless regions, or highly textured regions with repetitive patterns.

1.2.2 Learning-based Depth Estimators

Contemporary depth estimators commonly adopt deep learning models as feature extractors and substantially outperform conventional models mentioned above. Depending on the availability of the ground-truth labels, the learning-based depth estimator is roughly categorized into two categories: supervised approaches and unsupervised approaches.

Supervised approaches. This class of approaches leverages the ground-truth depth maps as the supervision signal. Assuming the availability of a large training set of RGB-D pair images, learning a depth predictor is treated as a pixel-level continuous regression problem. Commonly, L1 or L2 loss function is adopted to compute pixel-wise differences between the prediction and the ground-truth depth maps.

Numerous efforts have been made to collect RGB-D data to advance the study of supervised depth estimators. Due to the expensive data collecting process, the scale of the RGB-D dataset is generally smaller compared to other vision tasks, such as image recognition. NYUv2 [214] has 1449 densely labeled pairs of aligned RGB and depth images, collected in 464 scenes from 3 different cities. KITTI [80] contains 23,488 images from 32 scenes for training and 697 images from 29 scenes for testing in the eigen split. Make3D [203] contains 534 outdoor images, 400 for training and 134 for testing. The resolution of RGB images is 2272×1704 while the ground truth depth map resolutions are 55×305.

Many supervised approaches have been developed and evaluated on previously mentioned datasets. Eigen *et al*. [63] first propose to learn the depth estimator with multi-scale features from a deep CNN. Li *et al*. [138] enhance the performance of the depth estimators by applying conditional random fields (CRFs). Laina *et al*. [128] adopts residual blocks into the depth estimation of monocular images to achieve state-of-the-art performance. Fu *et al*. [71] treat the problem as an ordinal regression task instead of continuous depth map prediction.

Unsupervised approaches. Acquiring ground truth for supervised learning is expensive since the depth maps are collected with sensors instead of human annotations. Additionally, it is also hard to get diverse data. The expensive collection procedure limits the geographical diversity in the dataset.

Motivated by the issue of data scarcity, many approaches explore learning with unlabeled real data and labeled synthetic data. However, the biggest drawback of learning with synthetic data is the domain gap [75, 233]. Specifically, models learned with synthetic data suffer from drastic performance degradation when evaluated on the real testing data. Inspired by the domain adaptation works, researchers have proposed various methods to bridge the real-synthetic domain gap. These works can be roughly divided into two categories: aligning the feature space of synthetic [291, 126, 8] and real images and translating synthetic images to realistic-looking ones [291, 8, 284]. Common synthetic datasets in the field of monocular depth estimation include SUNCG [219] and vKITTI [73].

Alternatively, another line of approach explores the relationships between images to train depth estimators. The most common relations explored in the literature are to reproject pixels to adjacent frames with predicted depth maps and compute the pixel-wise photometric loss. However, dynamic objects and low-texture regions are still a challenge for these methods. Zhou *et al*. [296] proposed an unsupervised learning framework for monocular depth estimation and camera motion estimation from unlabeled video sequences. Godard *et al*. [84] proposed Monodepth2 with an auto-masking scheme to filter out invalid pixels from moving objects and introduced a minimum reprojection loss to address occlusions.

1.3 Contributions

We briefly summarize the main contributions, which will be elaborated in the next section that outlines the thesis.

We first reveal two overlooked challenges in learning monocular depth estimators. Specifically, (1) we find that learning depth estimators have difficulty predicting the "cluttered" scenes or novel objects when learning with synthetic data. (2) we find state-of-the-art depth predictors have significant performance degradation when predicting images with underrepresented camera poses, i.e., uncommon camera poses in the training data. For both challenges, we propose novel algorithms and learning strategies, which will be elaborated in detail in the following chapters.

We also explore downstream tasks that leverage depth maps to better understand 3D scene geometry and achieve state-of-the-art performance. Specifically, we explore the task of reference-based image inpainting, which inpaints the hole region in the source image by leveraging the texture from the target image. We also introduce a new benchmark task, instance tracking in 3D scenes from egocentric videos. This task is motivated by the increasing interest in building assistive agents running on AR/VR devices. In both tasks, we showcase leveraging the 3D geometric information from depth maps drastically improves the model performance. Additionally, when exploring the latter task, we also introduce and open-source a new benchmark dataset to support the study of the new problem.

1.4 Thesis Organization

Before concluding Chapter 1 as an introduction and history survey, we now give an overview of the thesis with the subsequent chapters. The introductory paragraphs of each chapter provide more detailed outlines.

Chapter 2

We introduce our novel attend-remove-complete (ARC) model to address the unsupervised domain adaptation problem in the monocular depth prediction task. Our proposed model is based on the empirical observation that depth predictors make significant errors over the challenging regions, "cluttered" scenes or novel objects to be specific. ARC learns to carry out completion over these removed regions in order to simplify them and bring them closer to the distribution of the synthetic domain. This real-to-synthetic translation ultimately makes better use of synthetic data in producing an accurate depth estimate. We show our model significantly boosts the performance of the depth predictor both quantitatively and qualitatively. This chapter is based on our peer-reviewed conference paper [288].

Chapter 3

We present two novel approaches to mitigate the camera pose distribution shift between training and testing data. This work is motivated by the observations that state-of-the-art depth predictors show performance degradation when evaluated on images with unseen camera poses. Specifically, we explore two approaches: (1) CPP – Encode camera pose information and pass it as part of the input to learn pose-conditional depth predictors; (2) PDA – Perspective-aware data augmentation that allows us to synthesize diverse training data with different camera poses. Our experiments show both approaches boost the performance of the depth predictor and combining them achieve even better performance. This chapter is based on our peer-reviewed conference paper [287].

Chapter 4

We describe a novel reference-based image inpainting model named GeoFill. Given a target image, a mask and a reference image, GeoFill inpaints the hole region by leveraging the textures from the source images. Specifically, GeoFill first reconstructs the non-planar geometry from predicted depth maps. A joint optimization module is adopted to refine the depth maps and camera poses. Based on both quantitative and qualitative results, GeoFill outperforms state-of-the-art approaches. This chapter is based on our peer-reviewed conference paper [286].

Chapter 5

In this chapter, we present a novel benchmark – instance tracking in 3D scenes from egocentric videos (IT3DEgo). Egocentric sensors such as AR/VR devices capture human-object interactions and offer the potential to provide task-assistance by recalling 3D locations of objects of interest in the surrounding environment. Motivated by this, we explore this problem by first introducing a new benchmark dataset, consisting of RGB and depth videos, per-frame camera pose, and instance-level annotations in both 2D camera and 3D world coordinates. We also propose a novel algorithm that leverages the recent foundation models to achieve state-of-the-art performance. This chapter is based on our peer-reviewed conference paper [289].

Chapter 6

This is the last chapter of this thesis. We revisit our contributions and outline directions for future research.

Chapter 2

Bridging high-level domain gaps in monocular depth predictions

Leveraging synthetically rendered data offers great potential to improve monocular depth estimation and other geometric estimation tasks, but closing the synthetic-real domain gap is a non-trivial and important task. While much recent work has focused on unsupervised domain adaptation, we consider a more realistic scenario where a large amount of synthetic training data is supplemented by a small set of real images with ground-truth. In this setting, we find that existing domain translation approaches are difficult to train and offer little advantage over simple baselines that use a mix of real and synthetic data. A key failure mode is that real-world images contain novel objects and clutter not present in synthetic training. This high-level domain shift isn't handled by existing image translation models.

Based on these observations, we develop an attention module that learns to identify and remove difficult out-of-domain regions in real images in order to improve depth prediction for a model trained primarily on synthetic data. We carry out extensive experiments to validate our attend-remove-complete approach (ARC) and find that it significantly outperforms state-

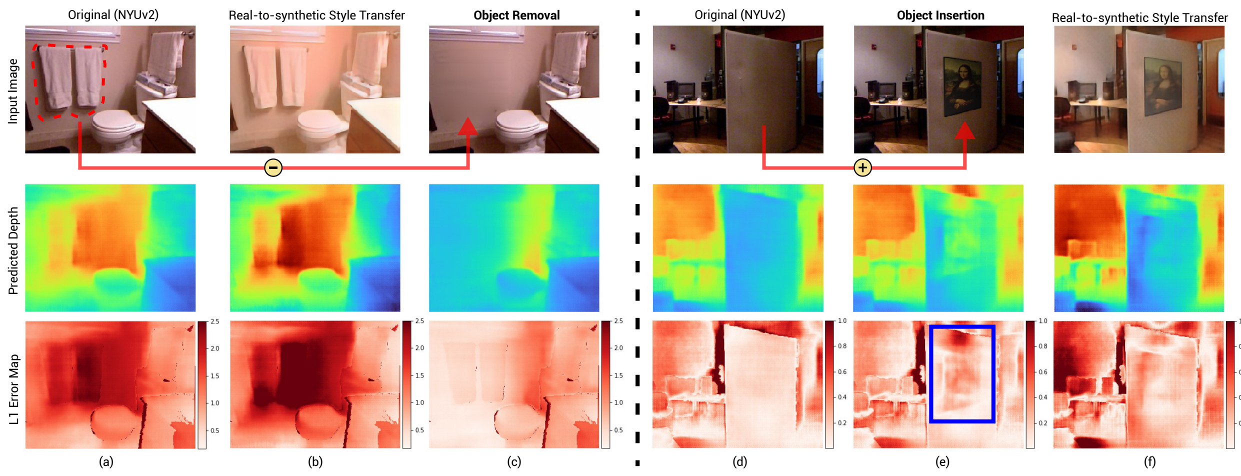


Figure 2.1: (a) The presence of novel objects and clutter can drastically degrade the output of a well-trained depth predictor. (b) Standard domain adaptation (*e.g.*, a style translator trained with CycleGAN) only changes low-level image statistics and fails to solve the problem (even trained with depth data from both synthetic and real domains), while removing the clutter entirely (c) yields a remarkably better prediction. Similarly, the insertion of a poster in (d,e) negatively affects the depth estimate and low-level domain adaptation (f) only serves to hurt overall performance.

of-the-art domain adaptation methods for depth prediction. Visualizing the removed regions provides interpretable insights into the synthetic-real domain gap.

2.1 Background

With a graphics rendering engine one can, in theory, synthesize an unlimited number of scene images of interest and their corresponding ground-truth annotations [273, 122, 280, 220]. Such large-scale synthetic data increasingly serves as a source of training data for high-capacity convolutional neural networks (CNN). Leveraging synthetic data is particularly important for tasks such as semantic segmentation that require fine-grained labels at each pixel and can be prohibitively expensive to manually annotate. Even more challenging are pixel-level regression tasks where the output space is continuous. One such task, the focus of this chapter, is monocular depth estimation, where the only available ground-truth for real-world images comes from specialized sensors that typically provide noisy and incomplete estimates.

Due to the domain gap between synthetic and real-world imagery, it is non-trivial to leverage synthetic data. Models naively trained over synthetic data often do not generalize well to the real-world images [74, 154, 232]. Therefore domain adaptation problem has attracted increasing attention from researchers aiming at closing the domain gap through unsupervised generative models (*e.g.* using GAN [86] or CycleGAN [301]). These methods assume that domain adaptation can be largely resolved by learning a domain-invariant feature space or translating synthetic images into realistic-looking ones. Both approaches rely on an adversarial discriminator to judge whether the features or translated images are similar across domains, without specific consideration of the task in question. For example, CyCADA translates images between synthetic and real-world domains with domain-wise cycle-constraints and adversarial learning [104]. It shows successful domain adaptation for multiple vision tasks where only the synthetic data have annotations while real ones do not. T^2Net exploits adversarial learning to penalize the domain-aware difference between both images and features [291], demonstrating successful monocular depth learning where the synthetic data alone provides the annotation for supervision.

Despite these successes, we observe two critical issues:

(1) Low-level vs. high-level domain adaptation. As noted in the literature [109, 302], unsupervised GAN models are limited in their ability to translate images and typically only modify low-level factors, *e.g.*, color and texture. As a result, current GAN-based domain translation methods are ill-equipped to deal with the fact that images from different domains contain high-level differences (*e.g.*, novel objects present only in one domain), that cannot be easily resolved. Figure 2.1 highlights this difficulty. High-level domain shifts in the form of novel objects or clutter can drastically disrupt predictions of models trained on synthetic images. To combat this lack of robustness, we argue that a better strategy may be to explicitly identify and remove these unknowns rather than letting them corrupt model predictions.

(2) Input vs. output domain adaptation. Unlike domain adaptation for image classification where appearances change but the set of labels stays constant, in depth regression the domain shift is not just in the appearance statistics of the input (image) but also in the statistics of the output (scene geometry). To understand how the statistics of geometry shifts between synthetic and real-world scenes, it is necessary that we have access to at least some real-world ground-truth. This precludes solutions that rely entirely on unsupervised domain adaptation. However, we argue that a likely scenario is that one has available a small quantity of real-world ground-truth along with a large supply of synthetic training data. As shown in our experiments, when we try to tailor existing unsupervised domain adaptation methods to this setup, surprisingly we find that none of them perform satisfactorily and sometimes even worse than simply training with small amounts of real data!

Motivated by these observations, we propose a principled approach that improves depth prediction on real images using a somewhat unconventional strategy of translating real images to make them more similar to the available bulk of synthetic training data. Concretely, we introduce an attention module that learns to detect problematic regions (*e.g.*, novel objects or clutter) in real-world images. Our attention module produces binary masks with the differentiable Gumbel-Max trick [89, 110, 239, 120], and uses the binary mask to remove these regions from the input images. We then develop an inpainting module that learns to complete the erased regions with realistic fill-in. Finally, a translation module adjusts the low-level factors such as color and texture to match the synthetic domain.

We name our translation model ARC, as it attends, removes and completes the real-world image regions. To train our ARC model, we utilize a modular coordinate descent training pipeline where we carefully train each module individually and then fine-tune as a whole to optimize depth prediction performance. We find this approach is necessary since, as with other domain translation methods, the multiple losses involved compete against each other and do not necessarily contribute to improve depth prediction.

To summarize our main contributions:

- We study the problem of leveraging synthetic data along with a small amount of annotated real data for learning better depth prediction, and reveal the limitations of current unsupervised domain adaptation methods in this setting.
- We propose a principled approach (ARC) that learns identify, remove and complete "hard" image regions in real-world images, such that we can translate the real images to close the synthetic-real domain gap to improve monocular depth prediction.
- We carry out extensive experiments to demonstrate the effectiveness of our ARC model, which not only outperforms state-of-the-art methods, but also offers good interpretability by explaining what to remove in the real images for better depth prediction.

2.2 Related Work

Learning from Synthetic Data is a promising direction in solving data scarcity, as the render engine could in theory produce unlimited number of synthetic data and their perfect annotations used for training. Many synthetic datasets have been released [280, 220, 73, 137, 26, 55], for various pixel-level prediction tasks like semantic segmentation, optical flow, and monocular depth prediction. A large body of work uses synthetic data to augment real-world datasets, which are already large in scale, to further improve the performance [144, 55, 237]. We consider a problem setting in which only a limited set of annotated real-world training data is available along with a large pool of synthetic data.

Synthetic-Real Domain Adaptation. Models trained purely on synthetic data often suffer limited generalization [177]. Assuming there is no annotated real-world data during training, one approach is to close synthetic-real domain gap with the help of adversarial training. These methods learn either a domain invariant feature space or an image-to-image translator that maps between images from synthetic and real-world domains. For the former, [155] introduces Maximum Mean Discrepancy to learn domain invariant features; [234] jointly minimizes MMD

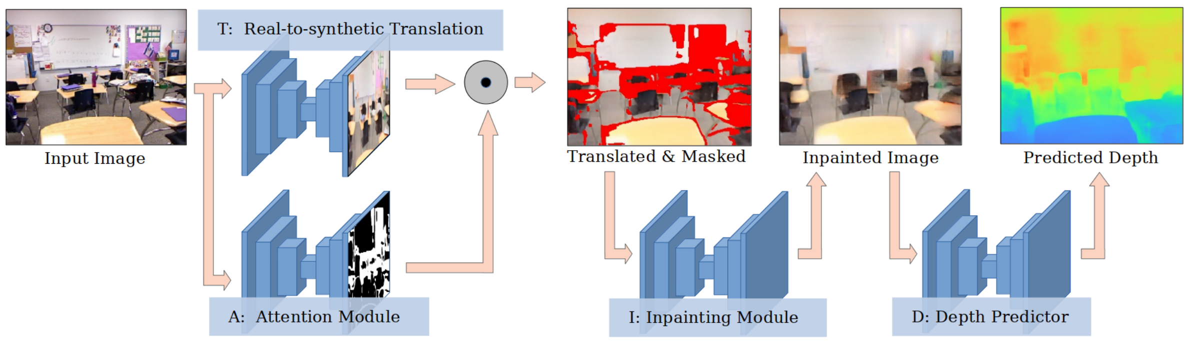


Figure 2.2: Flowchart of our whole ARC model in predicting the depth given a real-world image. The ARC framework performs real-to-synthetic translation of an input image to account for low-level domain shift and simultaneously detects the "hard" out-of-domain regions using a trained attention module $\mathcal{A}$. These regions are removed by multiplicative gating with the binary mask from $\mathcal{A}$ and the masked regions inpainted by module $\mathcal{I}$. The translated result is fed to final depth predictor module $\mathcal{D}$ which is trained to estimate depth from a mix of synthetic and (translated) real data.

and classification error to further improve domain adaptation performance; [233, 231] apply adversarial learning to aligning source and target domain features; [224] proposes to match the mean and variance of domain features. For the latter, CyCADA learns to translate images from synthetic and real-world domains with domain-wise cycle-constraints and adversarial learning [104]. T^{2}Net exploits adversarial learning to penalize the domain-aware difference between both images and features [291], demonstrating successful monocular depth learning where the synthetic data alone provide the annotation for supervision.

Attention and Interpretability. Our model utilizes a learnable attention mechanism similar to those that have been widely adopted in the community [225, 238, 78], improving not only the performance for the task in question [171, 120], but also improving interpretability and robustness from various perspectives [7, 6, 239, 66]. Specifically, we utilize the Gumbel-Max trick [89, 110, 239, 120], to learn binary decision variables in a differentiable training framework. This allows for efficient training while producing easily interpretable results that indicate which regions of real images introduce errors that hinder the performance of models trained primarily on synthetic data.

2.3 Attend, Remove, Complete (ARC)

Recent methods largely focus on how to leverage synthetic data (and their annotations) along with real-world images (where no annotations are available) to train a model that performs well on real images later [104, 291]. We consider a more relaxed (and we believe realistic) scenario in which there is a small amount of real-world ground-truth data available during training. More formally, given a set of real-world labeled data $X^r = \{\mathbf{x}_i^r, \mathbf{y}_i^r\}_{i=1}^M$ and a large amount of synthetic data $X^s = \{\mathbf{x}_j^s, \mathbf{y}_j^s\}_{j=1}^N$, where $M \ll N$, we would like to train a monocular depth predictor $\mathcal{D}$, that accurately estimates per-pixel depth on real-world test images. The challenges of this problem are two-fold. First, due to the synthetic-real domain gap, it is not clear when including synthetic training data improves the test-time performance of a depth predictor on real images. Second, assuming the model does indeed benefit from synthetic training data, it is an open question as to how to best leverage the knowledge of domain difference between real and synthetic.

Our experiments positively answer the first question: synthetic data can be indeed exploited for learning better depth models, but in a non-trivial way as shown later through experiments. Briefly, real-world images contain complex regions (*e.g.*, rare objects), which do not appear in the synthetic data. Such complex regions may negatively affect depth prediction by a model trained over large amounts of synthetic, clean images. Figure 2.2 demonstrates the inference flowchart of **ARC**, which learns to *attend*, *remove* and *complete* challenging regions in real-world test images in order to better match the low- and high-level domain statistics of synthetic training data. In this section, we elaborate each component module, and finally present the training pipeline.

2.3.1 Attention Module $\mathcal{A}$

How might we automatically discover the existence and appearance of "hard regions" that negatively affect depth learning and prediction? Such regions are not just those which are rare in the real images, but also include those which are common in real images but absent from our pool of synthetic training data. Finding such "hard regions" thus relies on both the depth predictor itself and synthetic data distribution. To discover this complex dependence, we utilize an attention module $\mathcal{A}$ that learns to automatically detect such "hard regions" from the real-world input images. Given a real image $\mathbf{x}^r \in \mathbb{R}^{H \times W \times 3}$ as input, the attention module produces a binary mask $\mathbf{M} \in \mathbb{R}^{H \times W}$ used for erasing the "hard regions" using simple Hadamard product $\mathbf{M} \odot \mathbf{x}^r$ to produce the resulting masked image.

One challenge is that producing a binary mask typically involves a hard-thresholding operation which is non-differentiable and prevents from end-to-end training using backpropagation. To solve this, we turn to the Gumbel-Max trick [89] that produces quasi binary masks using a continuous relaxation [110, 163].

We briefly summarize the "Gumbel max trick" [110, 163, 239]. A random variable g follows a Gumbel distribution if $g = -\log(-\log(u))$, where u follows a uniform distribution $U(0, 1)$. Let m be a discrete binary random variable[1] with probability $P(m = 1) \propto \alpha$, and let g be a Gumbel random variable. Then, a sample of m can be obtained by sampling g from the Gumbel distribution and computing:

$$m = \text{sigmoid}((\log(\alpha) + g)/\tau), \tag{2.1}$$

where the temperature $\tau \to 0$ drives the m to take on binary values and approximates the non-differentiable argmax operation. We use this operation to generate a binary mask of size $\mathbf{M} \in \mathbb{R}^{H \times W}$.

[1]A binary variable $m = 0$ indicates the current pixel will be removed.

To control the sparsity of the output mask $\mathbf{M}$, we penalize the empirical sparsity of the mask $\xi = \frac{1}{H*W}\sum_{i,j}^{H,W}\mathbf{M}_{i,j}$ using a KL divergence loss term [120]:

$$\ell_{KL} = \rho\log(\frac{\rho}{\xi}) + (1-\rho)\log(\frac{1-\rho}{1-\xi}). \tag{2.2}$$

where hyperparameter ρ controls the sparsity level (portion of pixels to keep). We apply the above loss term ℓ_{KL} in training our whole system, forcing the attention module $\mathcal{A}$ to identify the hard regions in an "intelligent" manner to target a given level of sparsity while still remaining the fidelity of depth predictions on the translated image. We find that training in conjunction with the other modules results in attention masks that tend to remove regions instead of isolated pixels (see Fig. 2.5)

2.3.2 Inpainting Module $\mathcal{I}$

The previous attention module $\mathcal{A}$ removes hard regions in $\mathbf{x}^r$[2] with sparse, binary mask $\mathbf{M}$, inducing holes in the image with the operation $\mathbf{M}\odot\mathbf{x}^r$. To avoid disrupting depth prediction we would like to fill in some reasonable values (without changing unmasked pixels). To this end, we adopt an inpainting module $\mathcal{I}$ that learns to fill in holes by leveraging knowledge from synthetic data distribution as well as the depth prediction loss. Mathematically we have:

$$\tilde{\mathbf{x}} = (1-\mathbf{M})\odot\mathcal{I}(\mathbf{M}\odot\mathbf{x}^r) + \mathbf{M}\odot\mathbf{x}^r. \tag{2.3}$$

To train the inpainting module $\mathcal{I}$, we pretrain with a self-supervised method by learning to

[2]Here, we present the inpainting module as a standalone piece. The final pipeline is shown in Fig. 2.2

reconstruct randomly removed regions using the reconstruction loss ℓ_{rec}^{rgb}:

$$\ell_{rec}^{rgb} = \mathbb{E}_{x^r \sim X^r}[||\mathcal{I}(\mathbf{M} \odot \mathbf{x}^r) - \mathbf{x}^r||_1] \tag{2.4}$$

As demonstrated in [211], ℓ_{rec}^{rgb} encourages the model to learn remove objects instead of reconstructing the original images since removed regions are random. Additionally, we use two perceptual losses [278]. The first penalizes feature reconstruction:

$$\ell_{rec}^{f} = \sum_{k=1}^{K} \mathbb{E}_{\mathbf{x}^r \sim X^r}[||\phi_k(\mathcal{I}(\mathbf{M} \odot \mathbf{x}^r)) - \phi_k(\mathbf{x}^r)||_1], \tag{2.5}$$

where $\phi_k(\cdot)$ is the output feature at the k^{th} layer of a VGG16 pretrained model [215]. The second perceptual loss is a style reconstruction loss that penalizes the differences in colors, textures, and common patterns.

$$\ell_{style} = \sum_{k=1}^{K} \mathbb{E}_{x^r \sim X^r}[||\sigma_k^{\phi}(\mathcal{I}(\mathbf{M} \odot \mathbf{x}^r)) - \sigma_k^{\phi}(\mathbf{x}^r)||_1], \tag{2.6}$$

where function $\sigma_k^{\phi}(\cdot)$ returns a Gram matrix. For the feature $\phi_k(\mathbf{x})$ of size $C_k \times H_k \times W_k$, the corresponding Gram matrix $\sigma_k^{\phi}(\mathbf{x}^r) \in \mathbb{R}^{C_k \times C_k}$ is computed as:

$$\sigma_k^{\phi}(\mathbf{x}^r) = \frac{1}{C_k H_k W_k} R(\phi_k(\mathbf{x}^r)) \cdot R(\phi_k(\mathbf{x}^r))^T, \tag{2.7}$$

where $R(\cdot)$ reshapes the feature $\phi_k(\mathbf{x})$ into $C_k \times H_k\ W_k$.

Lastly, we incorporate an adversarial loss ℓ_{adv} to force the inpainting module $\mathcal{I}$ to fill in pixel values that follow the *synthetic* data distribution:

$$\ell_{adv} = \mathbb{E}_{x^r \sim X^r}[\log(D(\tilde{\mathbf{x}}))] + \mathbb{E}_{\mathbf{x}^s \sim X^s}[\log(1 - D(\mathbf{x}^s)], \tag{2.8}$$

where D is a discriminator with learnable weights that is trained on the fly. To summarize, we use the following loss function to train our inpainting module $\mathcal{I}$:

$$\ell_{inp} = \ell_{rec}^{rgb} + \lambda_f \cdot \ell_{rec}^{f} + \lambda_{style} \cdot \ell_{style} + \lambda_{adv} \cdot \ell_{adv}, \tag{2.9}$$

where we set weight parameters as $\lambda_f = 1.0$, $\lambda_{style} = 1.0$, and $\lambda_{adv} = 0.01$.

2.3.3 Style Translator Module $\mathcal{T}$

The style translator module $\mathcal{T}$ is the final piece to translate the real images into the synthetic data domain. As the style translator adapts low-level feature (*e.g.*, color and texture) we apply it prior to inpainting. Following the literature, we train the style translator in a standard CycleGAN [301] pipeline, by minimizing the following loss:

$$\ell_{cycle} = \mathbb{E}_{x^r \sim X^r}[||G_{s2r}(G_{r2s}(x^r)) - x^r||_1] + \mathbb{E}_{x^s \sim X^s}[||G_{r2s}(G_{s2r}(x^s)) - x^s||_1], \tag{2.10}$$

where $\mathcal{T} = G_{r2s}$ is the translator from direction real to synthetic domain; while G_{s2r} is the other way around. Note that we need two adversarial losses ℓ_{adv}^r and ℓ_{adv}^s in the form of Eqn.(2.8) along with the cycle constraint loss ℓ_{cycle}. We further exploit the identity mapping constraint to encourage translators to preserve the geometric content and color composition between original and translated images:

$$\ell_{id} = \mathbb{E}_{x^r \sim X^r}[||G_{s2r}(x^r) - x^r||_1] + \mathbb{E}_{x^s \sim X^s}[||G_{r2s}(x^s) - x^s||_1]. \tag{2.11}$$

To summarize, the overall objective function for training the style translator $\mathcal{T}$ is:

$$\ell_{trans} = \lambda_{cycle} \cdot \ell_{cycle} + \lambda_{id} \cdot \ell_{id} + (\ell_{adv}^r + \ell_{adv}^s), \tag{2.12}$$

where we set the weights $\lambda_{cycle} = 10.0$, $\lambda_{id} = 5.0$.

2.3.4 Depth Predictor $\mathcal{D}$

We train our depth predictor $\mathcal{D}$ over the combined set of translated real training images $\tilde{\mathbf{x}}^r$ and synthetic images $\mathbf{x}^s$ using a simple L_1 norm based loss:

$$\ell_d = \mathbb{E}_{(\mathbf{x}^r,\mathbf{y}^r)\sim X^r}||\mathcal{D}(\tilde{\mathbf{x}}^r) - y^r||_1 + \mathbb{E}_{(\mathbf{x}^s,\mathbf{y}^s)\sim X^s}||\mathcal{D}(\mathbf{x}^s) - \mathbf{y}^s||_1. \tag{2.13}$$

2.3.5 Training by Modular Coordinate Descent

In principle, one might combine all the loss terms to train the ARC modules jointly. However, we found such practice difficult due to several reasons: bad local minima, mode collapse within the whole system, large memory consumption, etc. Instead, we present our proposed training pipeline that trains each module individually, followed by a fine-tuning step over the whole system. We note such a modular coordinate descent training protocol has been exploited in prior work, such as block coordinate descent methods [164, 3], layer pretraining in deep models [101, 215], stage-wise training of big complex systems [15, 39] and those with modular design [7, 66].

Concretely, we train the depth predictor module $\mathcal{D}$ by feeding the original images from either the synthetic set, or the real set or the mix of the two as a pretraining stage. For the synthetic-real style translator module $\mathcal{T}$, we first train it with CycleGAN. Then we insert the attention module $\mathcal{A}$ and the depth predictor module $\mathcal{D}$ into this CycleGAN, but fixing $\mathcal{D}$ and $\mathcal{T}$, and train the attention module $\mathcal{A}$ only. Note that after training the attention module $\mathcal{A}$, we fix it without updating it any more and switch the Gumbel transform to output real binary maps, on the assumption that it has already learned what to attend and remove with the help of depth loss and synthetic distribution. We train the inpainting module $\mathcal{I}$ over translated real-world images and synthetic images.

Table 2.1: A list of metrics used for evaluation in experiments, with their calculations, denoting by y and y^* the predicted and ground-truth depth in the validation set.

Abs Relative diff. (Rel)	$\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert y-y^*\rvert/y^*$
Squared Relative diff. (Sq-Rel)	$\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert y-y^*\rvert\rvert^2/y^*$
RMS	$\sqrt{\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert y_i-y_i^*\rvert\rvert^2}$
RMS-log	$\sqrt{\frac{1}{\lvert T\rvert}\sum_{y\in T}\lvert\lvert \log y_i-\log y_i^*\rvert\rvert^2}$
Threshold δ^i, $i\in\{1,2,3\}$	% of y_i s.t. $\max(\frac{y_i}{y_i^*},\frac{y_i^*}{y_i})<1.25^i$

The above procedure yields good initialization for all the modules, after which we may keep optimizing them one by one while fixing the others. In practice, simply fine-tune the whole model (still fixing $\mathcal{A}$) with the depth loss term only, by removing all the adversarial losses. To do this, we alternate between minimizing the following two losses:

$$\ell_d^1 = \mathbb{E}_{(\mathbf{x}^r,\mathbf{y}^r)\sim X^r}||\mathcal{D}(\mathcal{I}(\mathcal{T}(\mathbf{x}^r)\odot\mathcal{A}(\mathbf{x}_r)))) - \mathbf{y}^r||_1 + \mathbb{E}_{(\mathbf{x}^s,\mathbf{y}^s)\sim X^s}||\mathcal{D}(\mathbf{x}^s) - \mathbf{y}^s||_1, \tag{2.14}$$

$$\ell_d^2 = \mathbb{E}_{(\mathbf{x}^r,\mathbf{y}^r)\sim X^r}||\mathcal{D}(\mathcal{I}(\mathcal{T}(\mathbf{x}^r)\odot\mathcal{A}(\mathbf{x}_r)))) - \mathbf{y}^r||_1. \tag{2.15}$$

We find in practice that such fine-tuning better exploits synthetic data to avoid overfitting on the translated real images, and also avoids catastrophic forgetting [70, 119] on the real images in face of overwhelmingly large amounts of synthetic data.

2.4 Experiments

We carry out extensive experiments to validate our ARC model in leveraging synthetic data for depth prediction. We provide systematic ablation study to understand the contribution of each module and the sparsity of the attention module $\mathcal{A}$. We further visualize the intermediate results produced by ARC modules, along with failure cases, to better understand the whole ARC model and the high-level domain gap.

2.4.1 Implementation Details

Network Architecture. Every single module in our ARC framework is implemented by a simple encoder-decoder architecture as used in [301], which also defines our discriminator's architecture. We modify the decoder to output a single channel to train our attention module $\mathcal{A}$. As for the depth prediction module, we further add skip connections that help output high-resolution depth estimate [291].

Training. We first train each module individually for 50 epochs using the Adam optimizer [116], with initial learning rate $5e$-5 ($1e$-4 for discriminator if adopted) and coefficients 0.9 and 0.999 for computing running averages of gradient and its square. Then we fine-tune the whole ARC model with the proposed modular coordinate descent scheme with the same learning parameters.

Datasets. We evaluate on indoor scene and outdoor scene datasets. For indoor scene depth prediction, we use the real-world NYUv2 [214] and synthetic Physically-based Rendering (PBRS) [280] datasets. NYUv2 contains video frames captured using Microsoft Kinect, with 1,449 test frames and a large set of video (training) frames. From the video frames, we randomly sample 500 as our small amount of labeled real data (no overlap with the official testing set). PBRS contains large-scale synthetic images generated using the Mitsuba renderer and SUNCG CAD models [220]. We randomly sample 5,000 synthetic images for training. For outdoor scene depth prediction, we turn to the Kitti [79] and virtual Kitti (vKitti) [73] datasets. In Kitti, we use the Eigen testing set to evaluate [296, 84] and the first 1,000 frames as the small amount of real-world labeled data for training [63]. With vKitti, we use the split {clone, 15-deg-left, 15-deg-right, 30-deg-left, 30-deg-right} to form our synthetic training set consisting of 10,630 frames. Consistent with previous work, we clip the maximum depth in vKitti to 80.0m for training, and report performance on Kitti by capping at 80.0m for a fair comparison.

Comparisons and Baselines. We compare four classes of models. Firstly we have three baselines that train a single depth predictor on only synthetic data, only real data, or the combined set. Secondly we train state-of-the-art domain adaptation methods (T^{2}Net [291], CrDoCo [38] and GASDA [284]) with their released code. During training, we also modify them to use the small amount of annotated real-world data in addition to the large-scale synthetic data. We note that these methods originally perform unsupervised domain adaptation, but they perform much worse than our modified baselines which leverage some real-world training data. This supports our suspicion that multiple losses involved in these methods (*e.g.*, adversarial loss terms) do not necessarily contribute to reducing the depth loss. Thirdly, we have our ARC model and ablated variants to evaluate how each module helps improve depth learning. The fourth group includes a few top-performing fully-supervised methods which were trained specifically for the dataset over annotated real images only, but at a much larger scale For example, DORN [71] trains over more than 120K/20K frames for NYUv2/Kitti, respectively. This is 200/40 times larger than the labeled real images for training our ARC model.

Evaluation metrics. for depth prediction are standard and widely adopted in literature, as summarized in Table 2.1.

2.4.2 Indoor Scene Depth with NYUv2 & PBRS

Table 2.2 lists detailed comparison for indoor scene depth prediction. We observe that ARC outperforms other unsupervised domain adaptation methods by a substantial margin. This demonstrates two aspects. First, these domain adaptation methods have adversarial losses that force translation between domains to be more realistic, but there is no guarantee that "more realistic" is beneficial for depth learning. Second, removing "hard" regions in real images makes the real-to-synthetic translation easier and more effective for leveraging

Table 2.2: **Quantitative comparison** over NYUv2 testing set [214]. We train the state-of-the-art domain adaptation methods with the small amount of annotated real data in addition to the large-scale synthetic data. We design three baselines that only train a single depth predictor directly over synthetic or real images. Besides report *full* ARC model, we ablate each module or their combinations. We set ρ=0.85 in the attention module $\mathcal{A}$ if any, with more ablation study in Fig. 2.3. Finally, as reference, we also list a few top-performing methods that have been trained over several orders more annotated real-world frames.

Model/metric	↓ lower is better				↑ better		
	Rel	Sq-Rel	RMS	RMS-log	δ^1	δ^2	δ^3
State-of-the-art domain adaptation methods (*w/ real labeled data*)							
T^2Net [291]	0.202	0.192	0.723	0.254	0.696	0.911	0.975
CrDoCo [38]	0.222	0.213	0.798	0.271	0.667	0.903	0.974
GASDA [284]	0.219	0.220	0.801	0.269	0.661	0.902	0.974
Our (baseline) models.							
syn only	0.299	0.408	1.077	0.371	0.508	0.798	0.925
real only	0.222	0.240	0.810	0.284	0.640	0.885	0.967
mix training	0.200	0.194	0.722	0.257	0.698	0.911	0.975
ARC: $\mathcal{T}$	0.226	0.218	0.805	0.275	0.636	0.892	0.974
ARC:$\mathcal{A}$	0.204	0.208	0.762	0.268	0.681	0.901	0.971
ARC: $\mathcal{A}$&$\mathcal{T}$	0.189	0.181	0.726	0.255	0.702	0.913	0.976
ARC: $\mathcal{A}$&$\mathcal{I}$	0.195	0.191	0.731	0.259	0.698	0.909	0.974
ARC: *full*	**0.186**	**0.175**	**0.710**	**0.250**	**0.712**	**0.917**	**0.977**
Training over large-scale NYUv2 video sequences (>120K)							
DORN [71]	0.115	–	0.509	0.051	0.828	0.965	0.992
Laina [128]	0.127	–	0.573	0.055	0.811	0.953	0.988
Eigen [62]	0.158	0.121	0.641	0.214	0.769	0.950	0.988

synthetic data in terms of depth learning. The second point will be further verified through qualitative results. We also provide an ablation study adding in the attention module $\mathcal{A}$ leads to better performance than merely adding the synthetic-real style translator $\mathcal{T}$. This shows the improvement brought by $\mathcal{A}$. However, combining $\mathcal{A}$ with either $\mathcal{T}$ or $\mathcal{I}$ improves further, while $\mathcal{A}$ & $\mathcal{T}$ is better as removing the hard real regions more closely matches the synthetic training data.

2.4.3 Outdoor Scene Depth with Kitti & vKitti

We train the same set of domain adaptation methods and baselines on the outdoor data, and report detailed comparisons in Table 2.3. We observe similar trends as reported in

Table 2.3: **Quantitative comparison** over Kitti testing set [79]. The methods we compare are the same as described in Table 2.2, including three baselines, our ARC model and ablation studies, the state-of-the-art domain adaptation methods trained on both synthetic and real-world annotated data, as well as some top-performing methods on this dataset, which have been trained over three orders more annotated real-world frames from kitti videos.

Model/metric	↓ `lower is better`				↑ `better`		
	Rel	Sq-Rel	RMS	RMS-log	δ^1	δ^2	δ^3
State-of-the-art domain adaptation methods (*w/ real labeled data*)							
T^{2}Net [291]	0.151	0.993	4.693	0.253	0.791	0.914	0.966
CrDoCo [38]	0.275	2.083	5.908	0.347	0.635	0.839	0.930
GASDA [284]	0.253	1.802	5.337	0.339	0.647	0.852	0.951
Our (baseline) models.							
syn only	0.291	3.264	7.556	0.436	0.525	0.760	0.881
real only	0.155	1.050	4.685	0.250	0.798	0.922	0.968
mix training	0.152	0.988	4.751	0.257	0.784	0.918	0.966
ARC: $\mathcal{T}$	0.156	1.018	5.130	0.279	0.757	0.903	0.959
ARC: $\mathcal{A}$	0.154	0.998	5.141	0.278	0.761	0.908	0.962
ARC: $\mathcal{A}$&$\mathcal{T}$	0.147	0.947	4.864	0.259	0.784	0.916	0.966
ARC: $\mathcal{A}$&$\mathcal{I}$	0.152	0.995	5.054	0.271	0.766	0.908	0.962
ARC: *full*	**0.143**	**0.927**	**4.679**	**0.246**	**0.798**	**0.922**	**0.968**
Training over large-scale kitti video frames (>20K)							
DORN [71]	0.071	0.268	2.271	0.116	0.936	0.985	0.995
DVSO [261]	0.097	0.734	4.442	0.187	0.888	0.958	0.980
Guo [90]	0.096	0.641	4.095	0.168	0.892	0.967	0.986

the indoor scenario in Table 2.2. Specifically, $\mathcal{A}$ is shown to be effective in terms of better performance prediction; while combined with other modules (*e.g.*, $\mathcal{T}$ and $\mathcal{I}$) it achieves even better performance. By including all the modules, our ARC model (the *full* version) outperforms by a clear margin the other domain adaptation methods and the baselines. However, the performance gain here is not as remarkable as that in the indoor scenario. We conjecture this is due to several reasons: 1) depth annotation by LiDAR are very sparse while vKitti have annotations everywhere; 2) the Kitti and vKitti images are far less diverse than indoor scenes (*e.g.*, similar perspective structure with vanishing point around the image center).

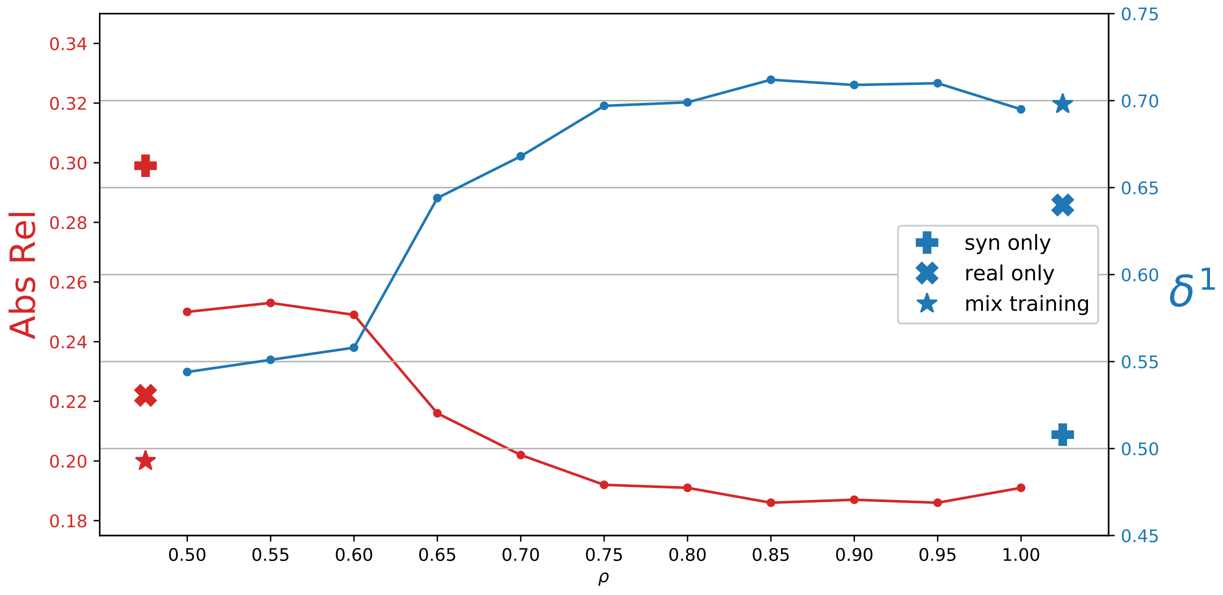


Figure 2.3: Ablation study of the sparsity factor ρ of the ARC model on NYUv2 dataset. We use "Abs-Rel" and "δ^1" to measure the performance (see Table 2.1 for their definition). Note that the sparsity level ρ cannot be exact 1.0 due to KL loss during training, so we present an approximate value with $\rho = 0.99999$.

2.4.4 Ablation Study and Qualitative Visualization

Sparsity level ρ controls the percentage of pixels to remove in the binary attention map. We are interested in studying how the hyperparameter ρ affects the overall depth prediction. We plot in Fig. 2.3 the performance vs. varied ρ on two metrics (Abs-Rel and δ^1). We can see that the overall depth prediction degenerates slowly with smaller ρ at first; then drastically degrades when ρ decreases to 0.8, meaning $\sim 20\%$ pixels are removed for a given image. We also depict our three baselines, showing that over a wide range of ρ. Note that ARC has slightly worse performance when $\rho = 1.0$ (strictly speaking, $\rho = 0.99999$) compared to $\rho = 0.95$. This observation shows that remove a certain amount of pixels indeed helps depth predictions. We also show the curve of how sparsity level ρ affects the performance of ARC on Kitti dataset. As shown in Fig. 2.6, the trend of the curve on Kitti dataset is similar to the curve plotted on NYUv2 dataset, but the slope is quite different. The performance changes very slightly with different sparsity level ρ. Considering the LiDAR depth map is

Table 2.4: **Quantitative comparison** between ARC and mix training baseline inside and outside of the mask region on NYUv2 testing set [214], where Δ represents the performance gain of ARC over mix training baseline under each metric.

Model/metric	↓ lower is better				↑ better		
	Rel	Sq-Rel	RMS	RMS-log	δ^1	δ^2	δ^3
Inside the mask (*e.g.*, removed/inpainted)							
mix training	0.221	0.259	0.870	0.282	0.661	0.889	0.966
ARC: *full*	0.206	0.232	0.851	0.273	0.675	0.895	0.970
Δ	↓0.015	↓0.027	↓0.019	↓0.009	↑0.014	↑0.006	↑0.004
Outside the mask							
mix training	0.198	0.191	0.715	0.256	0.700	0.913	0.976
ARC: *full*	0.185	0.173	0.703	0.249	0.713	0.918	0.977
Δ	↓0.013	↓0.018	↓0.012	↓0.007	↑0.013	↑0.005	↑0.001

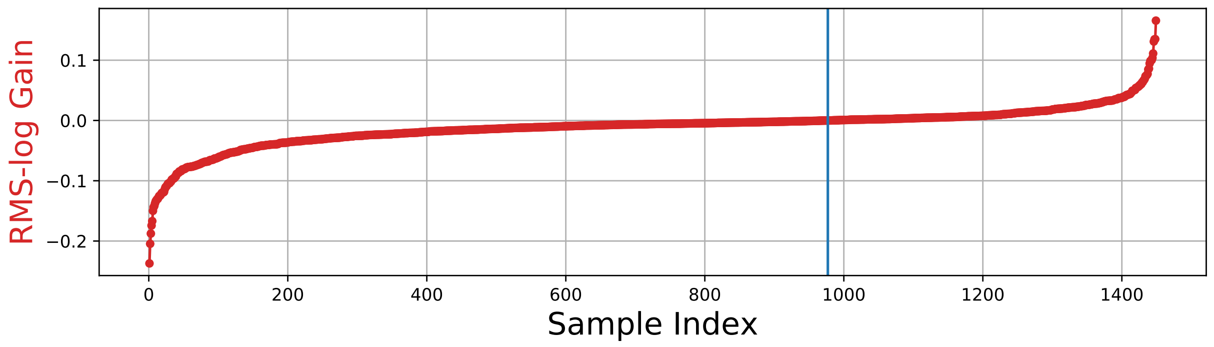


Figure 2.4: Sorted per-sample error reduction of ARC over the mix training baseline on NYUv2 dataset w.r.t. the RMS-log metric. The error reduction is computed as RMS-log(ARC) − RMS-log(mix training). The blue vertical line represents the index separating negative and positive error reduction.

sparse and there is no depth annotation in the sky regions of Kitti images, we believe this behavior matches our expectations. As shown in the previous section, the attention module $\mathcal{A}$ is likely to focus on the sky or pavement. As there is little supervision (no depth in sky regions) and large plain regions (*e.g.*, road), removing pixels from these regions to different extents does not significantly affect the overall depth prediction.

With the comprehensive study of sparsity hyper-parameter ρ from both Fig. 2.3 and Fig. 2.6, we see that the performance drops when the sparsity level ρ increases from 0.95 to 1.0 (strictly speaking, $\rho = 0.99999$[3]). Decrements in performance show that learning to remove a

[3]One cannot set $\rho = 1.0$ exactly due to the KL loss.

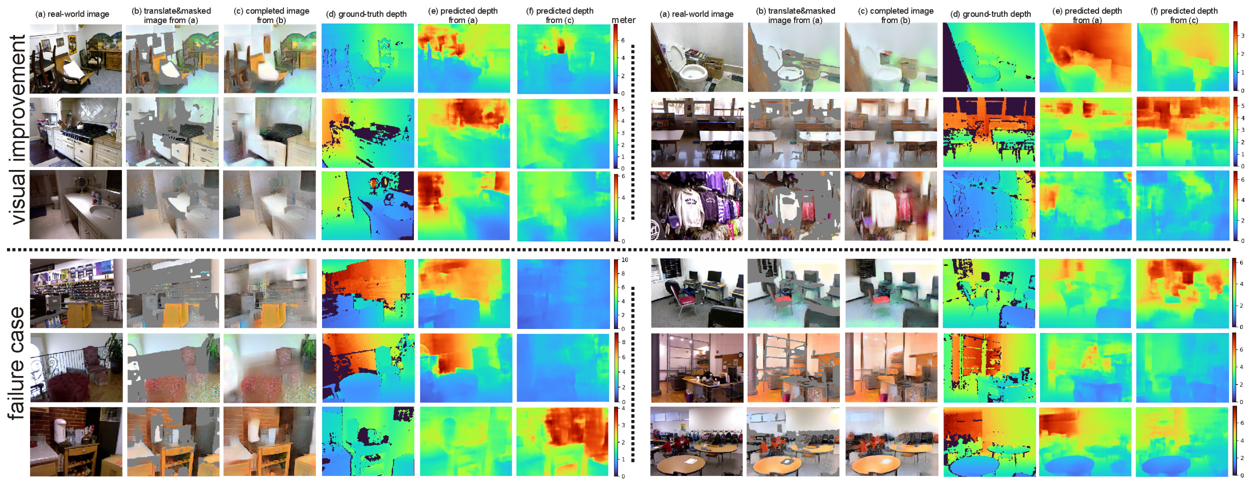

(a) real-world image
(b) translate&masked image from (a)
(c) completed image from (b)
(d) ground-truth depth
(e) predicted depth from (a)
(f) predicted depth from (c)
meter
visual improvement
failure case

Figure 2.5: Qualitative results list some images over which our ARC model improves the depth prediction remarkably, as well as failure cases. (Best viewed in color and zoomed in.)

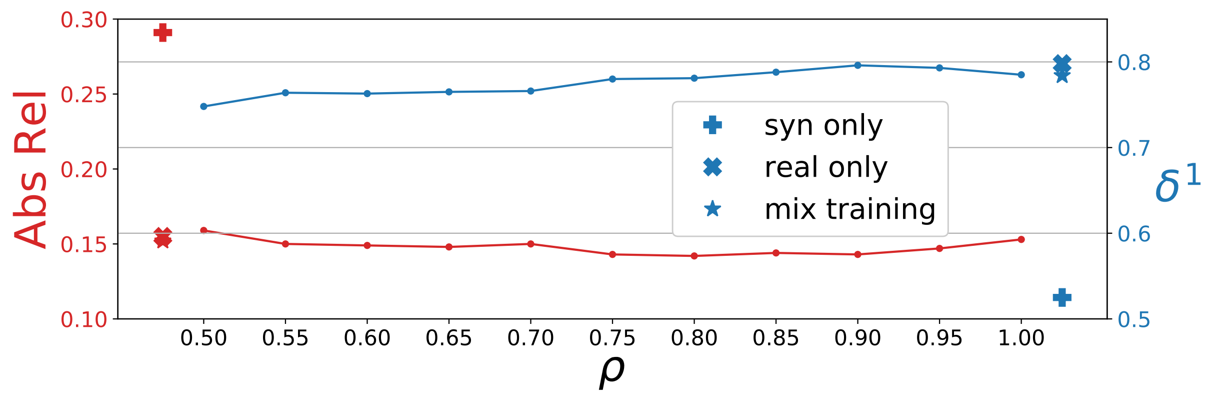


Figure 2.6: Study of the sparsity factor ρ of the ARC model on Kitti dataset.

reasonably portion of pixels, *e.g.*, $\rho = 0.90$ or $\rho = 0.95$, indeed helps improve depth prediction. Note that although $\rho = 1.0$ expects no sparse attention map output from the attention module, we observe training with $\rho = 1.0$ achieves better performance than simply training without attention modules. We believe one possible reason behind this observation is that learning $\mathcal{A}$ before its convergence during training still introduce pixel/region removal, which leads to more robust training as studied in literature [82].

Per-sample improvement study to understand where our ARC model performs better over a strong baseline. We sort and plot the per-image error reduction of ARC over the mix

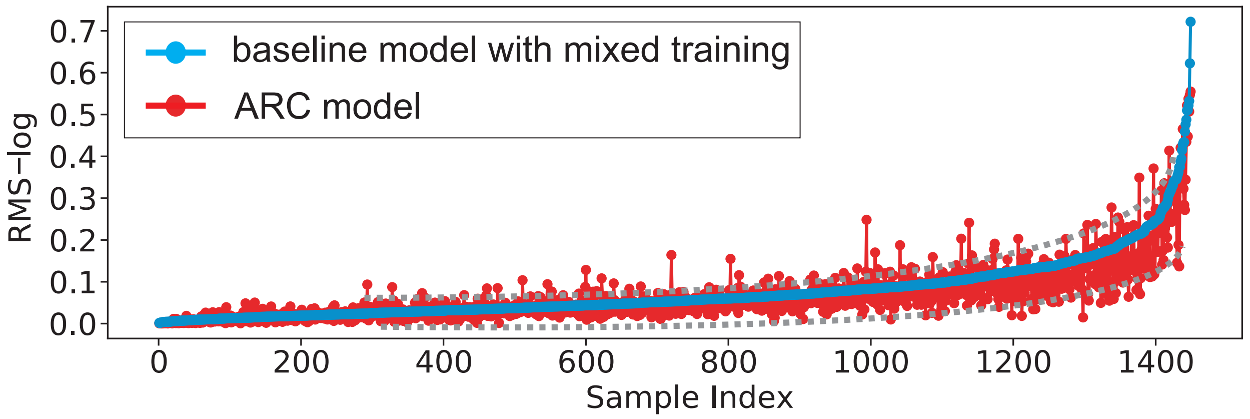


Figure 2.7: Per-sample improvement of ARC and the mix training baseline on NYUv2 testing set w.r.t RMS-log.

training baseline on NYUv2 testing set according to the RMS-log metric, shown in Fig. 2.4. The error reduction is computed as RMS-log(ARC) − RMS-log(mix training). It's easy to observe that ARC reduces the error for around 70% of the entire dataset. More importantly, ARC decreases the error over 0.2 at most when the average error of ARC and our mix training baseline are 0.252 and 0.257, respectively. Additionally, we also compute the per-image prediction of ARC and the mix training baseline on NYUv2 testing set w.r.t RMS-log and plot the result by sorting the baseline performance. As shown in Fig. 2.7, ARC reduces errors over a majority of samples as there are more red dots below the blue curve visually. This observation matches the result shown in Fig. 4 in this chapter, where experimentally proves ARC reduces error for around 70% of the sample in the dataset. More importantly, ARC reduces error even more when the mix training baseline has larger prediction errors, which demonstrates the effectiveness of removing "hard" pixels.

Masked regions study analyzes how ARC differs from mix training baseline on "hard" pixels and regular pixels. For each sample in the NYUv2 testing set, we independently compute the depth prediction performance inside and outside of the attention mask. As shown in Table 2.4, ARC improves the depth prediction not only inside but also outside of the mask regions on average. This observation suggests that ARC improves the depth prediction globally without sacrificing the performance outside of the mask.

Qualitative results of NYUv2 dataset are shown in Fig. 2.5, including some random failure cases (measured by performance drop when using ARC). These images are from NYUv2 dataset. From the good examples, we can see ARC indeed removes some cluttered regions that are intuitively challenging: replacing clutter with simplified contents, *e.g.*, preserving boundaries and replacing bright windows with wall colors. From an uncertainty perspective, removed pixels are places where models are less confident. By comparing with the depth estimate over the original image, ARC's learning strategy of "learn to admit what you don't know" is superior to making confident but often catastrophically poor predictions. It is also interesting to analyze the failure cases. For example, while ARC successfully removes and inpaints rare items like the frames and cluttered books in the shelf, it suffers from over-smooth areas that provide little cues to infer the scene structure. This suggests future research directions, *e.g.* improving modules with the unsupervised real-world images, inserting a high-level understanding of the scene with partial labels (*e.g.* easy or sparse annotations) for tasks in which real annotations are expensive or even impossible (*e.g.*, intrinsics).

Qualitatively Visualizations of Kitti dataset. We also present qualitative results of ARC on Kitti dataset. From Fig. 2.8, we observe that the attention module $\mathcal{A}$ attempts to mask out the sky, pavements, and overexposured areas in both improvements and failure cases. In failed cases, we find a pattern in some images that the sky and pavements are connected (*e.g.*, due to overexposure). Under such condition, the attention module is very likely to remove them together and the original *vanishing point* cannot be reliably inferred, we believe important to estimate the depth in Kitti images. Recall that Kitti images have similar structures as the car is moving forward and the vanishing point is around the image center in most images. It is worth noting that in real training images of Kitti dataset, the depth annotations are very sparse (due to LiDAR sensor) or missing in sky regions. So it is reasonable that the model learns to remove sky regions in a lazy way as there is no penalty from the depth loss on the sky region. Moreover, interestingly, the ARC model learns to paste "green trees" to the removed regions. We conjecture that the green trees are large

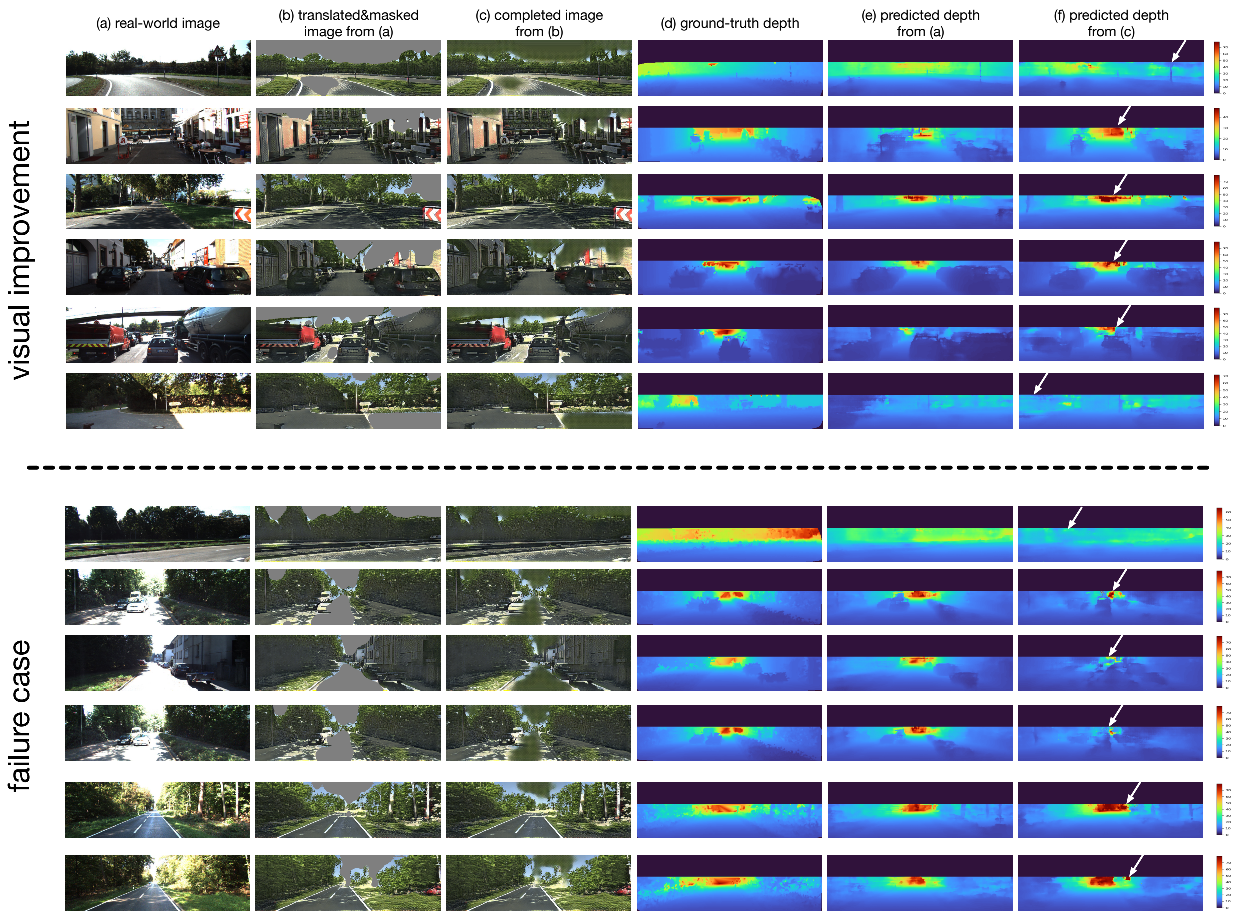

(a) real-world image
(b) translated&masked image from (a)
(c) completed image from (b)
(d) ground-truth depth
(e) predicted depth from (a)
(f) predicted depth from (c)
visual improvement
failure case

Figure 2.8: Qualitative visualizations of our ARC on Kitti dataset including improvements as well as failure cases. White arrows in the last column are used to highlight the regions over which the model improves or degrades visibly w.r.t depth prediction. We use the same color bar for the visualizing depth in each row. (Best view in color and zoomed in.)

regions besides the road and reliable cues to estimate vanishing point and thus better depth prediction.

2.5 Conclusion and Future Work

In this chapter, we present the ARC framework which learns to attend, remove and complete "hard regions" that depth predictors find not only challenging but detrimental to overall depth prediction performance. ARC learns to carry out completion over these removed regions in order to simplify them and bring them closer to the distribution of the synthetic domain. This real-to-synthetic translation ultimately makes better use of synthetic data in producing an accurate depth estimate. With our proposed modular coordinate descent training protocol, we train our ARC system and demonstrate its effectiveness through extensive experiments: ARC outperforms other state-of-the-art methods in depth prediction, with a limited amount of annotated training data and a large amount of synthetic data. We believe our ARC framework is also applicable of boosting performance on a broad range of other pixel-level prediction tasks, such as surface normals and intrinsic image decomposition, where per-pixel annotations are similarly expensive to collect. Moreover, ARC hints the benefit of uncertainty estimation that requires special attention to the "hard" regions for better prediction.

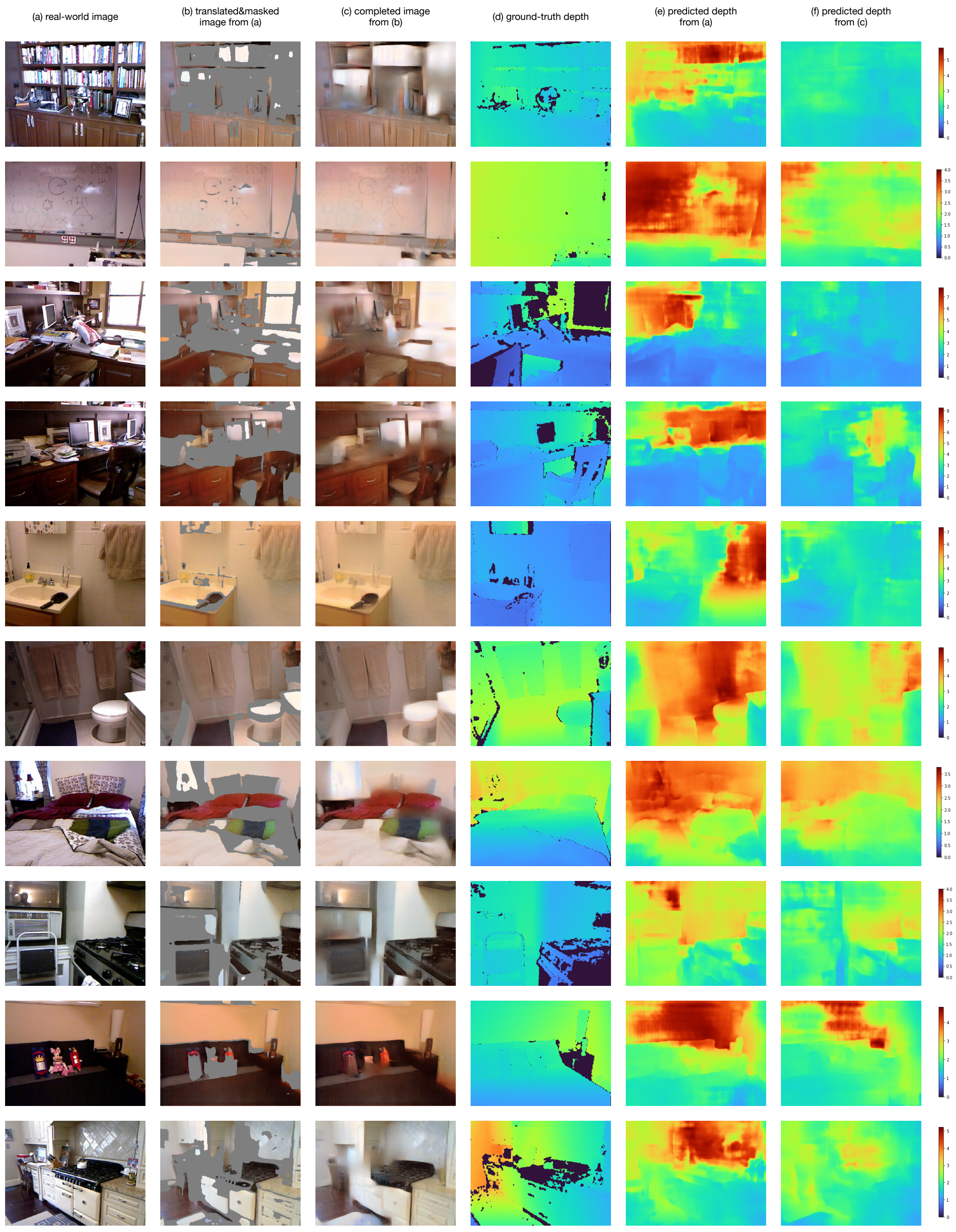

(a) real-world image
(b) translated&masked image from (a)
(c) completed image from (b)
(d) ground-truth depth
(e) predicted depth from (a)
(f) predicted depth from (c)

Figure 2.9: Additional qualitative visualizations of our ARC on NYUv2 dataset. (Best viewed in color and zoomed in.)

Chapter 3

Mitigating camera pose distribution shift in depth predictions

Monocular depth predictors are typically trained on large-scale training sets which are naturally biased w.r.t. the distribution of camera poses. As a result, trained predictors fail to make reliable depth predictions for testing examples captured under uncommon camera poses. To address this issue, we propose two novel techniques that exploit the camera pose during training and prediction. First, we introduce a simple perspective-aware data augmentation that synthesizes new training examples with more diverse views by perturbing the existing ones in a geometrically consistent manner. Second, we propose a conditional model that exploits the per-image camera pose as prior knowledge by encoding it as a part of the input. We show that jointly applying the two methods improves depth prediction on images captured under uncommon and even never-before-seen camera poses. We show that our methods improve performance when applied to a range of different predictor architectures. Lastly, we show that explicitly encoding the camera pose distribution improves the generalization performance of a synthetically trained depth predictor when evaluated on real images.

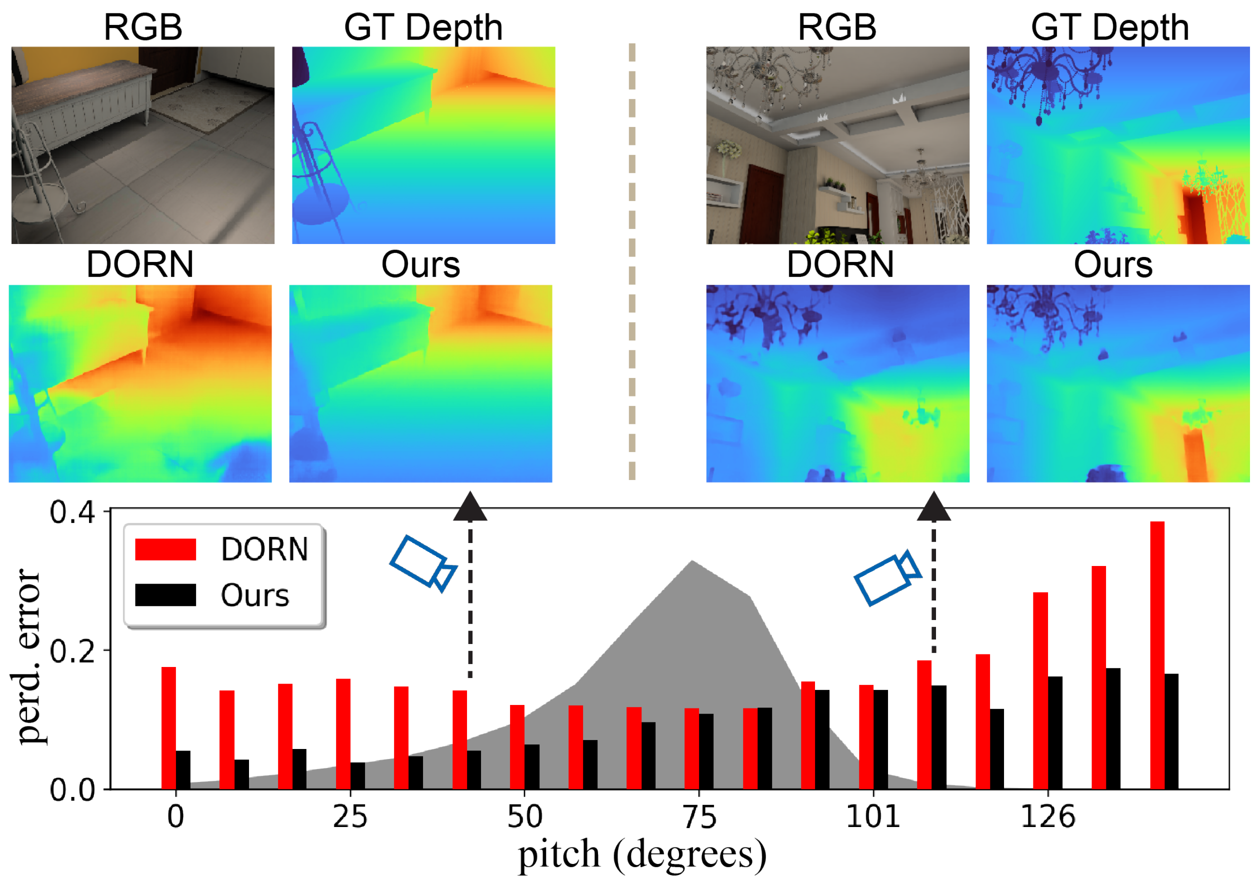


Figure 3.1: Contemporary monocular depth predictors, *e.g.*, DORN [71], rely on large-scale training data which is naturally *biased* w.r.t. the distribution of camera poses (*e.g.*, pitch angle distribution shown in gray). As a result, DORN makes unreliable predictions on test images captured with uncommon poses (red bars), *e.g.*, pitch angles >120°. To address this issue, we propose two novel techniques that drastically reduce prediction errors (*cf*. black bars) by leveraging perspective-aware data augmentation during training and known camera pose at test time. Qualitative examples with more extreme camera pitch angles (top) show that incorporating our techniques leads to notable improvements.

3.1 Background

Monocular depth prediction aims to estimate 3D scene geometry from a 2D image. Despite being a largely underdetermined problem, convolutional neural network (CNN) based depth predictors trained on a sufficiently large-scale dataset are able to learn the joint statistics of scene geometry and appearance, and achieve impressive performance [63, 62, 71, 90, 128, 267].

However, an important overlooked fact is that the distribution of camera poses in training sets are naturally *biased*. As a result, a learned depth predictor is unable to make reliable predictions on images captured from uncommon camera poses, as shown in Fig. 3.1. Importantly, camera poses of testing examples may follow a different distribution from that in the training set. This will exacerbate prediction errors on images that are captured by cameras with uncommon poses relative to the training set.

Contributions. To this end, we propose two novel approaches that significantly improve depth prediction under diverse test-time camera poses. First, we introduce a simple *perspective-aware data augmentation* (PDA) that synthesizes new geometrically consistent training examples with more diverse viewpoints by perturbing the camera pose of existing samples. In contrast, common data augmentation (CDA) methods such as random-crop, though widely adopted in prior work [71, 121, 282, 106, 267, 35], produce training examples where the resulting image and target depth are inconsistent with the perspective geometry (Fig. 3.2). Second, we propose training a conditional depth predictor which utilizes the camera pose (*e.g.*, acquired from IMU or other pose predictors) as a prior (CPP) when estimating depth. We propose an effective approach to encode CPP as an additional channel alongside the RGB input. We find incorporating pose using CPP yields more accurate depth predictors that generalize much better under diverse test-time camera poses.

Through extensive experiments, we show that these techniques significantly improve depth prediction on images captured from uncommon and even never-before-seen camera poses. Both techniques are general and broadly applicable to *any* network architecture. We show that incorporating them in recent state-of-the-art architectures improves their performance further. Lastly, we show that explicitly handling the biased camera pose distribution can improve the performance of a synthetically trained depth predictor when tested on real images. This highlights the importance that camera pose distribution plays in domain adaptation for 3D geometric prediction tasks.

3.2 Related Work

Monocular Depth Prediction and scene layout estimation have been greatly advanced since the seminal works [105, 202]. State-of-the-art approaches train increasingly sophisticated CNN-based predictors [148, 63, 128], utilize better training losses [71, 213, 267] and train on larger-scale training datasets [145, 130, 281, 288].

Surprisingly little attention has been paid to camera pose bias and out-of-distribution generalization. The recent study of [52] concluded that the learned depth predictors have a strong implicit bias causing them to perform poorly on test images when captured from differing camera poses. Our work systematically analyzes this pose bias/robustness in detail and offers technical contributions that improve generalization on test images captured under diverse camera poses.

Camera Pose Estimation plays an essential role in many traditional 3D geometry vision problems such as SLAM [11, 222, 114] and 3D reconstruction [217, 2]. Predicting the relative camera pose between a pair of frames has been widely exploited to perform self-supervised learning of monocular depth prediction [85, 296, 235]. Absolute camera pose is often represented implicitly in predictions of room layout [132, 305, 254, 251]. Closer to our work is [12] which estimates the absolute camera pose (height,pitch,roll) in order to regularize depth predictions in world coordinates.

We explore the benefit of providing the camera pose as an additional *input* to depth prediction. In practical applications, such pose information may come from other sensors (*e.g.*, pitch from IMU) or prior knowledge (*e.g.*, camera mounted at a known height and pitch on an autonomous vehicle). The work of [99, 64] encode camera intrinsics (*e.g.*, focal length) as a part of the input for depth prediction, with a goal to learn a universal depth predictor that generalizes to images captured by different cameras. Similarly, we propose to encode camera extrinsic parameters (*e.g.*, camera height) which we exploit for training better depth predictors

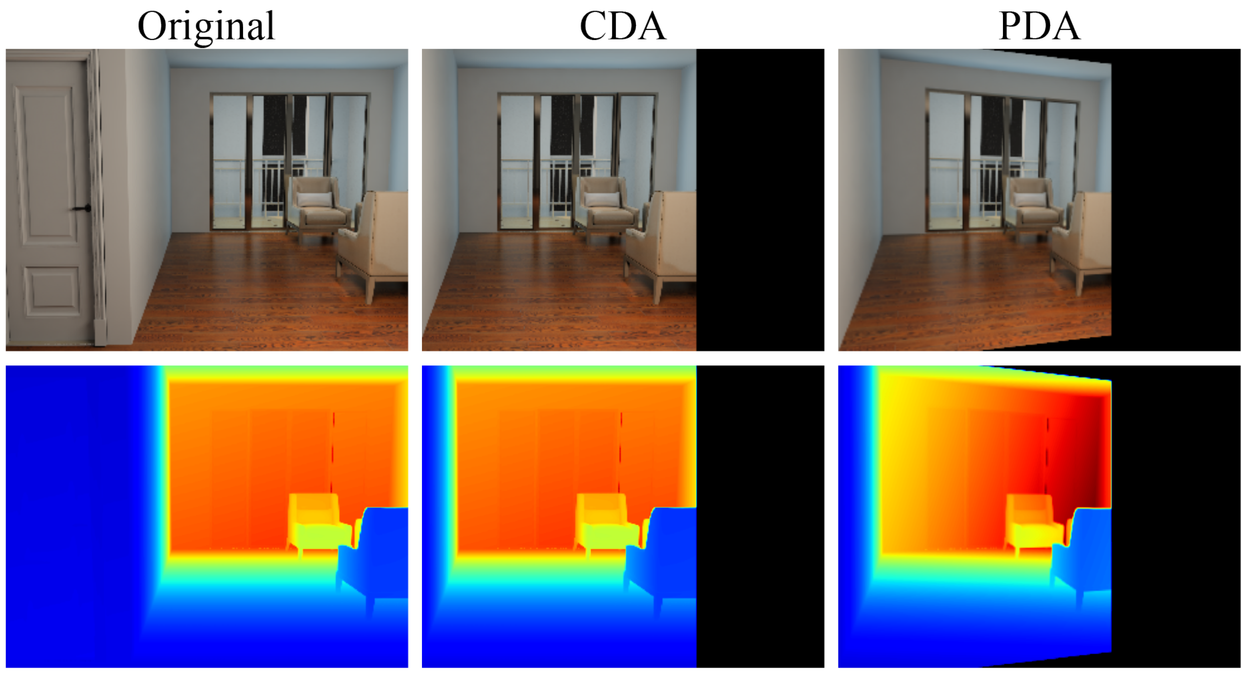


Figure 3.2: Visual comparison of PDA and CDA. From the original example (left), conventional data augmentation (CDA) synthesizes a new example (middle) by randomly cropping a sub-region. It ignores camera pose information and will simply copy the depth values w.r.t. the corresponding pixels. In contrast, perspective-aware augmentation (PDA) simulates a rotation of the camera and synthesizes a new training example with geometrically consistent depth values corresponding to the new camera pose (right).

that perform well on testing images captured with diverse camera extrinsic parameters.

Distribution Bias. Challenges of class imbalance and bias have been a widely discussed topic in the literature on classification and recognition [97, 304, 152]. Distribution bias naturally exists in a training set, implying that some examples are underrepresented or few in number w.r.t. some attributes (*e.g.*, class labels). As a result, a trained model is unlikely to perform well on the underrepresented testing inputs. Even worse, testing examples may come from out-of-distribution data [100, 146, 134, 290], meaning that the training set does not have similar examples. For example, in monocular depth prediction, training examples might be collected with cameras held vertical with minimal pitch and roll variations, but the testing scenario (*e.g.*, AR/VR headset) might have potentially large variations in pitch and roll.

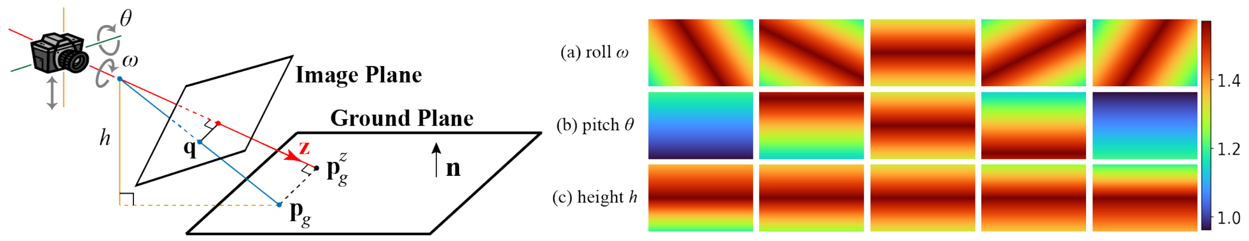


Figure 3.3: **Left:** We illustrate the proposed CPP which encodes camera pose (ω, θ, h) as a 2D image. Intuitively, for a spatial coordinate $\mathbf{q}$ on the image plane (*i.e.*, the encoding map), we find its physical point $\mathbf{p}_g$ on the ground plane, along the ray cast from the camera. Then we compute the pseudo depth value as the length from the camera to $\mathbf{p}_g^z$ which is the projection of $\mathbf{p}_g$ onto the depth direction $\mathbf{z}$ (red line) using Eqn. 3.7. This results in an encoded CPP map for a given camera pose. **Right:** We visualize some encoded CPP maps by varying the ω, θ and h independently.

3.3 Perspective-aware Data Augmentation

Due to the biased distribution of camera poses in the training set, some examples are underrepresented w.r.t. camera poses. To resolve this issue, we would like to augment the training data to cover these underrepresented poses.

Resampling (RS) the training data with replacement to enlarge the prevalence of uncommon poses in the train-set is perhaps the simplest approach. However, this does not increase the diversity of the train-set because it cannot generate new training examples. Perhaps even worse, in practice, training on repeated examples from uncommon camera poses forces the model to weight them more (*cf*. overfitting), while sacrificing performance on other examples captured under common camera poses.

Conventional data augmentation (CDA), specifically random-cropping of the training examples, is widely used to enrich the training data in prior work [63, 127, 71, 121, 267].

While CDA seems to increase the diversity of the training set, its generated data could adversely affect depth prediction. Cropping a subregion from an original example naively copies depth values without considering the view-dependent nature of depth maps that depend on both the camera pose and scene geometry. Such a cropped image is equivalent to capturing another image of the same scene using a camera with an off-center principle point (Fig. 3.2) and can make it difficult to train depth predictors [64]. In our work, we find CDA helps only if we crop sufficiently large regions, which presumably reduces this effect.

Perspective-aware data augmentation (PDA) is our solution that augments training examples consisting of RGB images and depth maps. Given a training example, PDA first perturbs the camera pose, re-projects all pixels to a new image, and recomputes the depth values for the new image using the new camera pose and the original depth map. Despite its simplicity, to the best of our knowledge, *no prior work exploits this idea for data augmentation during the training of a monocular depth predictor*.

Given the training image $\mathbf{I}$, depth $\mathbf{D}$ and camera intrinsic matrix $\mathbf{K}$, we would like to synthesize a new image $\mathbf{I}_s$ and depth $\mathbf{D}_s$ corresponding to a perturbed viewing direction. Let $\mathbf{T}_{rel}$ be the relative rotation between the two poses. Then for any point $\mathbf{q} = [u\ v]$ on $\mathbf{I}$, we compute the corresponding point $\mathbf{q}'$ on the new image $\mathbf{I}_s$ via the homography:

$$\mathbf{q}'_h \sim \mathbf{K}\mathbf{T}_{rel} z \mathbf{K}^{-1}\mathbf{q}_h, \tag{3.1}$$

where z is the corresponding depth value of the point $\mathbf{q}$ from the original depth map $\mathbf{D}$; $\mathbf{q}_h$ and $\mathbf{q}'_h$ are the homogeneous coordinates of points $\mathbf{q}$ and $\mathbf{q}'$, respectively.

Similarly, we compute the depth value for each pixel in the new depth map $\mathbf{D}_s(q')$:

$$z' = \mathbf{v}_{proj}^{\mathrm{T}} \mathbf{T}_{rel} z \mathbf{K}^{-1}\mathbf{q}_h, \tag{3.2}$$

where $\mathbf{v}_{proj} = [0, 0, 1]^{\mathrm{T}}$ is a unit vector pointed along the z-axis of the camera.

While the above demonstrates the computation of per-pixel depth values, in practice, we can compute the whole depth map $\mathbf{D}_s$ efficiently by using backward-warping and standard bilinear interpolation. For larger rotations, we note that the synthesized views will have void regions on the boundary, as shown Fig. 3.2. This does not pose a problem for depth since we simply exclude those regions from the loss during training. However, we find it helpful to pad the RGB void regions using values from the RGB images (using the “reflection” mode during warping).

We also considered applying PDA with camera translation. However, this requires forward-warping and introduces “holes” in the synthesized result [166] at disoccluion boundaries. Handling disocclusions is still an open problem and out of our scope [27, 179, 159], therefore, we choose to only augment camera rotations to avoid disocclusion artifacts and allow for efficient computation.

3.4 Depth Prediction with Camera Pose Prior

Depth maps are *view-dependent representations* that depend on both the scene geometry and camera pose. The camera pose inherently provides *prior knowledge* about the expected scene depth map. For example, knowing a camera is pointing down to the ground at one-meter height, we should expect a depth map of one-meter height roughly everywhere. Therefore, we are motivated to train a conditional depth predictor on camera pose as prior (CPP).

Camera pose is a six-degree-of-freedom (6DoF) vector that describes translation and rotation in 3D world coordinates. In typical terrestrial man-made scenes, we consider a global coordinate with one axis pointing upwards (as specified by gravitational acceleration) and fix the origin along that axis to be 0 at the ground plane. Since there is no unique origin along the two remaining axes, we assume that our camera pose prior should be uniform over translations parallel to the ground plane. Similarly, there is no unique way to specify the orientation of the axes parallel to the ground plane so our prior should necessarily be uniform over rotations of the camera about the up-axis. This leaves three degrees-of-freedom (3DoF):

the height of the camera above the ground plane h, the pitch (angle relative to the up-axis) of the camera θ, and any roll ω of the camera around its optical axis (Fig. 3.3).

A naive encoding approach. We now consider using this 3DoF camera pose as a part of the input (along with RGB) to depth predictors. To incorporate this as input into a CNN, inspired by the literature [64, 269], we convert 3DoF camera poses into 2D maps, which are concatenated with the RGB image as a whole input to learn the depth predictor. Naively, we can create three more channels of resolution $H \times W$, which copy values of roll ω, pitch θ and height h, *i.e.*, $\mathbf{M}_\omega[:,:] = \omega$, $\mathbf{M}_\theta[:,:] = \theta$ and $\mathbf{M}_h[:,:] = h$, respectively. However, the effect of pose on the depth distribution depends strongly on the position in the image relative to the camera center so translation-equivariant convolutions cannot fully exploit this encoding (except by relying on boundary artifacts). This is supported by an experimental comparison showing this naive encoding is inferior to our proposed *CPP* encoding method, elaborated in the following.

CPP encoding encodes the pose locally by assuming that the camera is placed in an empty indoor scene with an infinite floor and ceiling. Intuitively, it encodes the camera pose by intersecting rays from the camera center with predefined ground/ceiling planes and recording the depth, see Fig. 3.3.

Let $\mathbf{p} = [x\ y\ z]$ be a 3D point in the global coordinate, $\mathbf{q} = [u\ v]$ be a 2D point on the image plane whose homogeneous form is $\mathbf{q}_h$, $\mathbf{n} \in \mathbb{R}^3$ denotes the normal vector of ground planes, C be the distance between two planes in the up direction.

The projection from 3D coordinates to 2D image plane is:

$$\lambda \mathbf{q}_h = \mathbf{K}\mathbf{R}^{-1}(\mathbf{p} - \mathbf{t}), \tag{3.3}$$

where λ is a scale factor; $\mathbf{K}$ is the intrinsic matrix; $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are rotation and translation matrices known from the camera pose, respectively. We compute the 3D point $\mathbf{p}$ where the ray shooting from the camera center through point $\mathbf{q}$ eventually intersects with planes or the horizon. However, λ is an unknown scalar and we need extra constraints to compute it. Taking ground plane as an example, we know the collections of 3D points intersecting with ground plane is $\{\mathbf{p} : \mathbf{n}^{\mathrm{T}}\mathbf{p} = 0\}$. With this new constraint, we rearrange Eqn. 3.3 and multiply $\mathbf{n}^{\mathrm{T}}$ on both sides of the equation:

$$\mathbf{n}^{\mathrm{T}}\mathbf{p} = \lambda\mathbf{n}^{\mathrm{T}}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h + \mathbf{n}^{\mathrm{T}}\mathbf{t} \;\Rightarrow\; \lambda = \frac{-\mathbf{n}^{\mathrm{T}}\mathbf{t}}{\mathbf{n}^{\mathrm{T}}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h} \tag{3.4}$$

Now, we plug the computed λ back to Eqn. 3.3, then the 3D point $\mathbf{p}_g$ that intersects the ground plane for $\mathbf{q}$ is:

$$\mathbf{p}_g = \frac{-\mathbf{n}^{\mathrm{T}}\mathbf{t}}{\mathbf{n}^{\mathrm{T}}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h + \mathbf{t} \tag{3.5}$$

Similarly, with the constraint $\{\mathbf{p} : \mathbf{n}^{\mathrm{T}}\mathbf{p} = C\}$, the 3D point $\mathbf{p}_c$ that intersects with the ceiling plane for $\mathbf{q}$ is:

$$\mathbf{p}_c = \frac{C - \mathbf{n}^{\mathrm{T}}\mathbf{t}}{\mathbf{n}^{\mathrm{T}}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h}\mathbf{R}\mathbf{K}^{-1}\mathbf{q}_h + \mathbf{t} \tag{3.6}$$

Once we have the 3D intersection point $\mathbf{p}$ for each point $\mathbf{q}$ on the image plane, we compute the projection on the camera $\mathbf{z}$ direction (*i.e.*, the depth direction):

$$z(\mathbf{p}) = \mathbf{v}_{proj}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{p} - \mathbf{t}), \tag{3.7}$$

where $\mathbf{v}_{proj}$ is the projection vector that computes the projection of a 3D point along the camera depth direction. Finally, for each point $[u, v]$ in the encoding map $\mathbf{M} \in \mathbb{R}^{H\times W}$, we compute both $\mathbf{p}_g$ and $\mathbf{p}_c$ and take the maximum (positive) value as the pseudo depth value

(Fig. 3.3):

$$\mathbf{M}[u,v] = \max\{z(\mathbf{p}_g), z(\mathbf{p}_c)\} \tag{3.8}$$

The encoding map $\mathbf{M}$ can have infinite values, *e.g.*, when the ray shooting from the camera is parallel to the ground plane. To map values into a finite range, we apply the inverse tangent operator $\tan^{-1}(\cdot)$ to obtain our final encoding $\mathbf{M}_{CPP} = \tan^{-1}(\mathbf{M})$ which takes on values in the range $[\tan^{-1}(\min\{h, C-h\}), \frac{\pi}{2}]$. We visualize some CPP maps in Fig. 3.3-right.

To train a conditional depth prediction, we simply concatenate the CPP encoded map with the corresponding RGB as a four-channel input. This implies that our CPP encoding approach applies to *any* network architectures with a simple modification on the first convolution layer.

3.5 Experiments

We validate our methods through extensive experiments. Specifically, we aim to answer the following questions:[1]

- Do our methods improve depth prediction on a test-set that has a different camera pose distribution from the train-set?
- Does resampling improve depth prediction on a test-set that has a different camera pose distribution?
- Do our methods improve depth estimation on testing images captured with out-of-distribution camera poses?
- Does applying our methods to other state-of-the-art depth predictors improve their performance?
- Do our methods improve the performance of a depth predictor when training and

[1]Answers: yes, no, yes, yes, yes.

testing on different datasets?

Datasets. In our work, we use two publicly available large-scale indoor-scene datasets, InteriorNet [141] and ScanNet [43] which come with ground-truth depth and camera pose. Compared to other datasets such as [80, 40, 219], these were selected because they illustrate a much wider variety of camera poses. InteriorNet [141] consists of photo-realistic synthetic video sequences with randomly generated camera trajectories with a wide variety of pitch and height but minimal roll. ScanNet [43] contains real-world RGBD videos of millions of views from 1,513 indoor scenes collected with substantial variation in pitch, roll, and height. For each dataset, we create train/test sets by randomly splitting *scenes* into two disjoint sets with 85%/15% examples, respectively. We use a stratified sampling approach to avoid picking adjacent video sequences in the same set.

Sampling. To explore how the biased camera pose distribution affects depth learning and prediction, we sample three subsets from the test/train set (Fig. 3.4) which each have 10k/1k images but with different distributions of camera poses.

- *Natural* selects samples at random to reflect the natural distribution of the dataset.
- *Uniform* selects samples that simulate a uniform distribution over poses with priority: pitch$\rightarrow$roll$\rightarrow$height. Concretely, we quantize the range of camera pitch/roll/ height into equal-size bins, and sample an approximately equal number of examples in each bin.
- *Restricted* samples images within a narrow range: camera pitch $\theta \in [85°, 95°]$ and height $h \in [1.45, 1.55]$ (meters). While InteriorNet does not have roll variations, ScanNet does: roll $\omega \in [-5°, 5°]$. We create *Restricted* to particularly study how depth predictor performs on testing images captured with out-of-distribution camera poses.

Evaluation Metrics. There are several evaluation metrics widely used in the literature [63, 71, 267], including absolute relative difference (Abs^r), squared relative difference (Sq^r), root

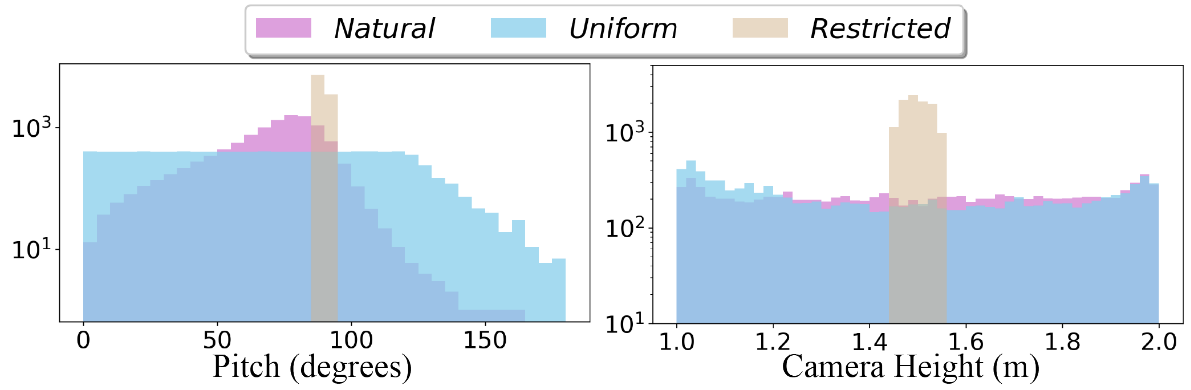


Figure 3.4: Distribution of camera pitch and heights for three subsets of images from InteriorNet. From the *Natural* subsect, we observe the dataset of InteriorNet does have a naturally biased distribution (esp. pitch). Please refer to the text on how we construct the three subsets.

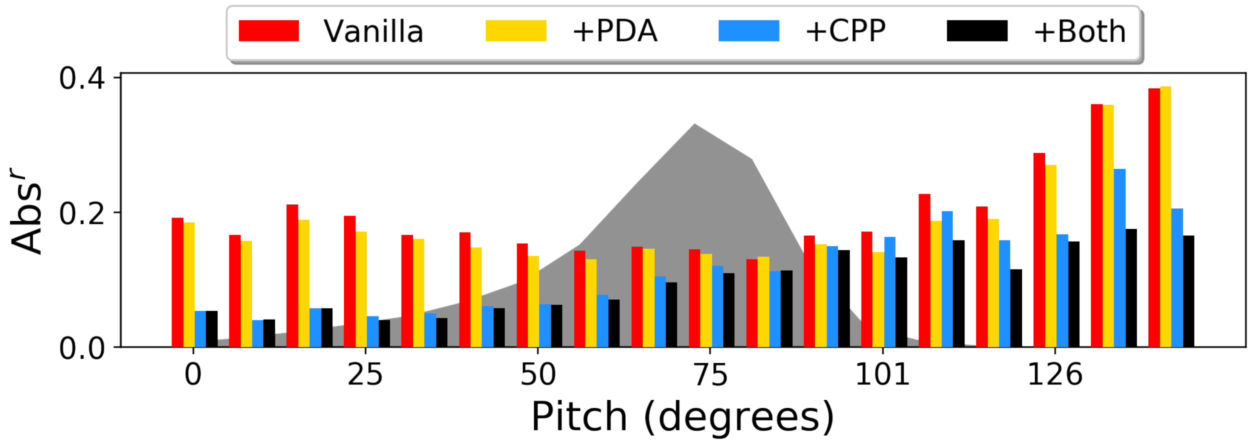


Figure 3.5: Breakdown analysis of depth prediction w.r.t. pitch. The background shade denotes the camera pose distribution w.r.t. pitch. Clearly, both PDA and CPP improve depth prediction in underrepresented camera poses. Surprisingly, CPP remarkably boosts depth prediction, while applying both CPP and PDA achieves the best performance "everywhere".

mean squared log error (RMS-log), and accuracy with a relative error threshold of $\delta^k < 1.25^k$, $i = 1, 2$.

Implementation. We use a standard UNet structured model to perform most of our experimental analysis [288, 284, 291]. We also demonstrate that our techniques apply to other depth predictor architectures (Section 3.5.3). Unless specified, all models are trained with the L1 loss and applied random left-right flip augmentation during training.

While we initially learn the conditional depth predictor using the *true* camera pose, we also

Table 3.1: **Within & cross-distribution evaluation**. In each dataset, we train depth predictors on their *Natural* train-sets and evaluate on both *Natural* and *Uniform* test-sets. We apply different methods to a Vanilla model. All models use the same network architecture. Vanilla performs poorly in cross-distribution evaluation (*cf*.*Natural*-test vs. *Uniform*-test), demonstrating that the biased camera pose distribution affects the training of depth predictors. As expected, RS hurts depth prediction compared to Vanilla. In contrast, CPP and PDA show better performance; jointly applying them performs the best (*i.e.*, "Both"). Finally, comparing alternative methods to our CPP (vs. Native) and PDA (vs. RS and CDA) shows the merits of our methods.

Models	*Natural-Test-Set*		*Uniform-Test-Set*	
	↓ `better`	↑ `better`	↓ `better`	↑ `better`
	Abs^r/Sq^r/RMS-log	δ^1 / δ^2	Abs^r/Sq^r/RMS-log	δ^1 / δ^2
	`InteriorNet`			
Vanilla	.154 / .148 / .229	.803 / .945	.183 / .146 / .250	.724 / .926
+ RS	.192 / .203 / .267	.726 / .918	.210 / .174 / .272	.661 / .906
+ CDA	.142 / .137 / .222	.825 / .950	.172 / .125 / .238	.738 / .934
+ PDA	.138 / .123 / .207	.834 / .957	.168 / .122 / .229	.757 / .942
+ Naive	.132 / .145 / .219	.835 / .944	.137 / .116 / .213	.810 / .944
+ CPP	.108 / .120 / .199	.872 / .958	.106 / .088 / .183	.876 / .961
+ Both	**.095 / .101 / .180**	**.898 / .966**	**.091 / .069 / .161**	**.903 / .973**
	`ScanNet`			
Vanilla	.125 / .068 / .186	.837 / .962	.177 / .121 / .265	.711 / .928
+ RS	.216 / .158 / .279	.619 / .889	.218 / .168 / .300	.630 / .881
+ CDA	.116 / .062 / .179	.853 / .964	.174 / .121 / .264	.727 / .922
+ PDA	.115 / .059 / .171	.860 / .970	.166 / .110 / .248	.752 / .938
+ Naive	.120 / .069 / .184	.846 / .959	.173 / .127 / .255	.755 / .923
+ CPP	.108 / .060 / .171	.871 / .965	.154 / .106 / .239	.781 / .943
+ Both	**.102 / .052 / .160**	**.882 / .973**	**.143 / .097 / .230**	**.809 / .952**

tested encoding a *predicted* camera pose in Section 3.5.4. For predicting camera pose, we train a pose predictor with the ResNet18 architecture [98] to directly regresses to camera pitch, roll, and height. We resize all images and depth maps to 240×320, and adjust camera intrinsic parameters (for CPP encoding) accordingly. We use PyTorch [180] to train all the models for 200 epochs on a single Titan Xp GPU. We use the Adam optimizer [117] with a learning rate 1e-3, and coefficients 0.5 and 0.999 for computing the running averages of gradient and its square.

For CPP encoding, we set the ceiling height $C = 3$ meters. We have studied different settings of C, but find little difference. For PDA, we randomly perturb camera pose within $[-0.1, 0.1]$ radius angle jointly w.r.t. pitch, roll, and yaw; we also ablate the perturbation scale in

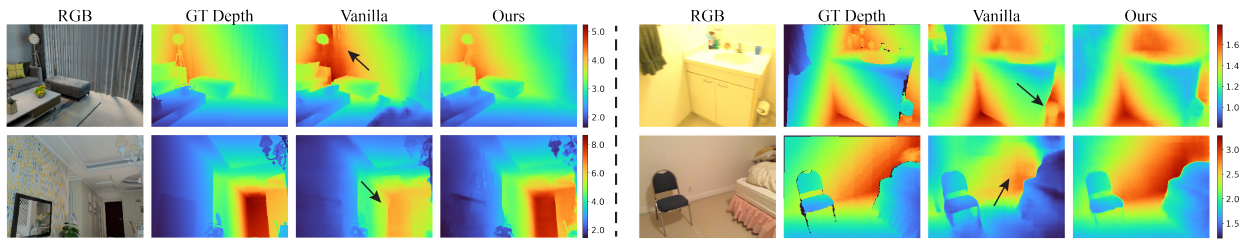


Figure 3.6: Qualitative comparison between Vanilla and our method (using both CPP and PDA) on random testing images from InteriorNet (left) and ScanNet (right). Depth maps for each example are shown with the same colorbar range. Notably, the images are captured under some uncommon camera angles relative to the training pose distribution (Fig. 3.5). The Vanilla model seems to make erroneous predictions w.r.t. the overall scale of the depth. In contrast, by applying CPP and PDA, the new model ("ours") produces visually improved results.

Section 3.5.5.

3.5.1 Within & Cross-Distribution Evaluation

We start with initial experiments designed to reveal how bias in camera pose statistics affects depth predictor performance (Fig. 3.4). The experiments also explore the design choices of our methods and validate their effectiveness. Specifically, we train depth predictors on the *Natural* train-sets, and test them on both *Natural* and *Uniform* test-sets. We apply a sequence of different modifications to a "Vanilla" baseline model, *i.e.*, a UNet-based predictor. Table 3.1 lists detailed comparisons, and Fig. 3.6 shows qualitative comparisons.

The Vanilla model degrades notably when evaluated on a test-set that has a different pose distribution, *i.e.*, from *Natural* to *Uniform*, showing the clear influence from the biased distribution of camera poses. In terms of data augmentation, simply resampling the training data (RS) hurts performance (*cf*. Vanilla and "+RS"). Moreover, our PDA outperforms CDA, demonstrating the importance of synthesizing geometric-aware training examples using the corresponding camera and the scene geometry in depth prediction. As for camera pose encoding, our CPP encoding method clearly outperforms the Naive method, Importantly, jointly applying CPP and PDA performs the best on both within-distribution (*Natural*) and cross-distribution (*Uniform*) test-sets. Finally, we breakdown the performance in Fig. 3.5 to analyze when (*i.e.*, w.r.t. pitch angle) PDA and CPP improve depth prediction. Generally, both of them help reduce prediction errors on testing images captured with underrepresented camera poses, while CPP yields the largest benefits. Applying both achieves the best performance "*everywhere*".

3.5.2 Out-of-Distribution Evaluation

We now study how our methods help when training depth predictors on a train-set which have a rather restricted range of camera poses. Specifically, we train models on the *Restricted* train-sets of InteriorNet and ScanNet, respectively, and test the models on their *Natural* test-sets. This setup is synthetic and unlikely to be a real-world scenario, but it allows for exclusive analysis of depth predictors when tested on images captured with out-of-distribution or never-before-seen camera poses.

Table 3.2 lists detailed comparisons. Vanilla model performs poorly when tested on *Natural* test-set. CPP slightly improves prediction, but PDA helps a lot. CDA (*i.e.*, random-crop as augmentation) also helps, presumably owing to more diverse training examples, but underperforms our PDA. This further confirms the importance of generating geometric-aware new training examples for the given scene and camera in depth prediction. Under expectation, applying both PDA and CPP performs the best.

3.5.3 Applicability to Other Predictor Networks

CPP and PDA are general and applicable to different model architectures. We study applying them to training two well-established depth predictors: DORN [71] and VNL [267]. Compared to the Vanilla model, both predictors adopt different network architectures [255, 98] and

Table 3.2: **Out-of-distribution evaluation**. We train depth predictors on the *Restricted* train-sets and test on both *Restricted* and *Natural* test-sets of each datasets. Vanilla model performs poorly on *Natural* test-sets, clearly showing the challenge of depth prediction on images captured under novel/never-before-seen camera poses. CPP slightly improves performance, but PDA helps more. CDA also improves performance presumably because it synthesizes more training examples, but underperforms PDA. As expected, jointly applying both CPP and PDA achieves the best performance on both *Restricted* and *Natural* test-sets.

Models	*Restricted-Test-Set*		*Natural-Test-Set*	
	↓ `better` Abs^r/Sq^r/RMS-log	↑ `better` δ^1 / δ^2	↓ `better` Abs^r/Sq^r/RMS-log	↑ `better` δ^1 / δ^2
		`InteriorNet`		
Vanilla	.150 / .234 / .296	.819 / .916	.265 / .368 / .412	.538 / .767
+ CPP	.149 / .237 / .301	.820 / .915	.264 / .350 / .397	.548 / .782
+ CDA	.139 / .228 / .286	.830 / .922	.227 / .279 / .336	.637 / .845
+ PDA	.136 / .178 / .239	.833 / .935	.237 / .266 / .295	.652 / .878
+ Both	**.131 / .170 / .237**	**.839 / .936**	**.216 / .231 / .286**	**.666 / .894**
		`ScanNet`		
Vanilla	.177 / .131 / .259	.705 / .897	.317 / .304 / .390	.444 / .748
+ CPP	.169 / .125 / .258	.726 / .897	.301 / .283 / .384	.464 / .756
+ CDA	.163 / .121 / .259	.724 / .904	.289 / .266 / .371	.467 / .771
+ PDA	.160 / .112 / .244	.729 / .914	.283 / .251 / .353	.493 / .795
+ Both	**.155 / .108 / .228**	**.731 / .918**	**.277 / .245 / .348**	**.504 / .804**

different loss functions. They convert continuous depth values to discrete bins and model depth prediction as a classification problem. Moreover, VNL incorporates a virtual surface normal loss which provides a geometric-aware regularization during training. We implement DORN and VNL using publicly-available third-party code. We train and test all models on the *Natural* train/test-sets, in InteriorNet and ScanNet, respectively.

Table 3.3 lists detailed results, which are comparable to the *Natural-Test-Set* column in Table 3.1. Consistent with previous experiments, both CPP and PDA improve the performance further based on DORN and VNL. As our CPP and PDA improve other depth predictors as a general approach, we suggest using them in future research of monocular depth estimation.

Table 3.3: **Applicability to other predictor architectures**. In each dataset, we train state-of-the-art depth predictors (DORN [71] and VNL [267]) by optionally applying our CPP and PDA approaches. All models are trained/tested on the *Natural* train/test-sets per dataset. Clearly, both CPP and PDA boost their performance.

Models	InteriorNet		ScanNet	
	↓ better	↑ better	↓ better	↑ better
	Abs^r/Sq^r/RMS-log	δ^1 / δ^2	Abs^r/Sq^r/RMS-log	δ^1 / δ^2
DORN	.128 / .126 / .218	.854 / .957	.112 / .065 / .197	.856 / .959
+ CPP	.098 / .105 / .197	.899 / .962	.108 / .062 / .191	.875 / .960
+ PDA	.115 / .112 / .207	.867 / .958	.110 / .068 / .195	.857 / .963
+ Both	**.085 / .088 / .124**	**.912 / .971**	**.093 / .049 / .153**	**.903 / .979**
VNL	.131 / .140 / .235	.853 / .954	.115 / .069 / .205	.855 / .960
+ CPP	.101 / .120 / .207	.893 / .960	.112 / .064 / .195	.871 / .961
+ PDA	.117 / .115 / .210	.861 / .959	.110 / .065 / .199	.855 / .962
+ Both	**.086 / .085 / .131**	**.909 / .970**	**.095 / .051 / .157**	**.901 / .980**

3.5.4 Synthetic-to-Real Generalization

Previous experiments have validated that addressing the biased camera pose distribution helps train a depth predictor that works better on another test-time distribution, or more generally another domain. Here we evaluate performance in the presence of substantial synthetic-to-real domain shift which includes both low-level shifts (*e.g.*, local material appearance) and high-level shifts (novel objects, layouts and camera poses) [288]. Specifically, we synthetically train a depth predictor (on InteriorNet *Natural* train-set) and test it on real images (ScanNet *Natural* and *Uniform* test-sets). We also consider a more practical scenario that one does not have access to the true camera pose but instead must rely on the predicted poses. To this end, we train a camera pose predictor on ScanNet *Natural* test-set to predict camera pitch, height and roll for a given RGB image. Then, we perform CPP encoding with the predictive pose, *i.e.*, CPP$_{pred}$

Table 3.4 lists detailed setups and results; we summarize the salient conclusions. Vanilla achieves worse performance compared to Table 3.1, showing a clear domain gap between the two datasets. Applying CDA hurts the performance, presumably because the generated training data by CDA disobey the projective geometry relations between scene geometry and

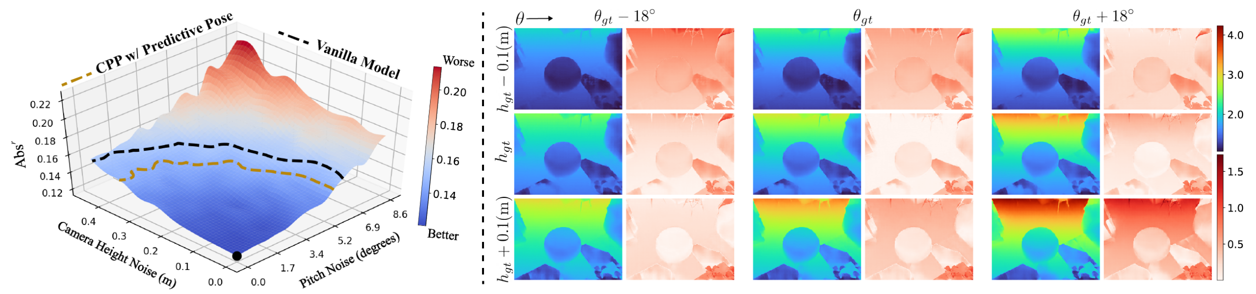


Figure 3.7: **Left:** We plot depth prediction error (Abs^r) w.r.t. different levels of noise in camera height and pitch. We apply CPP to train/test a depth predictor (based on Vanilla) on InteriorNet *Natural* train/test-set. For a given noise level δ, the trained model makes depth predictions using a CPP map computed with a perturbed camera pose, *e.g.*, the pitch is sampled from $\theta_{gt}+\mathbf{U}[-\delta,\delta]$. The black dot at the origin stands for the (best) performance using the true camera pose (*i.e.*, no noises are presented in pitch and height). The dashed lines represent the average performance levels for the Vanilla and CPP with predictive poses. **Right**: We visualize depth prediction by CPP model when encoding perturbed camera pitch angles $\theta_{gt}\pm 18^\circ$ and heights $h_{gt}\pm 0.1$(m). CPP model predicts shallower depth when both pitch and camera height decrease (*i.e.*, camera is tilted down or translated closer to the floor). This qualitatively confirms that the camera pose prior induces a meaningful shift in the estimator. The corresponding RGB image and ground-truth depth appear in Fig. 3.8.

camera model and pose (*cf*. Section 3.3). In contrast, our PDA helps, but CPP improves even more notably. Applying CPP with the predictive pose (CPP$_{pred}$) achieves a remarkable performance boost over PDA, suggesting that using predictive poses, or more generally exploiting camera poses during training, is quite valuable in depth prediction. We analyze in the next section how the model is resilient to the errors in predicted poses. Lastly, using the true camera pose in CPP performs the best.

3.5.5 Further Discussion and Ablation Study

"Blind" depth prediction without RGB. To characterize the prior knowledge carried by camera poses in terms of depth prediction, we train a "blind predictor" on the InteriorNet *Natural* train-set, taking as input only the CPP encoded maps of camera poses *without* RGB images. For comparison, we compute an average depth map over the whole *Natural* train-set. We qualitatively compare results on two testing examples in Fig. 3.8. Visually,

Table 3.4: **Mitigating distribution bias of camera poses improves synthetic-to-real domain adaptation**. We train depth predictors synthetically on InteriorNet (*Natural* train-set) and test them on real-world images from ScanNet *Natural* and *Uniform* test-sets. This is a typical setup for synthetic-to-real domain adaptation in the context of depth prediction. Interestingly, we find that CDA hurts the performance, presumably because the generated training examples by CDA do not obey the relations among camera model, scene geometry and camera pose, and hence do not necessarily help training a generalizable depth predictor. In contrast, our PDA helps synthetic-to-real generalization and applying CPP improves further. Importantly, applying CPP with predictive poses (CPP$_{pred}$) achieves a remarkable performance boost, whereas using the true camera pose in CPP performs the best.

Methods	*Natural-Test-Set*		*Uniform-Test-Set*	
	↓ `better`	↑ `better`	↓ `better`	↑ `better`
	Abs^r/Sq^r/RMS-log	δ^1 / δ^2	Abs^r/Sq^r/RMS-log	δ^1 / δ^2
Vanilla	.242 / .204 / .315	.570 / .852	.305 / .259 / .364	.457 / .797
CDA	.246 / .205 / .321	.568 / .843	.316 / .288 / .391	.433 / .771
PDA	.239 / .198 / .311	.575 / .857	.298 / .252 / .359	.461 / .804
PDA+CPP$_{pred}$	.219 / .190 / .305	.586 / .866	.273 / .231 / .346	.548 / .835
PDA+CPP	**.208 / .170 / .282**	**.677 / .893**	**.245 / .228 / .315**	**.620 / .858**

encoding camera pose alone using CPP reliably provides depth estimate on floor regions. This intuitively explains that using camera pose does serve as strong prior knowledge of scene depth.

Resilience to noisy camera pose. Camera pose estimates (*e.g.*, from IMU sensors) are potentially noisy, and the predicted camera poses are undoubtedly erroneous (*cf*. the previous experiment using predicted poses in CPP). We study how resilient CPP is to test-time errors in the camera pose. Specifically, when testing a trained model with CPP, we randomly perturb the camera poses of all testing examples up to a pre-defined scale, and measure the overall performance as a function of prediction error w.r.t. noise added to the true camera pitch θ_{gt} and height h_{gt}, as shown in Fig. 3.7. We find that CPP outperforms the vanilla model even with significant misspecification of the pose (*e.g.*, height error < 0.3m, pitch error < 5 degrees).

Augmentation scales in PDA. We ablate the augmentation scale in PDA during training depth predictors, as detailed in Fig. 3.9. Perhaps surprisingly, applying a larger scale PDA consistently improves depth prediction until a very large perturbation (*i.e.*, rotating at most

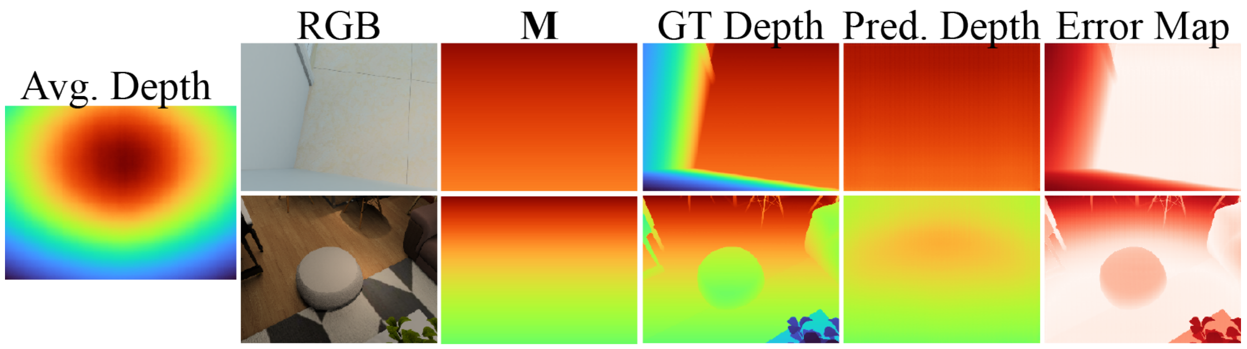


Figure 3.8: Camera pose alone provides a strong depth prior even for "blind" depth prediction. Specifically, over the InteriorNet *Natural* train-set, we train a depth predictor solely on the CPP encoded maps **M** *without* RGB as input. For visual comparison, we compute the average depth map (shown left). We visualize depth predictions on two random examples. All the depth maps are visualized with the same colormap range. Perhaps not surprisingly, **M** presents nearly the true depth in floor areas, suggesting that camera pose alone does provide strong prior depth information for these scenes.

80°), presumably when very large void regions are introduced in the synthesized training examples (Fig. 3.2).

Handling infinite values in CPP encoding. At the last step in computing CPP encoded maps $\mathbf{M}_{CPP}$, we apply the inverse tangent operator to eliminate the infinity values (happens when the ray shooting from camera is parallel to the ground plane) and maps the values of $\mathbf{M}$ (i.e. $\mathbf{M}_{CPP} = \tan^{-1}(\mathbf{M})$) to the range $[\tan^{-1}(\min\{h, C-h\}), \frac{\pi}{2}]$. However, the inverse tangent operator is not the only choice. We provide an ablation study that replaces the $\tan^{-1}(\cdot)$ with a clipping operation.

Specifically, we set a threshold τ that represents the prior knowledge of the distance from camera to the furthest point in the scene. Mathematically, for each point $[u, v]$ in the new CPP clipping encoded map $\mathbf{M}^{Clip}_{CPP} \in \mathbb{R}^{\mathbb{H}\times\mathbb{W}}$, we compute the pseudo depth value:

$$\mathbf{M}^{Clip}_{CPP}[u,v] = \begin{cases} M[u,v] & M[u,v] < \tau \\ \tau & \text{otherwise} \end{cases}$$

We set $\tau = 20.0$ in this experiment. After clipping, we linearly rescale the encoded map to

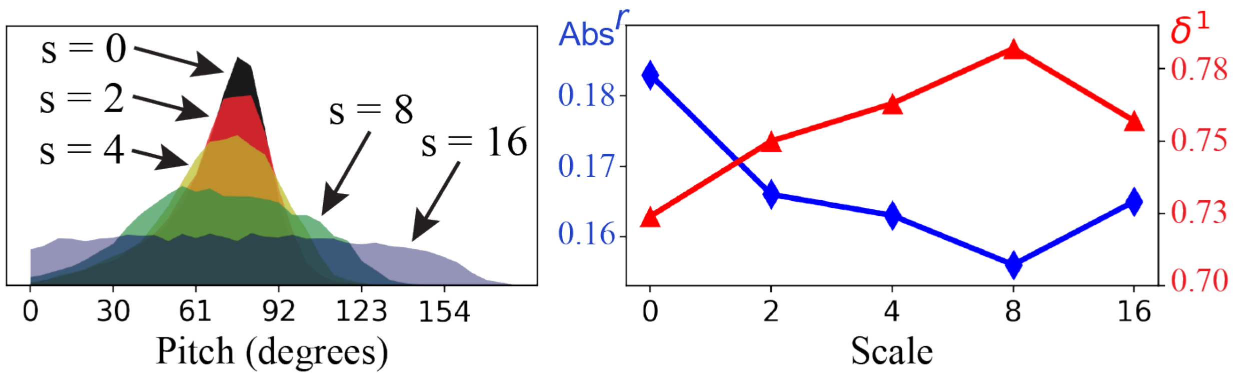


Figure 3.9: During training on InteriorNet *Natural* train-set, we randomly perturb camera pose to generate new training examples. We specify the scale of the perturbation $s = \{0, 2, 4, 8, 16\}$, meaning that, when $s = 2$, we randomly perturb pitch/roll/yaw angles by adding a perturbation within $[-s * 5°, s * 5°]$. **Left**: Applying more larger scale PDA "flattens" camera pose distribution of the whole training set. **Right**: We test each of the trained models on the InteriorNet *Uniform* test-set. We find applying more intense PDA consistently improves depth prediction until $s = 16$ (*i.e.*, rotating at most 80°), presumably when very large void regions are introduced in the synthesized training examples (Fig. 3.2).

the range of [-1.0, 1.0] to match the statistics of RGB images. We find this yields better performance than directly using $\mathbf{M}_{CPP}^{Clip}$. We visually compare some encoded maps in Fig. 3.10, where we see the clipping method introduces artificial stripes. Probably due to this, CPP-Clip does not perform as well as CPP that adopts inverse tangent transform, as shown in Table 3.5.

Hyperparameter analysis in CPP encoding. CPP encoding assumes that the camera moves in an empty indoor scene with an infinite floor and ceiling and the distance between two planes in the up direction is described by the parameter C. This distance is set to $C = 3$ meter in all experiments in this chapter unless otherwise specified. To verify the performance change w.r.t the distance C, we conduct experiments on InteriorNet *Natural* train-set with various distance $C = [4, 5, 6, 7, 8]$. As shown in Fig. 3.11, the performance of depth predictors are very robust in terms of the parameter C. In other words, CPP encoding improves the depth predictor performance and reduces the distribution bias consistently, regardless of the parameter C.

Quantitative Results of Blind Predictions. In this chapter, we visually demonstrate

Table 3.5: Comparisons of different encoding methods evaluated on InteriorNet test-sets. CPP applies an inverse tangent transform $\tan^{-1}$ in encoding the camera poses. In contrast, CPP-Clip replaces the $\tan^{-1}$ function with a clipping operation while keeping every other step the same as CPP encoding. Both CPP and CPP-Clip perform better than Vanilla model, demonstrating the effectiveness of our CPP method. Clearly, using the inverse tangent operator is better than clipping.

Models	*Natural-Test-Set*		*Uniform-Test-Set*	
	↓ better Abs^r/Sq^r/RMS-log	↑ better δ^1 / δ^2	↓ better Abs^r/Sq^r/RMS-log	↑ better δ^1 / δ^2
InteriorNet				
Vanilla	.154 / .148 / .229	.803 / .945	.183 / .146 / .250	.724 / .926
+ CPP-Clip	.109 / .124 / .204	.871 / .956	.111 / .096 / .189	.867 / .959
+ CPP	**.108 / .120 / .199**	**.872 / .958**	**.106 / .088 / .183**	**.876 / .961**
ScanNet				
Vanilla	.125 / .068 / .186	.837 / .962	.177 / .121 / .265	.711 / .928
+ CPP-Clip	.110 / .064 / .179	.869 / .964	.157 / .108 / .248	.758 / .936
+ CPP	**.108 / .060 / .171**	**.871 / .965**	**.154 / .106 / .239**	**.781 / .943**

Table 3.6: Comparison between using the average depth map (Avg) computed on the InteriorNet *Natural* train-set and the "blind predictor", which estimates depth solely from per-image CPP encoded maps *without* RGB images. We report results on InteriorNet *Natural* test-set. We find that "blind predictor" performs better than "avg depth map", implying the benefit of exploiting camera poses. We also report on two specific images on which "blind predictor" performs well compared to the average performance of Avg or Blind, as shown in Fig. 3.12. This further confirms that camera poses contain useful prior knowledge about scene depth.

Models	↓ better Abs-Rel/Sq-Rel/RMS-log	↑ better δ^1 / δ^2
Avg	.414 / .641 / .466	.346 / .638
Blind	**.342 / .519 / .395**	**.485 / .750**
Img-1	.041 / .010 / .082	.946 / .999
Img-2	.115 / .059 / .176	.793 / .990

that camera poses indeed contain prior knowledge of scene depth. The quantitative results of those visual examples are shown in Table 3.6, from which we find two key insights. First, the blind predictor achieves better performance than evaluating with average training depth maps, suggesting that camera poses alone contain the prior information about scene depth. In other words, training depth predictors with the camera pose alone are better than "random guess" from average training depth maps. Second, the blind predictor achieves promising performance on two images shown in Fig. 3.12 quantitatively. Together with the visualization of the prediction, we find that blind predictors make significantly more accurate depth prediction on floor regions, which confirms that the camera pose carries the prior knowledge

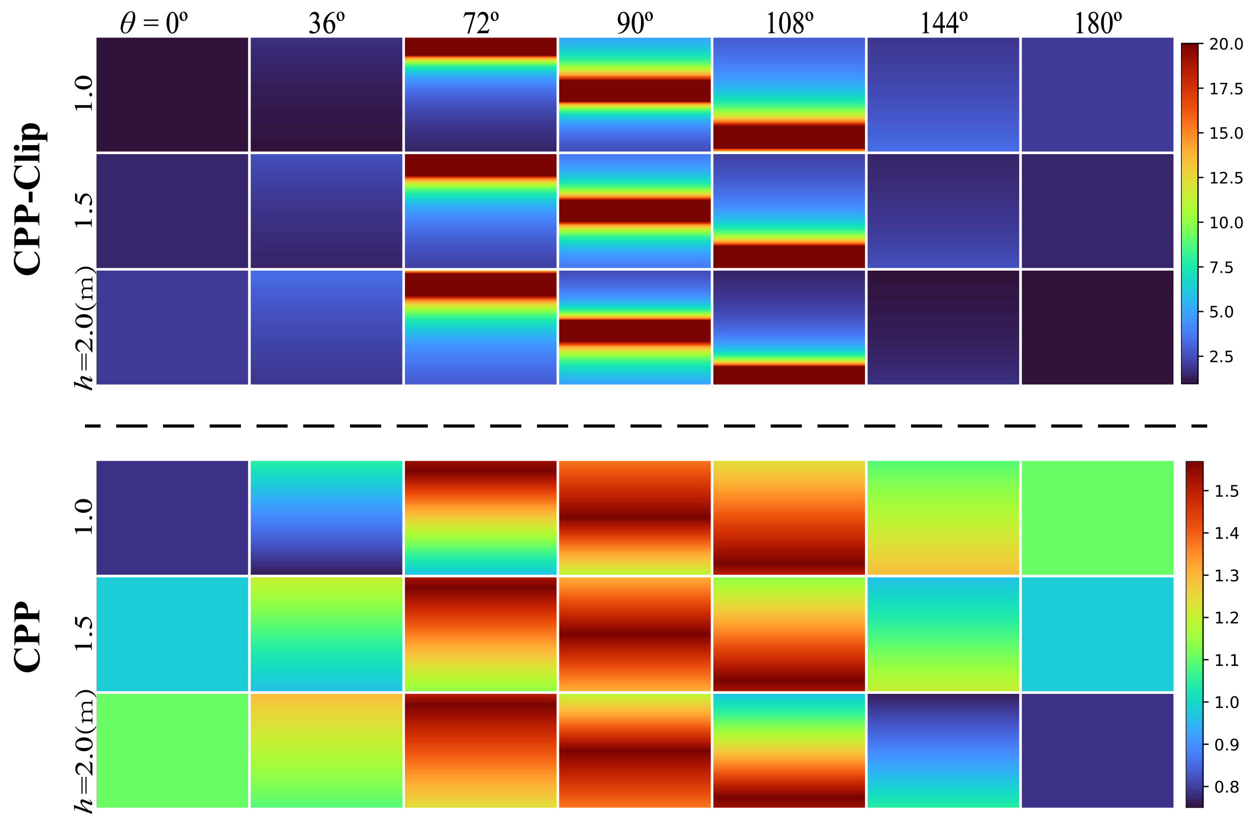


Figure 3.10: Visual comparisons of encoded maps of CPP and CPP-Clip with different pitch θ and camera height h. We set the threshold τ=20 for CPP-Clip. Encoded maps computed by CPP-Clip have the "red stripe" when the pitch is around 90° while CPP encoded maps have more smooth transitions when capturing the horizon.

about scene depth, especially on floor and ceiling regions.

Further Study of PDA Augmentation Scales. We provide a more detailed analysis of Vanilla depth predictor plus PDA with different scales of pitch and roll, individually. Please refer to Fig. 3.9 for detailed descriptions. As shown in Fig. 3.13, the performance of the depth predictor monotonically increase until augmenting pitch to the scale of 16. From the visual demonstrations, we believe that the performance drop is due to the introduction of large void regions. On the other hand, by rotating roll, we observe steady performance improvement, which demonstrates that PDA boosts the performance of depth predictors by generating training examples with diverse camera poses.

Further study of camera height and rotation in CPP encoding. CPP encodes

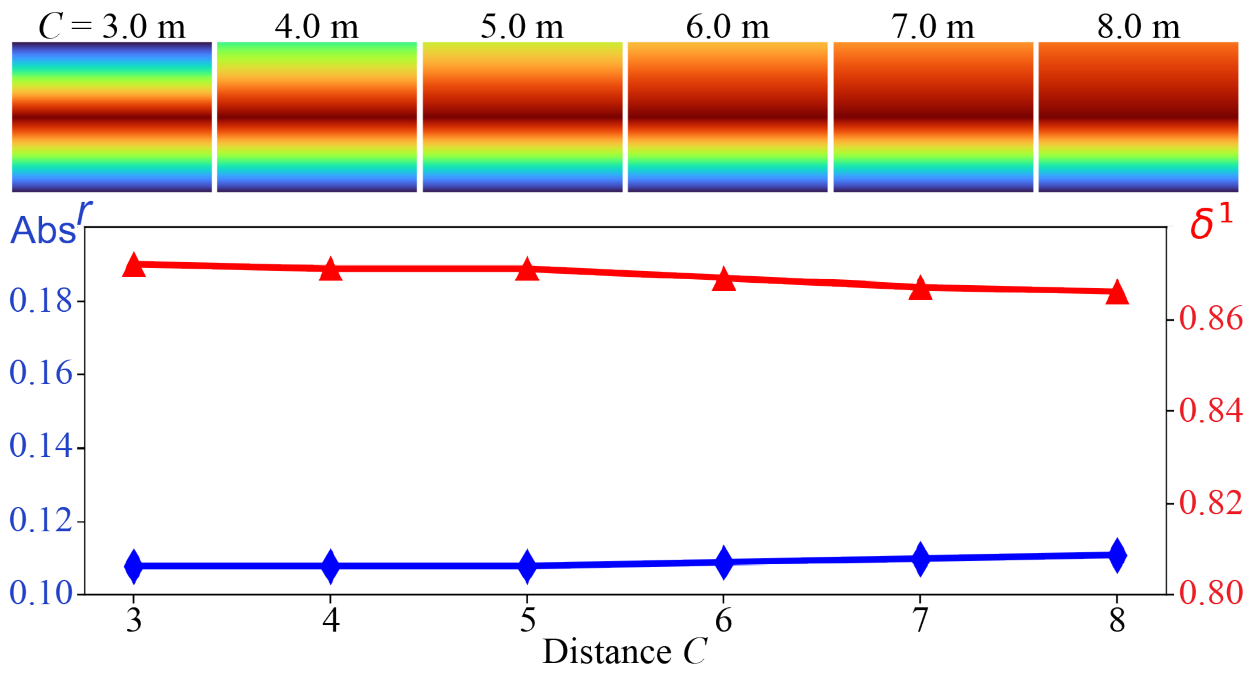


Figure 3.11: **Uppeer:** visualizations of CPP encoding with different hyper-parameter C (top); **bottom:** depth prediction performance as a function of hyper-parameter C. We train depth predictors on InteriorNet *Natural* train-set and test on its *Natural* test-set. From visual inspection, changing the parameter C only affects the part of CPP encoded maps where pixels are above the horizon. As shown by the performance curve, our proposed CPP encoding is very robust w.r.t different values of C.

rotation (roll and pitch) and camera height, however, it is still worth exploring which DOF is more important in CPP encoding. While it is nontrivial to define "importance" as pitch/roll and height have different units and ranges, we did study the pitch and height on InteriorNet (which has a nearly fixed roll). To apply CPP, we fixed either pitch or height and only encode the other with the true value. As shown in Fig. 3.14, we find that encoding the true camera height (top plot) performs better than the true pitch (bottom plot), and both perform better than the vanilla model. This implies that camera height is "more important" than pitch (probably roll as well).

CPP encoding with predicted poses. We study CPP to encode predicted poses. Specifically, we train depth predictors with CPP using true poses on *Natural* train-sets of the two datasets (Table 3.7). We test models on *Natural* and *Uniform* test-sets, respectively.

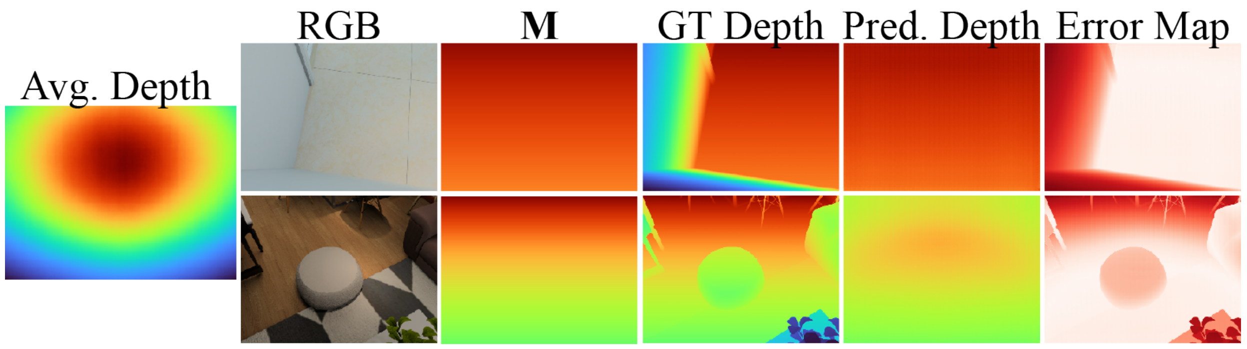


Figure 3.12: Illustration of how camera pose provides a strong depth prior through "blind depth prediction". Specifically, over the InteriorNet *Natural* train-set, we train a depth predictor solely on the CPP encoded maps **M** *without* RGB as input. For visual comparison, we compute an averaged depth map (shown left). We visualize depth predictions on two random examples. All the depth maps are visualized with the same colormap range. Perhaps not surprisingly, **M** presents nearly the true depth in floor areas, suggesting that camera pose alone does provide strong prior depth information for these scenes.

Note that in testing we encode the predicted poses given by a pose predictor. As shown in Table 3.7, CPP with predicted poses still outperforms Vanilla model; when jointly trained with PDA, CPP with predicted poses performs even better.

3.6 Conclusion

While large-scale datasets allow for end-to-end training of monocular depth predictors, we find the training sets naturally biased w.r.t. distribution of camera poses. As a result, trained predictors fail to make reliable depth predictions for testing examples captured under uncommon camera poses. We mitigate this bias with two novel methods, perspective-aware data augmentation (PDA) and camera pose prior encoding (CPP). We show that applying both our methods improves depth prediction on images captured under uncommon or never-before-seen camera poses. Moreover, our methods are general and readily applicable to other depth predictors, which can perform better when trained with PDA and CPP, suggesting using them in the future research of monocular depth estimation.

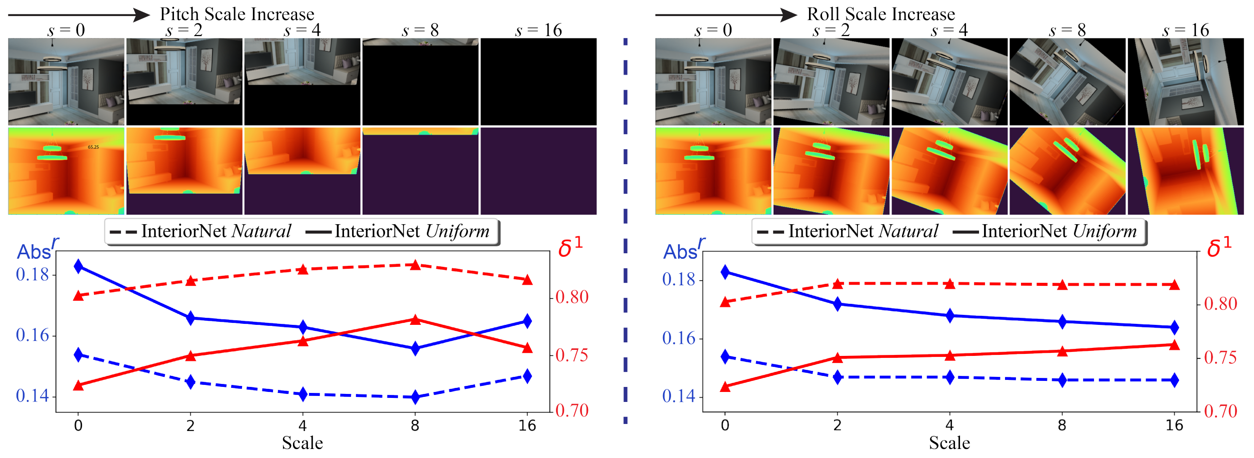


Figure 3.13: **Upper row:** visualizations of augmented examples using PDA with different scales. **Bottom row:** performance curves of depth predictor trained with PDA with different scales of pitch θ (left) and roll ω (right), respectively. Please refer to Fig. 3.9 for detailed descriptions. All models are trained on InteriorNet *Natural* train-set and evaluated on both *Natural* (dotted line) and *Uniform* (solid line) test-sets. As we increase the augmentation scale in pitch, the performance of depth predictors improves until scale s=16, when large void regions are introduced in the generated examples. On the other hand, increasing augmentation scales in roll lead to steady performance increments. In general, PDA consistently improves depth prediction over a Vanilla model trained without PDA.

Table 3.7: **CPP Encoding with Predicted Poses**. We train depth predictors with CPP using true poses on *Natural* train-sets of the two datasets. We test models on *Natural* and *Uniform* test-sets, respectively. Note that in testing we encode predicted poses given by a pose predictor. Clearly, CPP with predicted poses still outperforms Vanilla model; when jointly trained with PDA, CPP with predicted poses performs even better. Nevertheless, encoding predicted poses underperforms encoding true poses.

Models	*Natural-Test-Set*		*Uniform-Test-Set*	
	↓ better Abs^r/Sq^r/RMS-log	↑ better δ^1 / δ^2	↓ better Abs^r/Sq^r/RMS-log	↑ better δ^1 / δ^2
	`InteriorNet`			
Vanilla	.154 / .148 / .229	.803 / .945	.183 / .146 / .250	.724 / .926
+ CPP$_{pred}$	.142 / .132 / .212	.825 / .951	.164 / .121 / .228	.756 / .946
+ CPP	.108 / .120 / .199	.872 / .958	.106 / .088 / .183	.876 / .961
+ Both$_{pred}$	.135 / .127 / .205	.849 / .955	.148 / .114 / .213	.780 / .952
+ Both	**.095 / .101 / .180**	**.898 / .966**	**.091 / .069 / .161**	**.903 / .973**
	`ScanNet`			
Vanilla	.125 / .068 / .186	.837 / .962	.177 / .121 / .265	.711 / .928
+ CPP$_{pred}$	.116 / .065 / .180	.852 / .964	.169 / .117 / .255	.731 / .931
+ CPP	.108 / .060 / .171	.871 / .965	.154 / .106 / .239	.781 / .943
+ Both$_{pred}$	.111 / .061 / .173	.866 / .965	.159 / .111 / .247	.773 / .938
+ Both	**.102 / .052 / .160**	**.882 / .973**	**.143 / .097 / .230**	**.809 / .952**

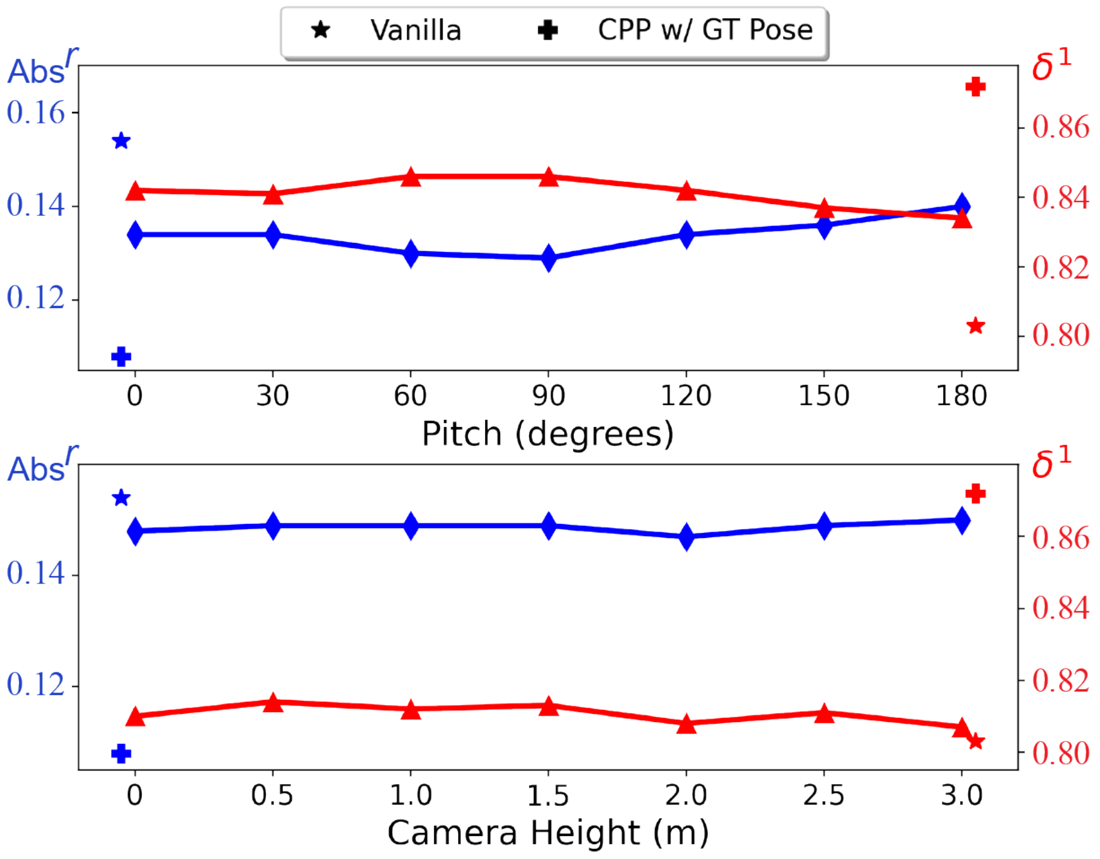


Figure 3.14: **Top:** CPP encoding with ground-truth camera height and fixed pitch. **Bottom:** CPP encoding with ground-truth pitch and fixed camera height. All models are trained/evaluated on InteriorNet *Natural* train/test-set. Comparing two blue or red curves across two plots, we find that encoding ground-truth height achieves better performance, suggesting height is "more important" than pitch. Moreover, encoding either ground-truth height or pitch outperforms Vanilla model.

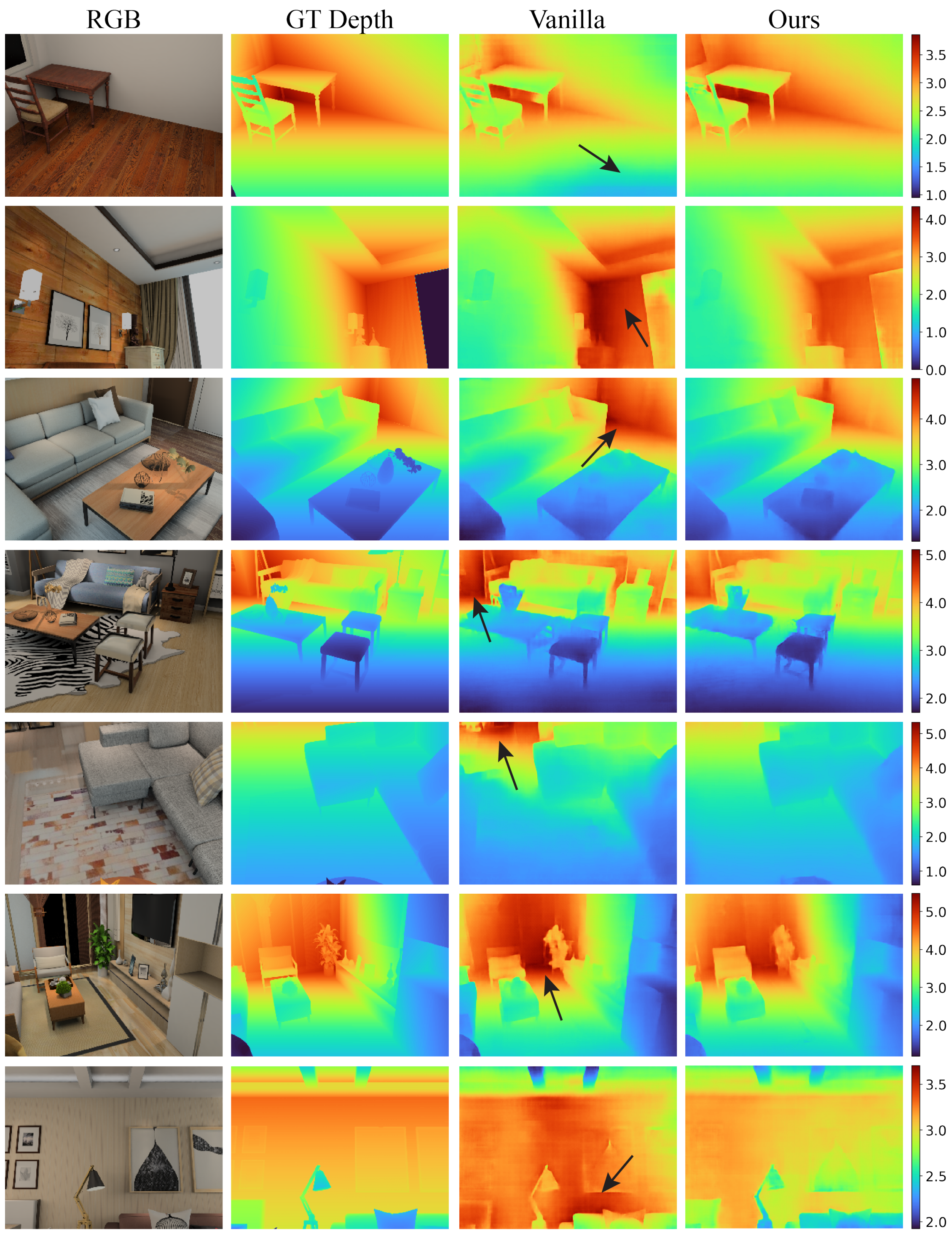


Figure 3.15: Depth predictions of Vanilla and our model (jointly applied CPP and PDA) on InteriorNet test-set. From these images captured under various camera poses, our model predicts better depth than Vanilla model in terms of the overall scale.

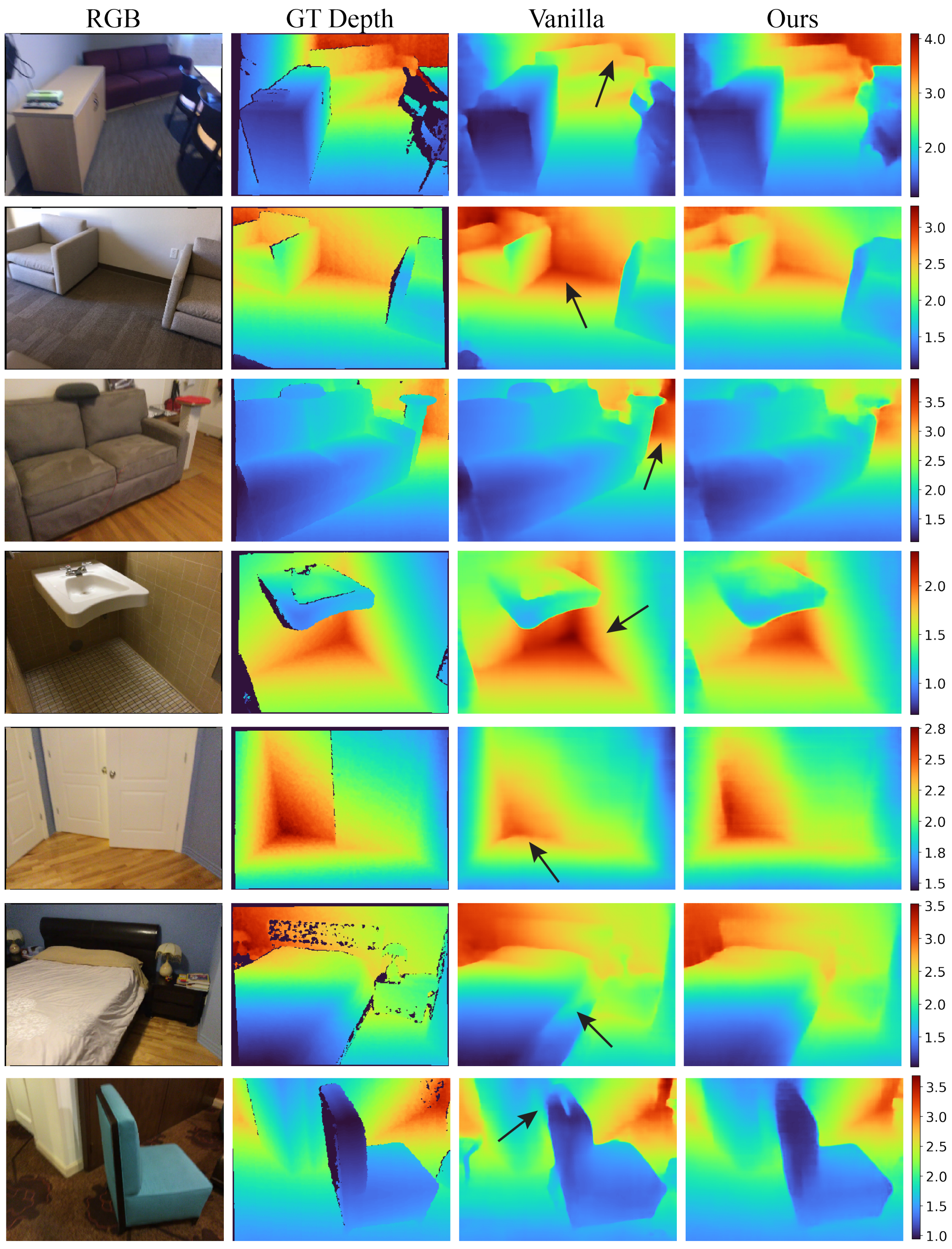


Figure 3.16: Depth predictions of Vanilla and our model (jointly applied CPP and PDA) on ScanNet test-set. From these images captured under various camera poses, our model predicts better depth than Vanilla model in terms of the overall scale.

Chapter 4

Reference-based image inpainting leveraging depth maps

Reference-guided image inpainting restores image pixels by leveraging the content from another single reference image. The primary challenge is how to precisely place the pixels from the reference image into the hole region. Therefore, understanding the 3D geometry that relates pixels between two views is a crucial step towards building a better model. Given the complexity of handling various types of reference images, we focus on the scenario where the images are captured by freely moving the same camera around. Compared to the previous work, we propose a principled approach that does not make heuristic assumptions about the planarity of the scene. We leverage a monocular depth estimate and predict relative pose between cameras, then align the reference image to the target by a differentiable 3D reprojection and a joint optimization of relative pose and depth map scale and offset. Our approach achieves state-of-the-art performance on both RealEstate10K and MannequinChallenge dataset with large baselines, complex geometry and extreme camera motions. We experimentally verify our approach is also better at handling large holes.

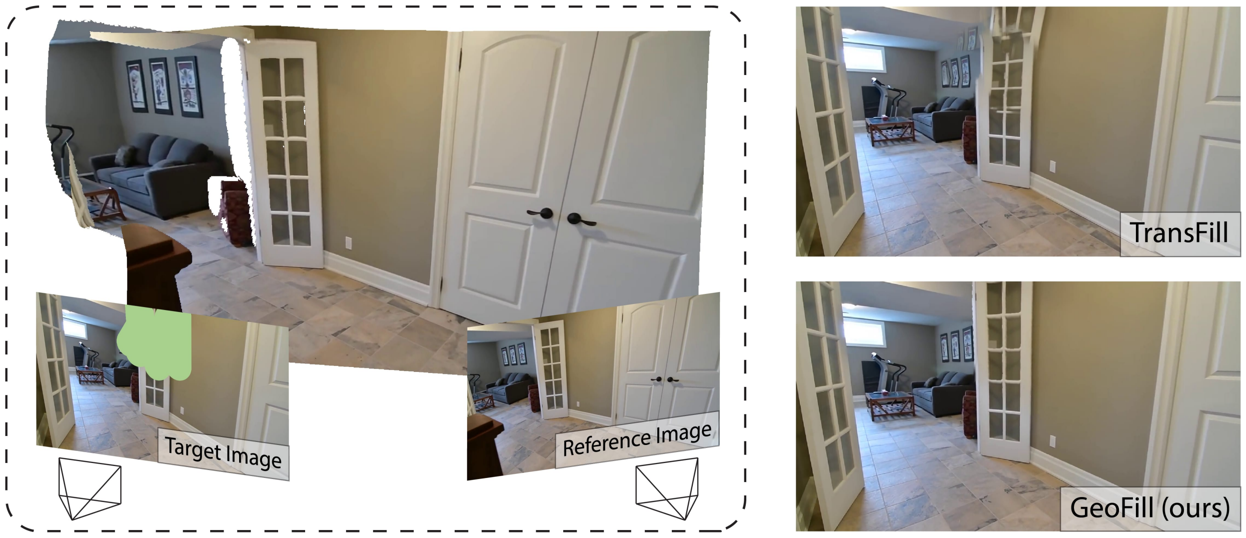


Figure 4.1: Given a reference image and a target image with hole, GeoFill utilizes predicted correspondence matching and depth maps to estimate a 3D mesh and relative camera pose and intrinsics. Compared to TransFill, the previous state-of-the-art approach, GeoFill handles complex scenes better by iteratively refining predicted depth maps and relative pose.

4.1 Background

Image inpainting aims to plausibly restore missing pixels within a given hole region. Existing single image inpainting models [270, 256, 271] solve this problem without additional information by leveraging the knowledge learned from large scale training data or existing patches within the image. These methods become less reliable when input images contains large holes and the filling regions are complex in structures and textures.

In 2021, Zhou *et al.* [299] proposed a novel inpainting task called reference-based image inpainting. It aims at filling the hole regions of a “target" image using another single source photo taken of the same scene. This foreground removal application is particularly useful when people take photos at museums or famous landmarks where the background is unique. It is nearly impossible for existing single image inpainting to faithfully restore what was *actually there in the background* for such cases. Inpainting with a reference image is appealing and indispensable, and also feasible, since other photos taken at different viewpoints of the same scene can be available, *e.g.*, in the users' albums or even downloaded from the Internet.

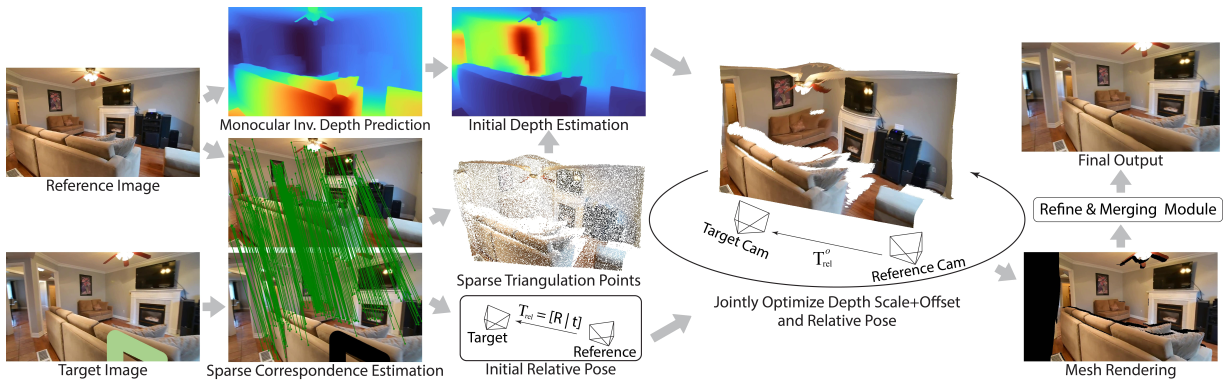


Figure 4.2: Overview of our system pipeline. We are given a target image with a hole and a source image. We aim at warping the single-source image to the target to fill the hole. We first estimate the relative pose as well as predict the monocular depth of the source, and then adjust the scale and offset of the depth map. After that, to mitigate the potential errors caused by deep models, we jointly optimize camera relative pose and depth map scale and offset to make the depth map and image contents well-align near the hole region. Finally, we render the reprojected source and refine it using post-processing.

However, reference-based image inpainting is very challenging and not well explored. That is because reference-based inpainting is a photography-oriented task that has only one single reference frame as “source", and has usually large baseline or challenging camera movements between the target and source. Therefore, the target and source images cannot be easily aligned to fill in the hole due to issues such as parallax. To address the alignment problems, the previous state-of-the-art method TransFill [299] has a strong assumption that the scenes can be blended by multiple planar structures. It clusters the predicted depth on matched feature points and utilizes multiple homographies to fill the hole. However, real scenes rarely consist of only a few planar surfaces, and even when they do, it can be hard to identify the relevant planar surfaces. This indicates that to better solve the problem of reference-based inpainting, *it is crucial to understand the camera positions and geometry of the 3D scene, especially near the hole region* in order to find the appropriate content for hole filling.

In this work, we propose a more principled approach that fills the hole region by explicitly estimating the 3D scene structure from two limited views. Specifically, we first estimate sparse feature correspondences, from which we derive an initial relative pose between the two views. We then predict a monocular dense depth map of the source image and determine a scale and offset that aligns the depth map with the target using a sparse 3D triangulated point cloud. To mitigate the prediction errors and improve alignment accuracy, we next jointly optimize the depth scale and offset and the relative pose using a fast differentiable 3D reprojection of the source image to the target. We synthesize a warped source image by rendering a textured source mesh with the optimized depth and target pose, and fill dis-occluded regions with single image inpainting. Lastly, we adjust the exposure, white balance, lighting, and correct any residual misalignments before pasting the result back into the hole region.

In summary, our GeoFill is the first to apply a more principled approach for reference-based inpainting, *i.e.*, the first to leverage an explicit non-planar 3D scene representation given only limited two-view RGB images (no camera pose information). Compared with the previous state-of-the-art, GeoFill better handles complex 3D scene structures within the hole, wide-baseline image pairs and larger holes. Extensive experiments demonstrate that our methods achieve the best perceptual quality on various scenes in benchmarks and real user cases.

4.2 Related Work

Image inpainting. Traditional image inpainting models rely on hand-crafted heuristics. Diffusion-based approaches [17] propagate pixel colors from the background to the hole region. These approaches generate artifacts when the hole size is large or texture variations are significant. Alternatively, patch-based approaches [249, 13] search for similar patches outside the hole region to complete the missing regions. Although these approaches offer high quality texture by copying texture patches, the filled regions may be inconsistent with regions around the hole due to lack of high-level structural understanding of the entire image.

Recent deep models fill the hole by learning from large amounts of training data. Context encoders [183] generate semantically plausible content in the hole by encoding the surroundings. Iizuka et al. [108] adopt two discriminators to ensure the inpainted content is both locally and globally consistent. Artifacts can be reduced along the hole boundary by filtering using partial [149] or gated [271] convolutions. Some recent inpainting models improve the generated image quality with additional information, such as edges [168], segmentation masks [221], and low frequency structures [194, 147]. Moreover, several papers show deep neural networks can fill holes on high resolution images [260, 264, 275]. Despite the significant advancements in single inpainting models, filling with one single image remains fundamentally an ill-posed problem [292]. Image inpainting with additional information is also explored in the literature, such as inpainting with stereo images [244, 19, 9, 162, 161] and utilizing more than one image [230]. TransFill [299] is closely related to our work: it performs reference-guided inpainting by warping a reference image with multiple homographies. However, due to the planar nature of homographies, TransFill has a limited ability to handle image pairs with complex 3D structures, wide baselines, or significant disocclusions.

Video Inpainting. Classical works mainly focus on globally optimizing patch-based energies [184, 250, 87]. Recent work often adopts deep generative models for better inpainting performance. Wang *et al.* introduce a data-driven framework that jointly learns temporal structure and spatial details [240]. Onion-Peel Network (OPN) proposes to fill in the missing region progressively with the spatio-temporal attention [174]. Spatial-Temporal Transformer Network (STTN) adopts a deep generative model with adversarial training along the spatial-temporal dimension to mitigate blurriness and temporal artifacts [274]. Note that video inpainting approaches heavily exploit the dense temporal information in the video while we only have one single reference image which is a much harder scenario.

Two-view Geometry. SfM establishes correspondences between two monocular frames and subsequently estimates 3D structure [94, 205, 296, 243, 160]. In classic geometric vision, it is well understood that the camera poses as well as depth for corresponding points can be computed from feature matching points alone [156, 96]. Traditional methods utilize hand-crafted descriptors [157, 16, 197] to build sparse correspondence for the subsequent fundamental matrix estimation with the 8-point algorithm [95]. Learned local features have shown great success in recent works [263, 49, 50] together with the learning-based feature matching models, such as SuperGlue [200] or differentiable formations of RANSAC [23, 191, 24]. Another alternative is to directly estimate relative pose using an end-to-end pose estimation network [114]. We leverage these recent advances (specifically OANet [276]). Our method uses components inspired by SfM, however, our differentiable joint optimization stage is novel and has been carefully formulated for our task.

Monocular Depth Estimation. Predicting depth from a single image is an ill-posed problem. However, learning based approaches have shown impressive performance by either treating monocular depth estimation as a regression or classification task [63, 128, 257, 92, 258, 71, 106, 5, 133, 288, 107, 287, 209]. Recent advances include BTS [133], which introduces local planar guidance layers to guide the features to full resolution instead of standard upsampling layers during the decoding phase. DAV [107] proposes to exploit the co-planarity of objects in the scene via depth-attention volume. DPT [190] leverages the high quality intermediate representation from transformers and become state-of-the-art.

4.3 Method

Suppose we are given a target image $\mathbf{I}_t$ with a hole $\mathbf{M}$ to be filled, and a reference (source) image $\mathbf{I}_s$ of the same scene. Our goal is to find a 3D-aware warped source image $\mathbf{I}_{s\to t}$ that geometrically aligns the source image to the target image which can be used to fill the hole. The final composite image can be represented as $\mathbf{I}_t^{\text{comp}} = \mathbf{I}_t \odot \mathbf{M} + (\mathbf{M}_{\text{single}}\mathbf{I}_{s\to t} + (1 - \mathbf{M}_{\text{single}}) \odot \mathbf{I}_{\text{single}}) \odot (1 - \mathbf{M})$, where $\mathbf{M}_{\text{single}}$ is the blending map to merge the warped source with the single image inpainting result $\mathbf{I}_{\text{single}}$. Note that ideally there should be enough contents that

GeoFill can copy from the source image to the target hole region, *i.e.*, the source image I_s is useful. Cases with very few target-source content pixels overlapping inside the hole will cause GeoFill to fall back to the single image inpainting $\mathbf{I}_{\text{single}}$.

To compute the final warping matrix and use it to reproject the source image, as shown in Figure 4.2, we propose a pipeline consisting of three stages named initialization, joint optimization, and rendering and post-processing. In the first stage, we establish sparse correspondences between $\mathbf{I}_s$ and $\mathbf{I}_t$ and estimate the relative pose $\mathbf{T}_{\text{rel}}$ between two views. Meanwhile, we obtain a dense depth map of the source image using a pretrained deep model. Then we align the scale and offset of the predicted depth map with the sparse 3D triangulated feature points. In the second stage, we mitigate the potential errors of the initial guess of pose and depth by optimizing the related parameters so contents well-align near the hole. Finally, we render the warped image using the optimized parameters, and post-process it to address any residual spatial and color misalignments like TransFill. We will introduce each stage in the following sections.

4.3.1 Initialization Stage

Initialize Relative Pose Our approach first estimates the relative pose $\mathbf{T}_{\text{rel}}$ based on predicted sparse correspondences. We extract sparse corresponding feature points $\mathbf{P}_t$ and $\mathbf{P}_s$ between the target and source images, and compute the fundamental matrix $\mathbf{F}$ between $\mathbf{I}_s$ and $\mathbf{I}_t$ via the normalized 8-point algorithm [95] using RANSAC [69]. From $\mathbf{F}$, we derive the relative pose $\mathbf{T}_{\text{rel}}$ using the classic multi-view geometry algorithm mentioned in [95].

Initialize Dense Depth Map We then predict the inverse depth map on the source image using a pretrained monocular depth estimator as the cues to estimate the real source depth. Mathematically, we only need the source depth and $\mathbf{T}_{\text{rel}}$ to compute the 3D-aware warp $\mathbf{I}_{s\to t}$. However, the source depth is predicted up to an unknown scale and offset by a pretrained deep model, therefore, we need to solve for these such that the initial source depth $\mathbf{D}_s^i$ best matches the estimated relative pose. Note the estimated translation in $\mathbf{T}_{\text{rel}}$ is normalized and will not match the arbitrary scale in the predicted depth maps from pretrained deep models. As suggested in [285], aligning dense depth to sparse triangulated points is much simpler than rescaling the relative pose. Therefore, we first triangulate points with the relative pose, then align the scale of depth predictions with triangulation to subsequently match the scale of the relative pose.

Specifically, a 3D triangulated point $\mathbf{x}$ with point $q_s \in \mathbf{P}_s$ and $q_t \in \mathbf{P}_t$ is computed as,

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}[E(\mathbf{r}_s, \mathbf{x})]^2 + [E(\mathbf{r}_t, \mathbf{x})]^2, \tag{4.1}$$

where $\mathbf{r}_s$ represents the ray shooting from the source camera center through the point q_s on the image plane, $\mathbf{r}_t$ is the ray from the target camera following a similar analogy, and E measures the Euclidean distance between two inputs. In this way, we compute a set of 3D triangulated points $\mathbf{X}$ using all matching sparse correspondences. In order to form the linear problem to compute the scale and offset, we first compute a sparse triangulated depth map $\mathbf{D}_{\text{tri}}$ by projecting 3D triangulated points to source camera coordinates. Note $\mathbf{D}_{\text{tri}}$ is in the same scale as the relative pose. Therefore, we correct $\mathbf{D}_s$ to match $\mathbf{D}_{\text{tri}}$, which subsequently matches the scale of the relative pose. We correct $\mathbf{D}_s$ by estimating two scalars, the scale s^i and offset b^i associated with the depth map, by solving a linear least square problem. The initial depth maps are then expressed as: $\mathbf{D}_s^i = s^i\mathbf{D}_s + b^i$.

4.3.2 Joint Optimization Stage

To mitigate the effects of potential errors in sparse correspondence and depth estimation since deep models can be not robust or generalized enough, we further introduce an optimization module to improve the quality of $\mathbf{I}_{s\to t}$. We optimize the depth scale, offset, and the relative pose that jointly define $\mathbf{I}_{s\to t}$ in the 3D scene. Specifically, we convert the rotation matrix into quaternions, which leads to a total of 9 parameters to optimize. Both relative pose and the initial depth computed in the previous section are used as the initial guess for the optimization. Our optimization contains 3 different loss functions: a multiscale photometric loss $\mathcal{L}_{\text{photo}}$, a feature correspondence loss $\mathcal{L}_{\text{feat}}$, and a negative depth penalty $\mathcal{L}_{\text{negD}}$.

Multiscale Photometric Loss $\mathcal{L}_{\text{photo}}$ measures the pixel-level color difference between $\mathbf{I}_{s\to t}$ and the $\mathbf{I}_t$ outside the hole region. We downsample both $\mathbf{I}_{s\to t}$ and $\mathbf{I}_t$ and sum the normalized color difference across different resolutions. Specifically, we build Gaussian pyramids on both $\mathbf{I}_{s\to t}$ and $\mathbf{I}_t$ using an RGB representation for the source image and an alpha-premultiplied RGBA representation on the target image to incorporate the hole region properly into the target image. Computing a multi-scale photometric loss within each iteration is obviously computationally expensive. Moreover, the optimization might also get trapped into the local minima associated with the finest resolution due to poor initialization [303]. To accelerate the computation speed and find better solutions, we adopt a coarse-to-fine optimization strategy, which means we first compute photometric loss on the most coarse level and move to the finer level once the convergence criteria at the current pyramid level are met. Additionally, instead of building a 3D triangle mesh and rendering from the target view at each iteration, we use a much more efficient differentiable 3D reprojection to find a warping field that computes $\mathbf{I}_{s\to t}$ from $\mathbf{I}_s$ with bilinear interpolation. Mathematically, we have:

$$\mathbf{I}_{s\to t} = \text{bilinear}(\mathbf{I}_s, \text{reproj}(\mathbf{K}, \mathbf{T}_{\text{rel}}, \mathbf{D}_s^o)), \tag{4.2}$$

where reproj() represents the reprojection operation. The photometric loss at a given resolution with the pyramid is:

$$\mathcal{L}_{\text{photo}} = \frac{1}{|\mathbf{M}|}\sum \mathbf{W} \odot ||\mathbf{I}_{s\to t} \odot \mathbf{M} - I_t \odot \mathbf{M}||^2. \tag{4.3}$$

Here $\mathbf{W}$ is a pixel importance weight map discussed shortly.

Feature Correspondence Loss $\mathcal{L}_{\text{feat}}$ computes the distance between reprojected matching feature points in the source images and the target images. We use the reproj() operator on all 2D image coordinates in $\mathbf{P}_s$ to get another set $\mathbf{P}_{s\to t}$. Then, we compute the average distance between $\mathbf{P}_{s\to t}$ and $\mathbf{P}_t$. However, the average distance of all points is very sensitive to outliers, *i.e.*, very few outliers dominate the loss function. To reduce the effects of the outliers on the loss function, we adopt the general robust loss function from [14]. The general form of the loss function is:

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right), \tag{4.4}$$

where α and c are the shape and scale parameters, respectively. In our experiments, we set $\alpha = -2$ and $c = 10$. Then, feature correspondence loss is written as:

$$\mathcal{L}_{\text{feat}} = \frac{1}{|P_{s\to t}|} \sum_{m=0}^{|P_{s\to t}|} \mathbf{W}(\mathbf{q}_t^m) f(||\mathbf{q}_{s\to t}^m - \mathbf{q}_t^m||, \alpha, c), \tag{4.5}$$

where f is the general robust loss function, $\mathbf{q}_{s\to t}^m$ and $\mathbf{q}_t^m$ is the m^{th} point in $P_{s\to t}$ and P_t, respectively.

Negative Depth Penalty $\mathcal{L}_{\text{negD}}$ aims to penalize negative values in the remapped depth. Although depth predictions have arbitrary scale and offset, they should have all positive values, meaning that a fragment of geometry associated with a pixel should never move behind the camera. Mathematically, we adopt a hinge loss function:

$$\mathcal{L}_{\text{negD}} = \sum \max\{0, -\mathbf{D}_s^o\}. \tag{4.6}$$

Our final objective function is written as: $l = \lambda_1 \mathcal{L}_{\text{photo}} + \lambda_2 \mathcal{L}_{\text{feat}} + \lambda_3 \mathcal{L}_{\text{negD}}$, where $\{\lambda_j\}$ are weights. The convergence criteria and the average running time are discussed in the supplemental material.

Pixel Importance Weight Map W. When computing the photometric loss or feature correspondence loss, we assign weights to each pixel to replace uniform weighting. It encourages our optimization to better align the warped source with the target image both locally and globally.

The first type of weighting strategy is hole-distance weighting $\mathbf{W}_h$. We encourage the optimization to focus on the regions close to the hole boundary since those pixels are more important for filling the hole regions. To achieve this, we apply the distance transform to the hole image and obtain a distance map $\mathbf{M}_h$, where each pixel records the Euclidean distance to the closest boundary pixel of the hole $\mathbf{M}_h$. We compute the weight at each pixel using a Gaussian function $\mathbf{W}_h = \exp(-\mathbf{M}_h^2/2\sigma^2)$, where σ is a hyperparameter that adjusts how weights change w.r.t distance to the hole boundary.

The second type is the edge-based weighting $\mathbf{W}_e$. This is because high-gradient edge regions are more salient while checking the alignment quality so we intend to give strong edges larger weights. We compute a multi-scale Canny edge map by first applying Gaussian blur on $\mathbf{I}_t$ with N different kernel sizes, running the Canny edge detector [30], and dilating each edge to get $\{e_1, e_2, \cdots, e_N\}$. Our pixel-level edge-based weight map becomes $\mathbf{W}_e = \sum_{k=1}^{N} e_k / \sum_p e_k(p)$, where the inner sum is over spatial coordinates. Our overall weighting map is $\mathbf{W} = \mathbf{W}_h \odot \mathbf{W}_e$.

4.3.3 Rendering and Postprocessing Stage

Mesh Rendering. After optimization, we find $\mathbf{T}^o_{\text{rel}}$, s^o, and b^o, the camera relative pose and depth map scale and offset, respectively. However, computing $\mathbf{I}_{s\to t}$ using depth reprojection always has some gaps between valid pixels due to it relying on a forward warping for efficiency in the inner loop of the optimization. One way to overcome this problem is interpolation but that has two disadvantages. First, the reprojected image is sparse and for regions with multiple layers of depth, pixels for a far-away depth layer might splat between pixels of a closer depth layer, which could result in interpolations that do not fully remove hidden surfaces. Additionally, interpolation cannot distinguish between holes due to disoccluded regions or simple gaps between the pixels due to the forward warping.

To address the above problems, we choose to render a textured mesh to get the final $\mathbf{I}_{s\to t}$. We first build a triangle mesh with a regular grid from the source view. The mesh vertices are computed by projecting the optimized depth $\mathbf{D}_s^o$ to the 3D space and the texture is the RGB colors of the source image. After obtaining the mesh, we drop the edges around depth discontinuities. We adopt a simplified version of the footprint algorithm [248] by comparing the depth values between connected vertices. We drop the edge between two vertices v_i and v_j if $\frac{2|d(v_i)-d(v_j)|}{d(v_i)+d(v_j)} > \epsilon_{\text{edge}}$, where $d(v_i)$ is the depth value of the vertex v_i and ϵ_{edge} is a predefined threshold. After building the triangle mesh, we render the target view with $\mathbf{T}_{\text{rel}}^o$. Note that we also normalize the mesh to the unit size before rendering. The textured mesh densely fills in pixels and removes hidden surfaces. It also allows us to use the rendered alpha channel to find pixels where there is no ray intersection with the mesh, which represent disoccluded regions near depth discontinuities or regions outside the photo: these are later filled by single image inpainting. We use PyTorch3D [193] as our renderer.

Refinement and Merging. With the rendered image, we apply the color-spatial transformation (CST) module from TransFill to further improve any small residual spatial misalignments and correct color and exposure differences. Lastly, we merge the output from CST with results from the single image inpainting model as in TransFill to handle the disocclusions and regions outside the photo.

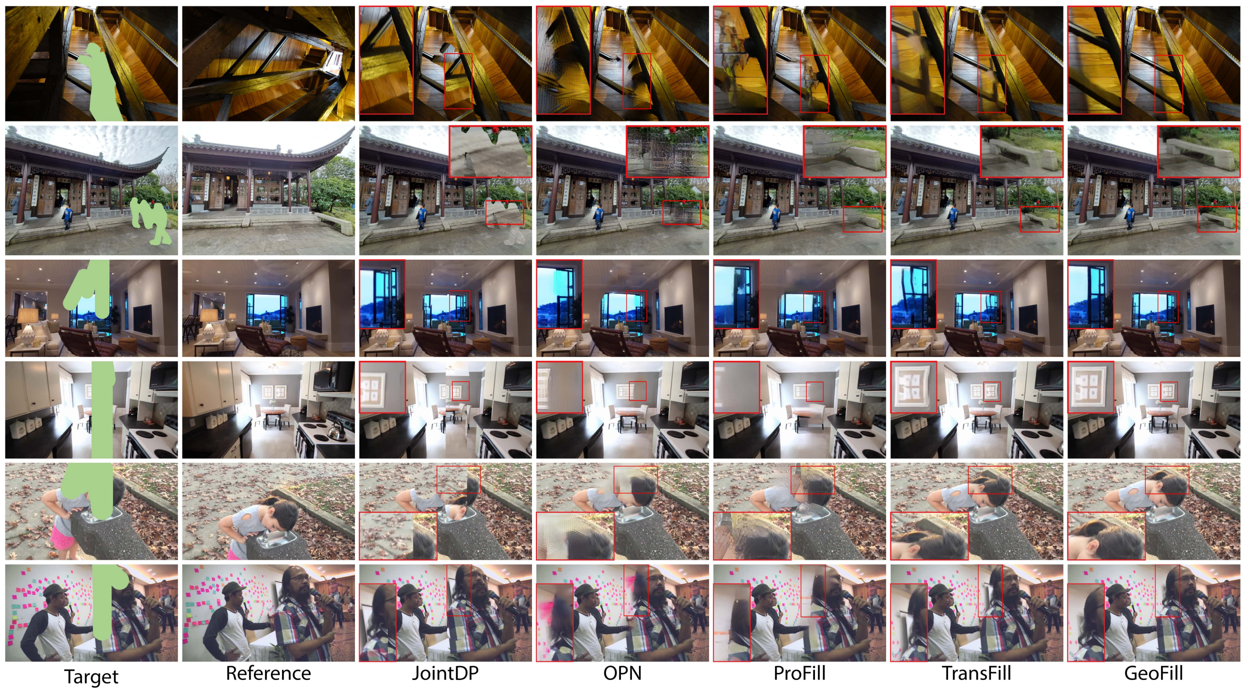


Figure 4.3: Qualitative comparison of GeoFill against other baselines on user-provided images (top 2 rows), RealEstate10K (mid 2 rows), and MannequinChallenge dataset (last two rows).

4.4 Experiments

Datasets. No large-scale image-based dataset for reference-based inpainting is available. So we follow TransFill to randomly sample multiple image pairs from video-based datasets because it is easier to simulate user behaviours and analyze the target-source difference (or camera view changes). During evaluation we only use one single reference frame. This is different from what video inpainting works did. We follow DeepFillv2 [271] to generate random free-form brush stroke masks and evaluate GeoFill and other baselines on the following datasets.

RealEstate10K [297]: It contains a diverse collection of YouTube video sequences shot from a moving camera for both indoor and outdoor scenes. Each video clip contains a variety of views of the same scene. We randomly sample 500 videos and select one pair of images in each video sequence with a specific frame difference (FD). Specifically, we sample FD=25, 50, and 75 to build three different sets with a resolution of 720×1280 while automatically filtering out image pairs without enough overlapping content inside the hole by checking the number of matching sparse features. Note that our filtering mechanism is only for the purpose of simulating user behaviors and removing non-useful image pairs. In practice, we believe users could simply check overlap between photos by visual inspection.

MannequinChallenge [143]: This is a challenging dataset with video sequences shot from a hand-held camera freely roaming in a scene of people with frozen poses. This dataset contains more than 170K frames and corresponding camera poses derived from about 2k YouTube videos. The camera motion in this dataset is more extreme and scene complexity is much higher due to diverse frozen human poses and rich background objects. We reduce the sampling FD to ensure enough overlap between image pairs. Similar to the previous dataset, we randomly sample 3 subsets with FD=10, 20, 30, where each contains 500 image pairs of 720×1280.

Real User-provided Images: We also evaluate our approach on the real user-provided images like TransFill to validate the generalization ability and practicability.

Baselines. In addition to TransFill, which is directly related to our work, we compare our approach against several different types of baselines to evaluate final inpainting performance. The first type is state-of-the-art video completion models. OPN [174] achieves high quality inpainting results with spatio-temporal attention. STTN [274] proposes to optimize a spatial-temporal adversarial loss function. In addition, we compare against state-of-the-art single-image inpainting methods including ProFill [275] and CoModGAN [283]. We use their off-the-shelf pretrained weights directly since they are both trained on more diverse scenes in Places2 [294] than RealEstate10K and have strong generalization ability. Lastly, we compare a two-view SfM based approach [285] — which we refer to as JointDP — by warping the source image with the jointly estimated relative pose and depth using dense correspondence. To ensure fairness in the comparison, we use the same depth predictor and rendering process as in GeoFill while keeping all other settings the same as in the original work.

Table 4.1: Quantitative comparisons of GeoFill against other baselines on the RealEstate10K dataset.

Model	FD=25	FD=50	FD=75
JointDP [285]	22.46 / 0.9469 / 0.1011	21.76 / 0.9457 / 0.1063	20.89 / 0.9423 / 0.1122
OPN [174]	28.41 / 0.9684 / 0.0525	27.80 / 0.9669 / 0.0570	26.91 / 0.9634 / 0.0624
STTN [274]	28.83 / 0.9696 / 0.0710	28.26 / 0.9697 / 0.0721	27.59 / 0.9680 / 0.0751
ProFill [275]	27.45 / 0.9642 / 0.0775	27.67 / 0.9654 / 0.0755	27.37 / 0.9639 / 0.0768
CoModGAN [283]	26.02 / 0.9594 / 0.0703	26.14 / 0.9607 / 0.0686	25.88 / 0.9596 / 0.0697
TransFill [299]	32.03 / 0.9764 / **0.0461**	30.64 / 0.9732 / 0.0540	29.24 / 0.9694 / 0.0608
GeoFill (Ours)	**32.57** / **0.9775** / 0.0467	**31.47** / **0.9748** / **0.0525**	**30.43** / **0.9717** / **0.0581**

Implementation Details. We follow TransFill to extract SIFT features [158] and feed them into OANet [276] to reject outliers and establish correspondences. This combination was already proven to be quite robust. A better matching strategy could always be adopted to further improve results while leaving the rest of our framework intact. Our pretrained monocular depth predictor is DPT [190]. No ground-truth camera intrinsic information is required to run our approach. We use fixed camera intrinsic parameters for all images by setting focal length to 750 and principal point to the image center. We use the pretrained CST module from TransFill [299] without finetuning. This module generalizes well to MannequinChallenge and user provided images. Our pipeline is implemented with PyTorch [181] and we choose DiffGrad [58] as our optimizer due to its fast convergence speed. In the optimization step, we use a constant learning rate 10^{-2} and the maximum number of iteration is set to 10^4. The loss weights λ_1, λ_2, and λ_3 are 10, 10, 0.5, respectively. In the coarse-to-fine optimization strategy, the number of pyramid levels is 4 and the maximum number of cumulative iterations at each level from coarse to fine are 4×10^3, 7×10^3, 9×10^3, 10^4. We set the σ in hole-based weighting to 192 pixels. In edge-based weight, we compute 4 different Canny edge maps and dilate each of them with a kernel size equal to 4. In mesh rendering, the edge threshold ϵ_{edge} is 4×10^{-2}.

Table 4.2: Quantitative comparisons of GeoFill against other baselines on the MannequinChallenge dataset.

Model	FD=10	FD=20	FD=30
JointDP [285]	20.13 / 0.9346 / 0.1087	19.52 / 0.9290 / 0.1195	19.38 / 0.9315 / 0.1177
OPN [174]	25.63 / 0.9628 / 0.0605	24.92 / 0.9584 / 0.0698	24.84 / 0.9591 / 0.0702
STTN [274]	25.60 / 0.9623 / 0.0803	25.09 / 0.9602 / 0.0865	24.94 / 0.9613 / 0.0844
ProFill [275]	25.04 / 0.9589 / 0.0808	25.02 / 0.9582 / 0.0836	25.22 / 0.9599 / 0.0810
CoModGAN [283]	23.39 / 0.9504 / 0.0770	23.14 / 0.9486 / 0.0808	23.36 / 0.9503 / 0.0791
TransFill [299]	28.01 / 0.9680 / 0.0569	26.56 / 0.9628 / 0.0688	26.17 / 0.9632 / 0.0701
GeoFill (Ours)	**28.85 / 0.9702 / 0.0553**	**27.72 / 0.9658 / 0.0652**	**27.44 / 0.9664 / 0.0665**

4.4.1 Quantitative Results

The quantitative results comparing our approach with other baselines are shown in Table 4.1 and 4.2. We report the PSNR, SSIM and LPIPS[279] on the RealEstate10K and the MannequinChallenge datasets. Single image inpainting models are not competitive enough for image pairs with larger scale differences and wider baselines. Video inpainting approaches also show bad performance due to the lack of dense temporal information and multiple reference frames. JointDP is based on optical flow and is not able to accurately estimate parameters like the camera pose needed to correctly align the image pairs. Our method demonstrates superiority over TransFill because we have a better understanding of the 3D structures of the scenes, and better leverage the depth estimation. Note that GeoFill has higher performance gain over TransFill on MannequinChallenge than on RealEstate10K dataset. TransFill relies on the fusion module to handle multiple homographies for the final hole filling and it is less robust on the images different from the training data. In contrast, GeoFill only has a single proposal to merge during inpainting, which also greatly reduces blending artifacts that arise from multiple proposals. Therefore, our GeoFill is robust and stable on image pairs with even larger frame differences.

4.4.2 Qualitative Results

Figure 4.3 shows the visual comparisons with other baseline algorithms on the user-provided images, the RealEstate10K and the MannequinChallenge dataset. JointDP utilizes estimated optical flow for the initial matching, thus the results fail to obtain accurate depth and camera pose if the baseline of the image pair is wide. The contents inside the hole are often misaligned with the target image. The original OPN uses five reference frames to achieve a more efficient non-local matching among frames, but a single reference frame makes the results less visually-pleasing. ProFill is not able to take advantage of the reference image contents, and TransFill usually has blending artifacts or content misalignment issues when objects inside the hole regions occupy multiple depth planes. However, GeoFill avoids the blending artifacts by using one single proposal, and aligns the objects well by understanding the camera poses and reconstructing the 3D scene from two images.

4.4.3 Ablation Study

In this section, we first study how different components in the joint optimization step contribute the final results. We demonstrate the importance of the joint optimization module by comparing both inpainting performance and the accuracy of optimized parameters before and after the optimization. Second, we compare the performance of TransFill and GeoFill on cases with larger holes and their alignment accuracy without CST. Additionally, We show the inpainting results of GeoFills under various scenarios, such as different focal lengths and appearance changes from camera movement. We also show a per-sample improvement study and a user study to further analyze the performance gain. Lastly, we provide additional visualizations of GeoFill and some failure cases of the model. All experiment results in the following section are reported for the RealEstate10K FD=50 subset unless specified.

Table 4.3: Ablations on the objective functions in the joint optimization stage of GeoFill.

$\mathcal{L}_{\text{photo}}$	$\mathcal{L}_{\text{feat}}$	$\mathcal{L}_{\text{negD}}$	PSNR↑	SSIM↑	LPIPS↓
✓	✓	✓	**31.47**	**0.9748**	**0.0525**
✗	✓	✓	31.19	0.9742	0.0533
✓	✗	✓	30.88	0.9734	0.0554
✓	✓	✗	31.23	0.9742	0.0532

Table 4.4: Ablations on the pixel importance weight map $\mathbf{W}$.

Hole $\mathbf{W}_h$	Edge $\mathbf{W}_e$	PSNR↑	SSIM↑	LPIPS↓
✓	✓	**31.47**	**0.9748**	**0.0525**
✓	✗	31.20	0.9740	0.0534
✗	✓	31.12	0.9739	0.0539
✗	✗	30.95	0.9734	0.0552

Analysis of objective functions. We claim that all the objective functions used in the optimization process contribute to a higher perceptual and reconstruction quality as shown in Table 4.3. Comparing GeoFill without photometric loss against without feature correspondence loss, the feature loss contributes to the performance the most. We find that photometric loss by itself may get distracted by local textures and fall into local minima, such that it achieves lower RGB errors on average but ignores the global structure. However, using the photometric loss does help improve alignments. Lastly, GeoFill without negative depth penalty still has a performance drop, suggesting negative depth penalty is able to prevent corner cases where depth scale or offset estimates are non-robust.

Analysis of pixel importance weight map. We present results of GeoFill with different combinations of pixel importance weight maps in the optimization in Table 4.4. This suggests that hole-distance weighting $\mathbf{W}_h$ which puts higher weights around the hole lead to better local alignment around the hole. Edge-based weighting $\mathbf{W}_e$ also helped by matching strong edges within the image. Using a uniform weighting map leads to the worst performance, indicating the effectiveness of the weighting maps.

Inpainting performance w/ and w/o the joint optimization. We show the importance of our optimization module by comparing the performance of GeoFill with initial estimated parameters and optimized parameters. As shown in Table 4.5, GeoFill with optimized parameters has substantially better performance. Initial parameters are computed from SIFT and pretrained models such as OANet, which can make erroneous predictions, especially for image pairs with holes. Experimental results demonstrate our optimization module successfully mitigates such errors and improves the inpainting performance.

Depth and pose accuracy before and after the joint optimization. In this section, we further demonstrate the effectiveness of our joint optimization step by measuring the depth and pose accuracy before and after the optimization. Since both RealEstate10K and MannequinChallenge do not have ground-truth labels, we choose the ScanNet [44] dataset which comes with ground-truth camera poses and depth maps. We randomly sampled 75 pairs of images with approximately 30 frame difference. We generate random holes in the same manner as described before. Each image pair comes from a unique scene in the dataset. Note that ScanNet includes images with heavy motion blur, which we manually filtered out. We evaluate depth and relative camera pose *separately* by providing the ground-truth for one of these (depth or camera pose) when evaluating the accuracy of the other one. For example, when evaluating the accuracy of depth maps, we first follow the same pipeline described in this chapter. Then, instead of estimating the relative pose, we provide the ground-truth camera pose and evaluate the accuracy of the depth map determined by our pipeline before and after the optimization. Note that we only optimize scale and offset when evaluating depth accuracy. A similar analogy applies when evaluating the accuracy of camera poses: we provide the ground truth depth map to our pipeline and then evaluate the accuracy of the relative pose before and after optimization. For pose evaluation, we report the geodesic errors [34] for both rotations and translation directions. For depth evaluation, we follow the commonly adopted metrics used in the literature [63, 71]. As shown in Table 4.6 and Table 4.7, both depth and pose errors are significantly reduced after the optimization module,

Table 4.5: Quantitative comparison of our method with initially estimated parameters and optimized parameters.

Model	PSNR↑	SSIM↑	LPIPS↓
GeoFill (optim)	**31.47**	**0.9748**	**0.0525**
GeoFill (init)	30.66	0.9719	0.0548

Table 4.6: Relative camera pose evaluation of initial guess and our optimized results, where R and t represent the rotation and translation, respectively.

	$R \downarrow$		$t \downarrow$	
	mean (°)	med (°)	mean (°)	med (°)
GeoFill (optim)	**1.588**	**1.062**	**3.688**	**3.457**
GeoFill (init)	7.378	7.807	11.861	11.096

which demonstrates the ability of the optimization module to find more accurate depth and poses in our challenging case where the images have holes.

Performance w.r.t hole size. This ablation study aims to examine the performance of our approach against TransFill under more difficult settings. As the hole becomes larger, inevitably we have fewer matching points, which makes $\mathbf{I}_{s \to t}$ hard to align with $\mathbf{I}_t$. We generate holes with different average stroke widths ranging from 90 to 210 pixels. As shown in Figure 4.4, GeoFill has an increasing performance gain over TransFill until the hole average stroke width reaches 180 pixels. As the hole size grows, the complexity of the content in the hole also increases. In other words, we are more likely to encounter a growing number of depth layers, a more complicated objects layout, and a higher chance of occlusions due to camera translation in a larger hole. These problems are harder for homography-based models, therefore, GeoFill has a greater advantage when the hole is larger. The drop in performance gain at the end of the curve can be because there are not enough matching points for GeoFill to infer an accurate 3D structure. We also provide visual comparisons to better understand the performance boost for larger holes. We simulate larger holes by generating the same hole shape with larger stroke width. As shown in Fig. 4.5, GeoFill has a robust performance while TransFill has ghosting artifacts and misalignments as the hole grows larger.

Table 4.7: Depth evaluation of our initial guess and optimized results. The evaluation metrics include absolute relative difference (Abs^r), squared relative difference (Sq^r), root mean squared log error (RMS-log), and accuracy with a relative error threshold of $\delta^k < 1.25^k$, k = 1, 2.

Models	Abs^r ↓	Sq^r ↓	RMS-log↓	δ^1 ↑	δ^2 ↑
GeoFill (optim)	**.258**	**.233**	**.302**	**.683**	**.851**
GeoFill (init)	.372	.766	.403	.609	.788

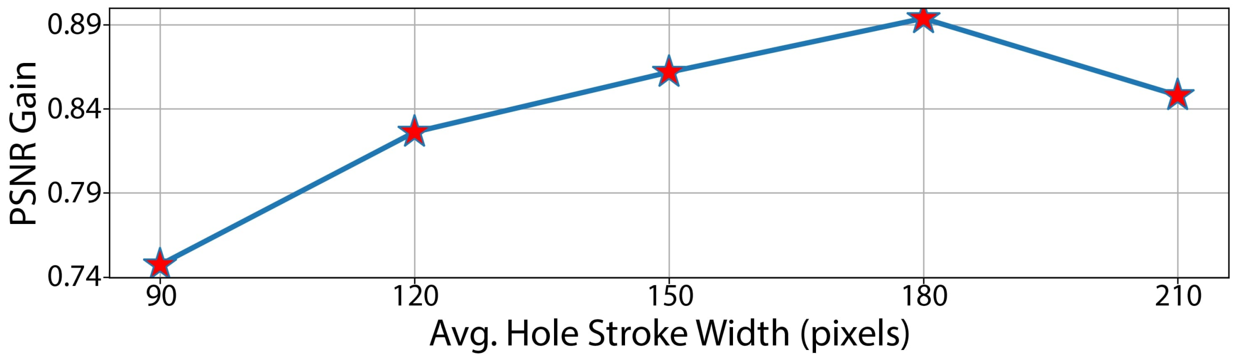


Figure 4.4: Performance gain of our method compared to TransFill w.r.t the average hole size. GeoFill has a greater advantage when the hole is larger.

Initial alignment comparisons against TransFill. We adopted the CST module from TransFill to adjust auto exposure, lighting conditions, and potential remaining misalignments. In this experiment, we analyze the performance of using the aligned images directly, e.g. without using CST for both models. We retain the multiple homographies used by TransFill, drop the CST module, and keep the merging module so TransFill can merge its different regions. Table 4.8 shows that GeoFill maintains better performance, dropping 2.12 in PSNR, while TransFill drops 4.62 without CST. We visually compare the quality of our single proposal to the merged proposal from TransFill without the CST on RealEstate10K dataset. As shown in Fig. 4.6, the single proposal from GeoFill is significantly more accurate than merged heuristic proposals from TransFill, demonstrating the superiority of our approach over TransFill. Additionally, we also show the quantitative comparisons of GeoFill and TransFill without the CST module on the MannequinChallenge dataset. As shown in Table 4.9, we find GeoFill without the CST module has a huge advantage over TransFill merged homographies. This experimentally validates the proposal from GeoFill is much more accurate than TransFill

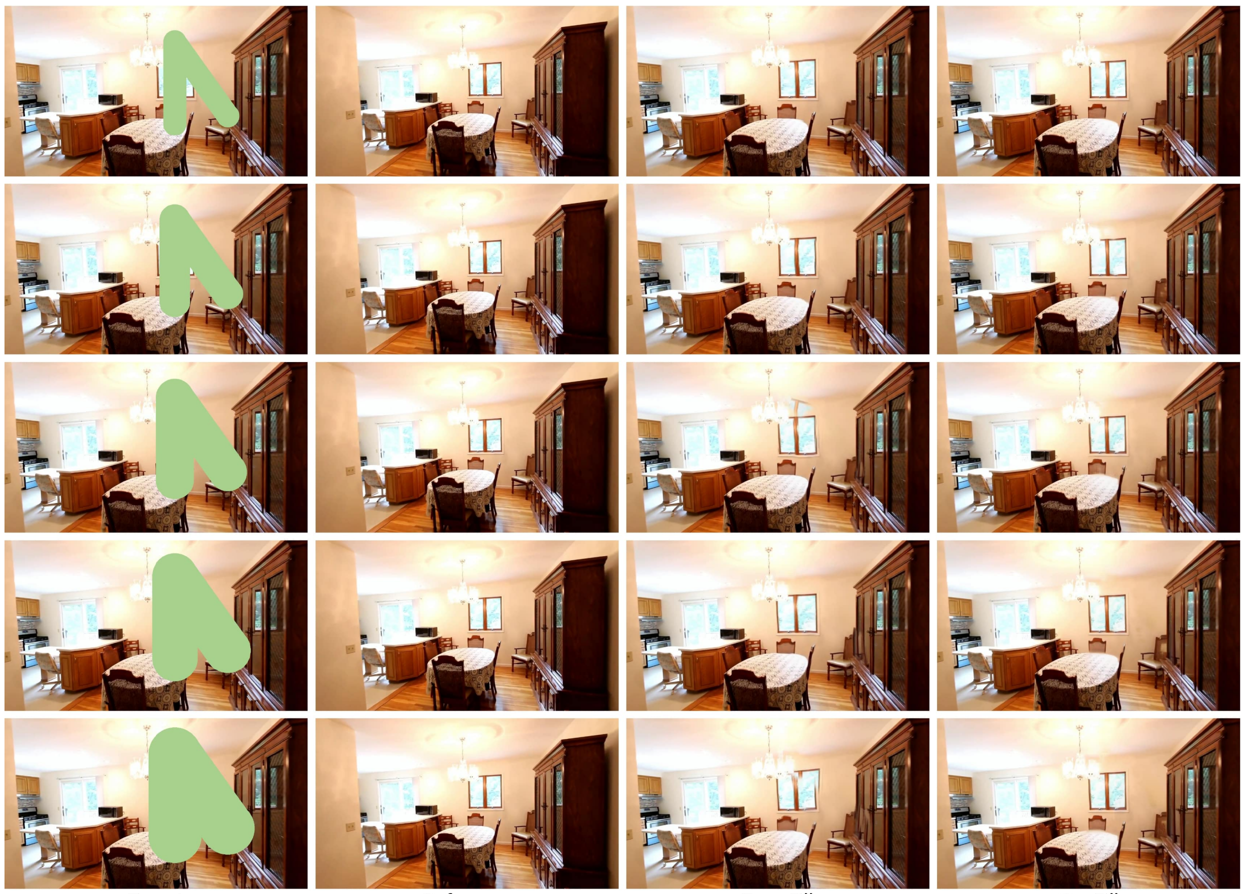


Figure 4.5: Qualitative comparisons of our approach against TransFill with different hole sizes. **Please zoom in** to see that ours looks good but there are broken structures, ghosting, and distortion artifacts in TransFill.

by better leveraging the depth.

Performance w.r.t intrinsic parameters. GeoFill handles incoming image pairs using fixed camera intrinsic parameters instead of explicitly knowing the ground-truth camera intrinsic parameters. In the previous experiments, we use fixed camera intrinsic parameters by setting the focal length of all images to 750 pixels and the principal point to the center of the image. It is intuitive to set the principal point to the center of the images with unknown intrinsic parameters, therefore, we focus on studying the effect of focal lengths. We compare the performance of GeoFill with the camera focal lengths of 600, 750, 900, 1050, and 1200 pixels. As shown in Fig. 4.7, GeoFill with different focal lengths has very slight differences

Table 4.8: Ablations on Initial alignment comparisons of our method compared to TransFill without the CST Module.

Model	PSNR↑	SSIM↑	LPIPS↓
GeoFill	**31.47**	**0.9748**	**0.0525**
GeoFill (no CST)	29.35	0.9688	0.0579
TransFill	30.64	0.9732	0.0540
TransFill (no CST)	26.03	0.9598	0.0742

Table 4.9: Initial alignment comparisons of our method compared to TransFill without the CST Module on MannequinChallenge dataset (FD=10).

Model	PSNR↑	SSIM↑	LPIPS↓
GeoFill	**28.85**	**0.9702**	**0.0553**
GeoFill (no CST)	26.84	0.9626	0.0615
TransFill	28.01	0.9680	0.0569
TransFill (no CST)	24.04	0.9526	0.0760

in terms of PSNR. There is a slight trend that the performance drops as the focal length increases. We believe this indicates that the ground-truth focal length is close to 600 and higher focal lengths make the optimization have a harder time finding improved relative poses. Nevertheless, GeoFill can still adapt to different focal lengths by jointly optimizing depth scale, offset, and relative pose, therefore, it still can render similar images across a variety of focal lengths.

Handling appearance changes from camera movement. As we stated in the previous sections, we focus on the common scenario of capturing photos with the same camera *freely* moving around. However, there are potential appearance changes of the same parts of the scene due to the camera movement, for example, changes due to automatic exposure or automatic white balance between source and target images. This is a common yet non-trivial challenge when applying GeoFill in real-world applications. In this section, we show some visual examples of image pairs with appearance changes in the dataset. As shown in Figure 4.8, GeoFill still inpaints plausible results even when the appearance of the same part of the scene is different between source and target images.

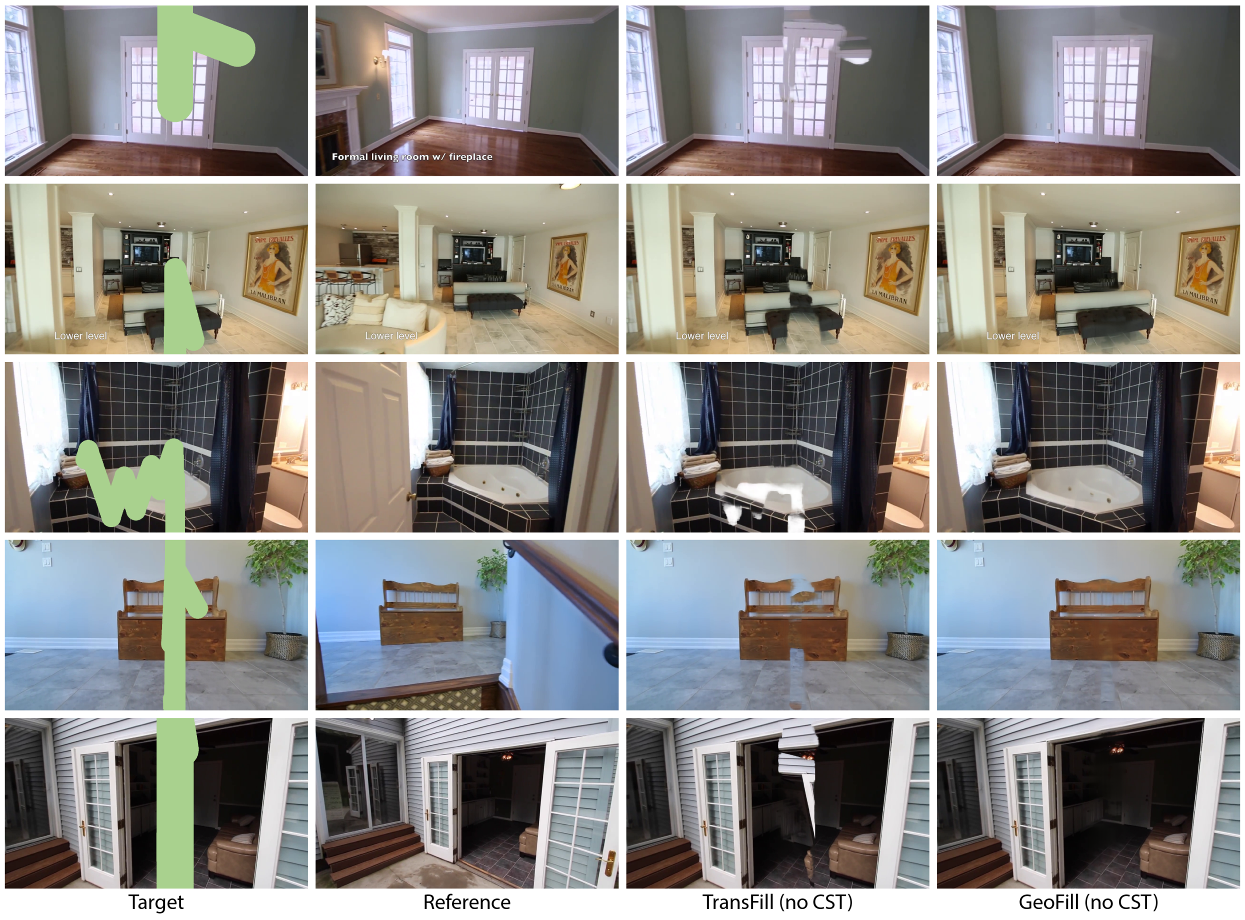


Figure 4.6: Visual comparisons of GeoFill and TransFill without the CST module.

Per-sample improvement study. This ablation is designed to break down the numbers shown in Table 4.1 and 4.2 by comparing the performance of GeoFill against TransFill over each individual sample. We sort and plot the per-image PSNR difference over TransFill, shown in Figure 4.9. Specifically, PSNR difference is computed as GeoFill PSNR − TransFill PSNR, therefore a positive PSNR gain indicates GeoFill is better. GeoFill improves performance on the majority of samples, specifically for around 75.4% of the entire subset. Moreover, the PSNR gains when GeoFill outperforms are much greater: up to 3 dB, as opposed to the PSNR losses when TransFill outperforms.

User study. To better evaluate the performance of GeoFill against other baselines, we conduct a user study via Amazon Mechanical Turk (AMT). We compare our method against OPN, ProFill, and TransFill by showing users image pairs with binary choice questions. The

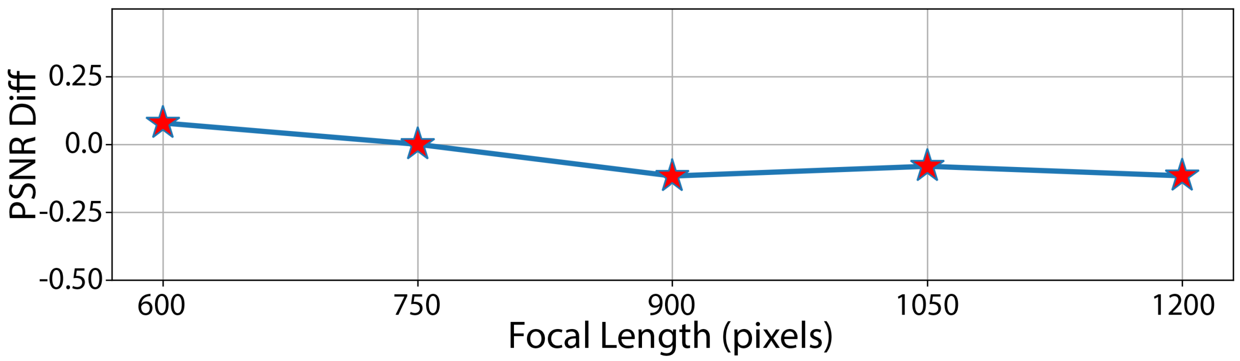


Figure 4.7: Visual plots showing the performance of GeoFill with different focal lengths. PSNR diff is computed by using GeoFill with new focal length subtract GeoFill with focal length equals to 750.

Table 4.10: User study results of GeoFill against ProFill, OPN, and TransFill.

	Filtered		Non-Filtered	
Model	PR	p-value	PR	p-value
ProFill	100%	$p < 10^{-6}$	96.25%	$p < 10^{-6}$
OPN	97.37%	$p < 10^{-6}$	95.00%	$p < 10^{-6}$
TransFill	70.90%	$p < 2 \times 10^{-3}$	68.13%	$p < 2 \times 10^{-3}$

users are requested to choose the inpainting results that look more realistic and faithful. To improve the quality of collected data, we adopt a qualification test with trivial questions to filter noisy results. For each method pair, we randomly sampled 80 examples in RealEstate10K dataset with FD=50, and each example was evaluated by 7 independent users. We present two approaches to computing the preference rate. The first one is the filtered approach, in which we filter the responses to retain only those where one method is “preferred" if 6 or more users select it. The filtering helps suppress noise in the responses of Mechanical Turk workers, whose work quality can vary. The second one is the non-filtered approach where we retain all responses and choose the method as “preferred" where a simple majority of 4 or more users select it. We reported GeoFill’s Preference Rate (PR) in Table 4.10. GeoFill has much higher preference rates against OPN and ProFill. Compared against TransFill, we receive a PR around 70% on filtered and non-filtered approaches. TransFill is still very robust on small holes and relatively small camera motions in the randomly sampled data. Therefore, GeoFill is favored by users over TransFill but less strongly than in the other comparisons.

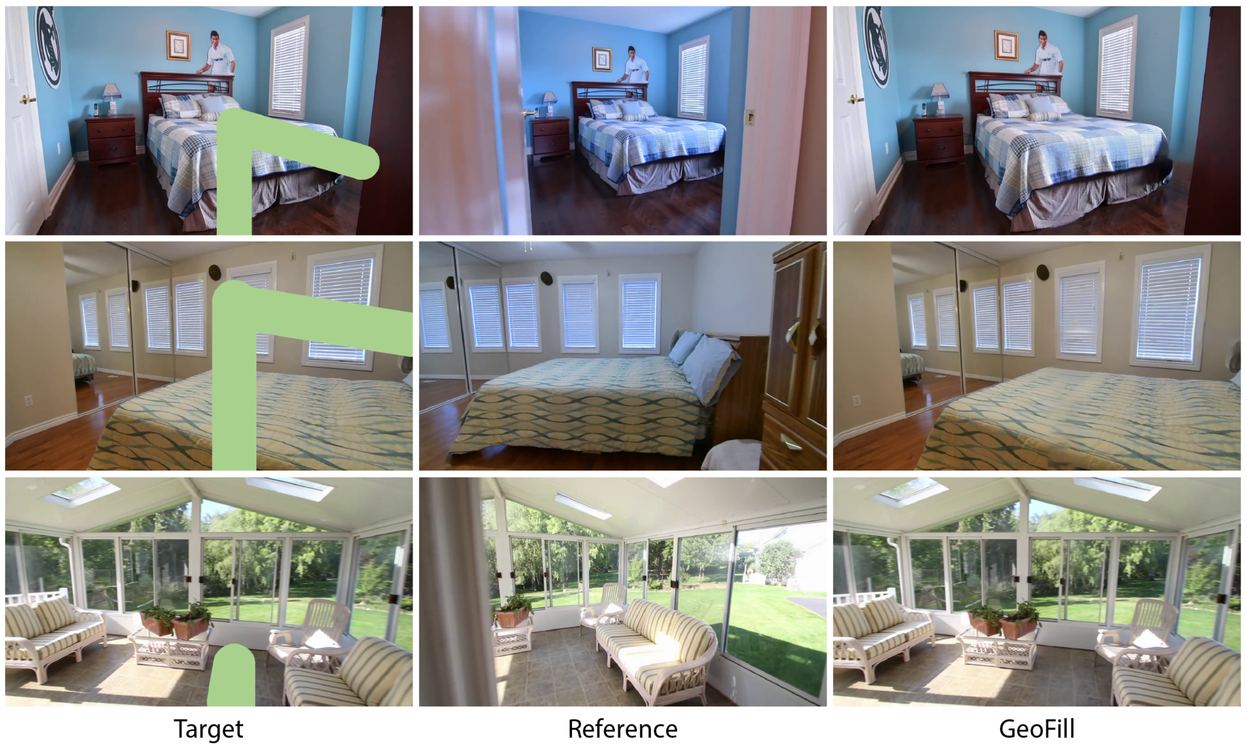


Figure 4.8: Qualitative results of GeoFill handling some common appearance changes such as in white balance and exposure due to camera movement.

We performed a one sample permutation t test with 10^6 simulations using the null hypothesis that each pair are preferred equally by users: the p-values are all sufficiently small that the preference for our method is statistically significant.

Failure cases. We show some failure cases of GeoFill under extreme conditions. Fig. 4.10 shows three common failure cases of GeoFill. The image pair on the left contains transparent surfaces in the images. These objects often cause monocular depth estimators to fail and can lead to bad optimization results. In the second failure case, the drastic changes in the lighting environment affect the feature correspondence matching and depth prediction, which makes the final result from GeoFill less accurate. In the last case, dynamic objects, e.g., pedestrians, make our optimization module estimate inaccurate parameters. We discuss in the last section of this chapter ways that future work might address these issues.

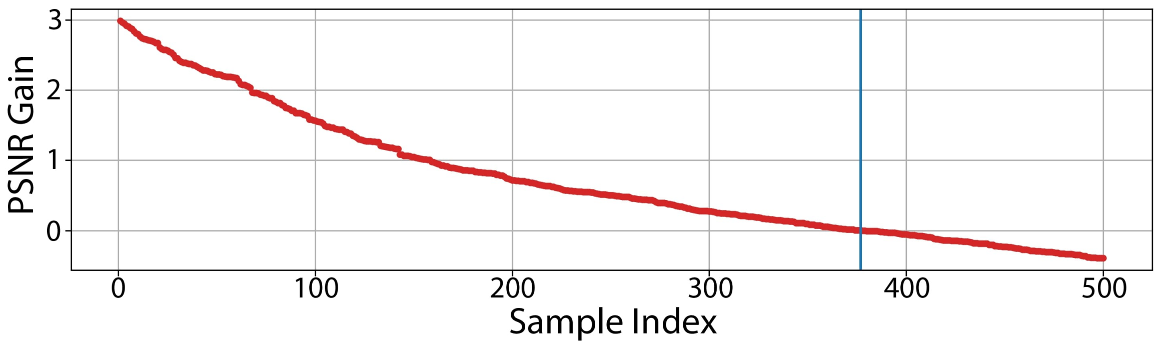


Figure 4.9: Per-sample performance gain of ours compared to TransFill. The blue vertical line separates positive and negative PSNR gain.

Additional visual results. We include additional qualitative comparisons of GeoFill against other baselines in Fig. 4.11. Additionally, we also show the inpainting performance of GeoFill on user-provided images, RealEstate10K, and MannequinChallenge dataset in Fig. 4.12.

4.5 Discussion, Limitations, and Conclusion

Limitations and Future Work. GeoFill inpaints the hole with one image pair with no auxiliary pose or depth information from sensors, therefore, our pipeline may not work well when the quality of feature matching points is poor, *e.g.*, matching points are inaccurate or too few. Under these cases, our relative pose and triangulated points can be inaccurate, which may be hard for optimization to correct. Additionally, our pipeline is also sensitive to depth prediction quality: artifacts such as blurry depth discontinuities or wrong order of depth planes can lead to potential bad inpainting results. Future work might mitigate these problems by jointly reasoning about monocular depth and the stereo cues established by triangulation. In the optimization, we used a 3D reprojection based on forward warping because it is much faster than rendering a triangle mesh even though it does not fully remove hidden surfaces in the rare cases where a mesh occludes itself: this could be addressed in future work by testing and pruning those splatted points. GeoFill utilizes the CST module to adjust auto exposure and lighting condition changes, which still suffers when the scene environment changes drastically, *e.g.*, day to night, spring to fall. Future work could better address these by incorporating specialized lighting estimation (*e.g.*, [277]) and relighting modules. One last limitation of GeoFill is that our pipeline only handles static scenes. Any dynamic objects, *e.g.*, walking people or moving cars, not in the hole could lead to bad relative pose estimation and our optimization may cause misalignments under such cases. A simple solution would be masking out objects that are likely to move such as people and cars, but how to correctly identify all moving objects is still an open question.

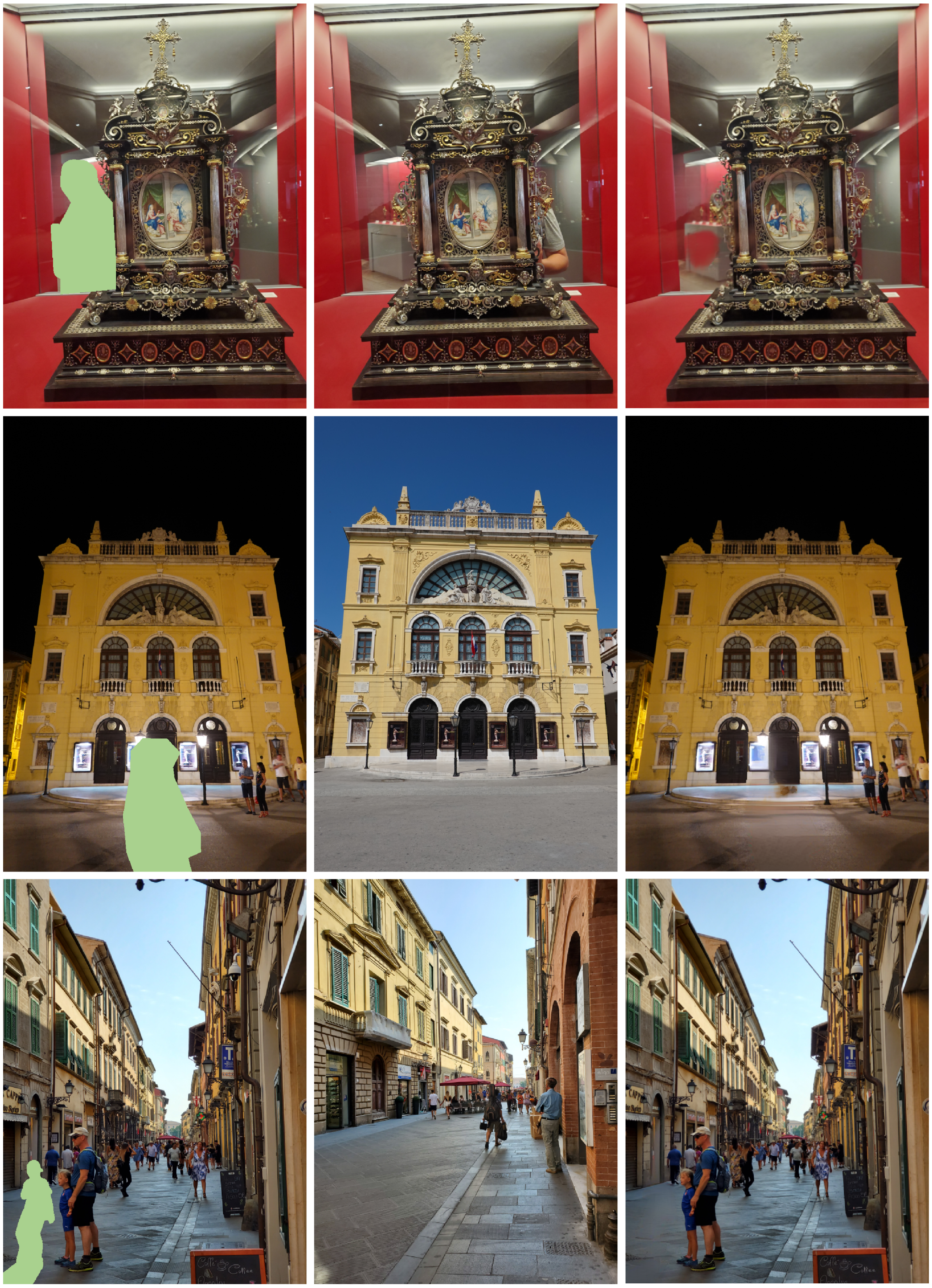


Figure 4.10: Visual examples of failure cases of GeoFill.

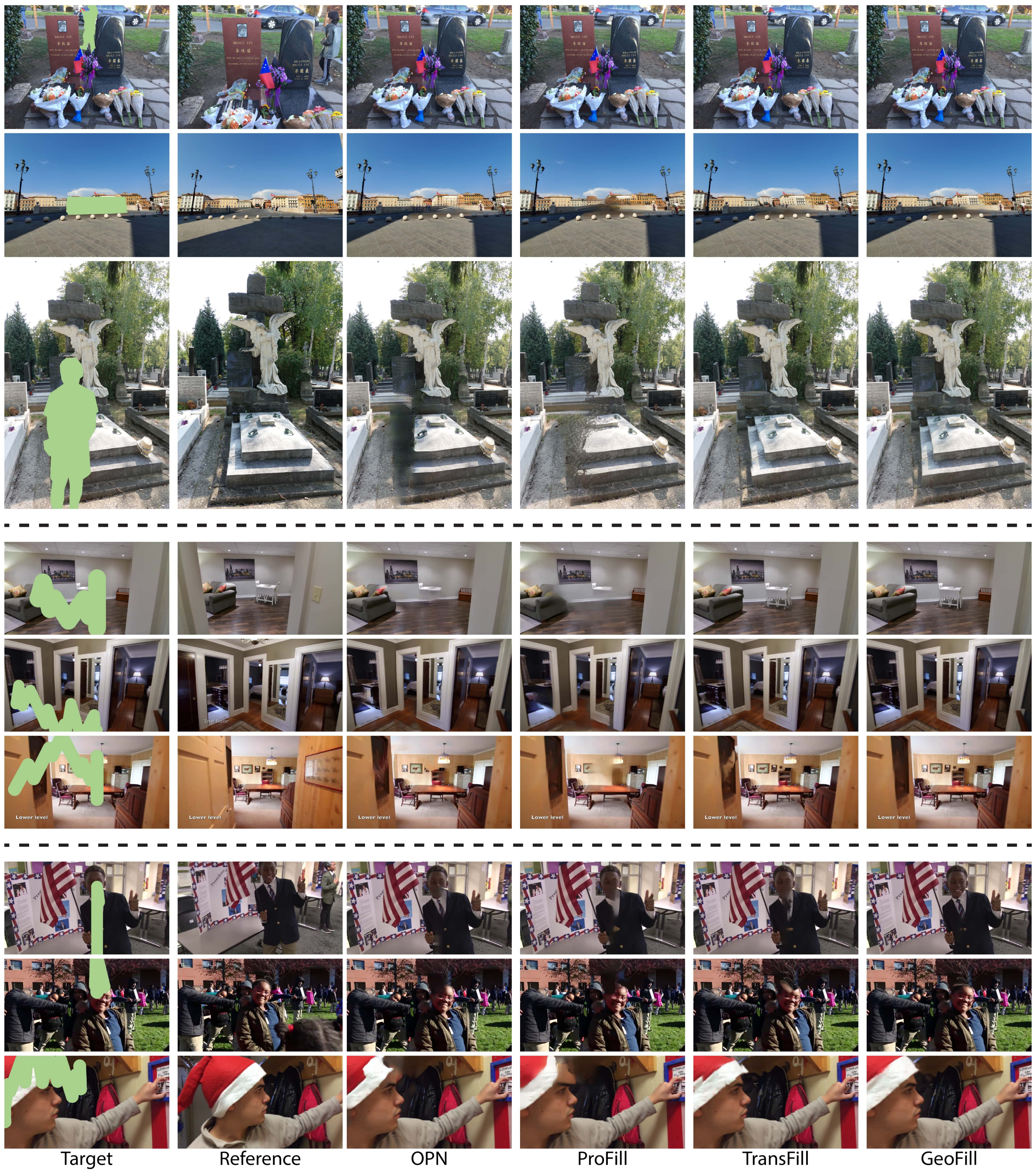


Figure 4.11: Qualitatively comparison of GeoFill against other baselines on user-provided images (top 3 rows), RealEstate10K (mid 3 rows), and MannequinChallenge dataset (last 3 rows).

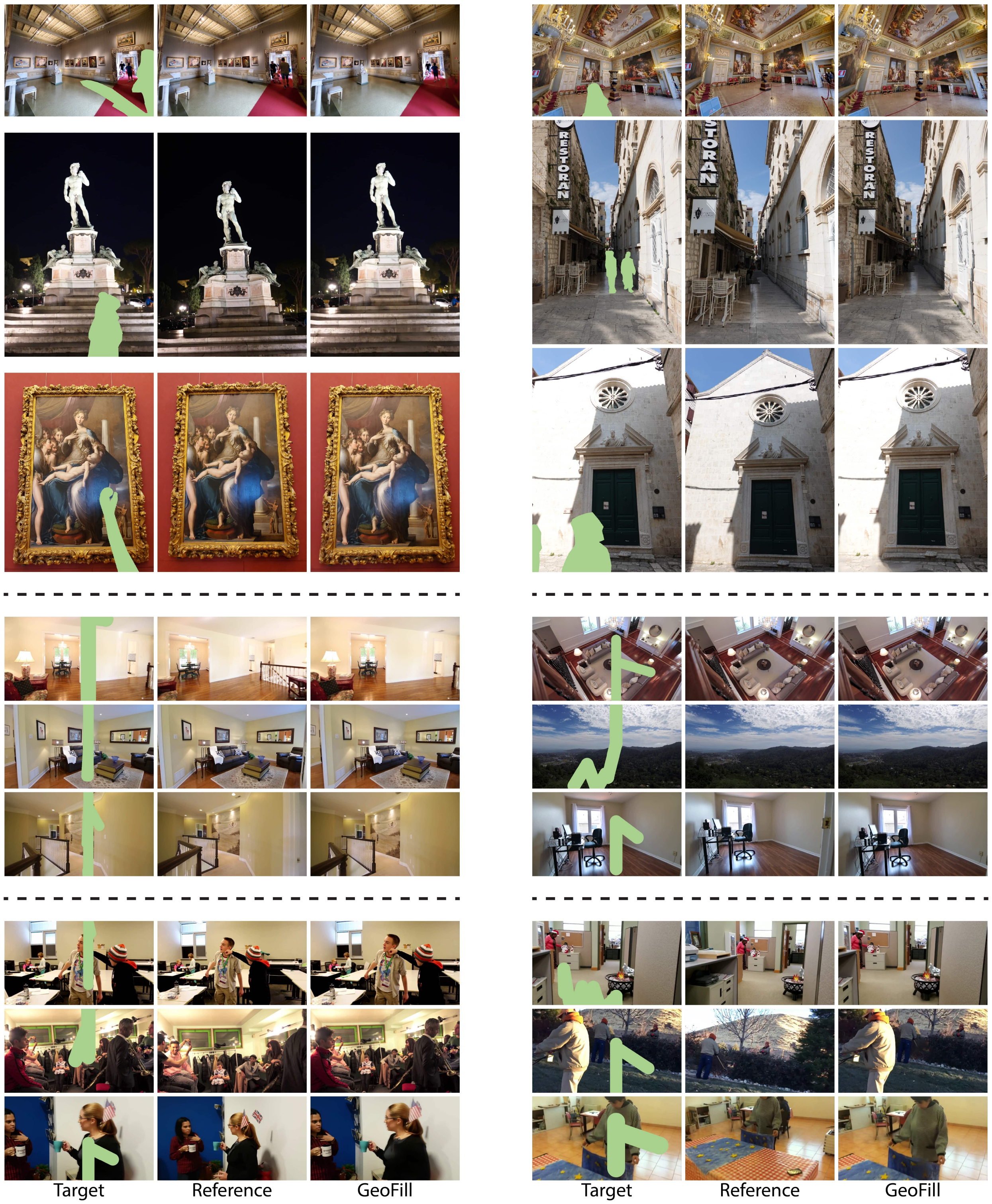


Figure 4.12: Visual illustration of inpaiting performance of GeoFill on user-provided images, RealEstate10K, and MannequinChallenge dataset.

Chapter 5

Instance tracking in 3D scenes with RGBD egocentric videos

Egocentric sensors such as AR/VR devices capture human-object interactions and offer the potential to provide task-assistance by recalling 3D locations of objects of interest in the surrounding environment. This capability requires instance tracking in real-world 3D scenes from egocentric videos (IT3DEgo). We explore this problem by first introducing a new benchmark dataset, consisting of RGB and depth videos, per-frame camera pose, and instance-level annotations in both 2D camera and 3D world coordinates. We present an evaluation protocol which evaluates tracking performance in 3D coordinates with two settings for enrolling instances to track: (1) single-view online enrollment where an instance is specified on-the-fly based on the human wearer's interactions. and (2) multi-view pre-enrollment where images of an instance to be tracked are stored in memory ahead of time. To address IT3DEgo, we first re-purpose methods from relevant areas, e.g., single object tracking (SOT) — running SOT methods to track instances in 2D frames and lifting them to 3D using camera pose and depth. We also present a simple method that leverages pretrained segmentation and detection models to generate proposals from RGB frames and match proposals with enrolled

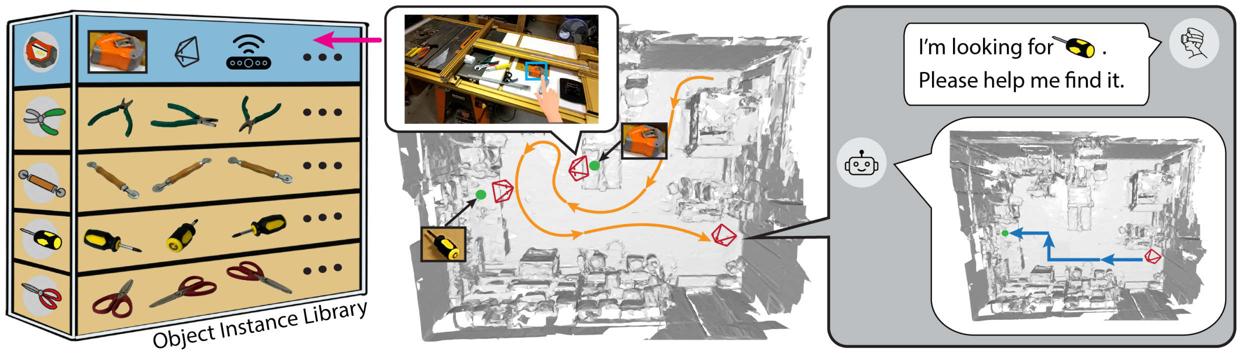


Figure 5.1: **Motivation for the proposed IT3DEgo benchmark task**. We envision the real-world application of an assistive agent that continuously tracks enrolled object instances in 3D and can provide navigation guidance to users to retrieve object instances at any time. Tracked objects are either enrolled online (first row in the library) where objects of interest are identified automatically based on user interactions or pre-enrolled (bottom four rows in the library), where task-relevant objects are modeled from a collection of photos taken from different views. The former setup comes with additional in-context sensor information, such as camera pose and depth while the latter features richer visual information.

instance images. Our experiments show that our method (with no finetuning) significantly outperforms SOT-based approaches in the egocentric setting. We conclude by arguing that the problem of egocentric instance tracking is made easier by leveraging camera pose and using a 3D allocentric (world) coordinate representation.

5.1 Background

Egocentric video obtained from AR/VR devices provides a unique perspective that captures the interaction between the human wearer and the surrounding 3D environment. With the rapid development of AR/VR hardware, there is increasing interest in building assistive agents [170, 226, 252, 212], that track the user's environment and provide contextual guidance on the location of objects of interest (illustrated in Figure 5.1). We argue that developing such an agent requires solving the largely unexplored problem of *tracking object instances in 3D* from egocentric video.

Why this problem? First, tracking in egocentric video is a novel and underexplored problem, compared to the well-studied tracking from fixed, third-person viewpoints. More broadly, egocentric visual understanding tasks, such as human pose estimation and trajectory prediction [20, 242, 60] are a growing area of interest. Second, tracking in 3D scenes is essential in robotics, autonomous driving, and AR/VR applications. Compared to the 2D counterpart, tracking objects in 3D is crucial for an agent to not only understand the surrounding 3D environment but also to determine precise locations for planning and navigation. Combining the two perspectives above, there is a broader question of what information processing constraints govern how the human visual system integrates egocentric sensory data into a seemingly allocentric perception of the world around us.

Challenges and new opportunities. (1) Egocentric video often features motion blur, hand occlusions, and frequent object disappearances and reappearances which make the 2D tracking problem very challenging from pure visual signals [60, 227]. Tracking in 3D offers an opportunity to fuse additional sensor streams, such as depth and camera pose, to improve accuracy. Unlike 2D tracking with a moving camera, 3D tracking in world coordinates allows the model to leverage the unique prior information – *an object should remain still unless being interacted with the human operator*. (2) For the downstream application of task guidance, we propose exploring novel approaches to identify or *enroll* object instances to be tracked. One approach is automatically enrolling objects with which the user interacts or identifies via hand gestures such as pointing. Alternatively, object instances relevant to a particular task could be *pre-enrolled* based on a collection of images that specify the visual appearance of the object in advance.

Contribution 1: Dataset collection. To our best knowledge, no existing dataset supports exploring the problem of IT3DEgo (c.f. Table 5.1). The recent Ego4D dataset [88] highlights some of these challenges. However, the Ego4D dataset only provides RGB frames[1] and

[1]Ego4D does provide a sparse set of camera poses (less than 15% of frames) estimated with COLMAP and predicted depth maps using monocular depth estimation.

sparse annotations (may miss potential object location changes), making it unsuitable to fully explore the problem. We collect a new benchmark dataset with HoloLens2, including an RGB camera, a depth sensor, four grayscale cameras, per-frame camera pose and coarse scene geometry as a mesh. We describe the details of dataset statistics, capture procedures, and annotations in Section 5.3.2.

Contribution 2: Benchmarking protocol. We propose a new IT3DEgo Benchmark for studying instance tracking in 3D scenes from egocentric videos with two settings for how objects are selected for tracking. (1) *Tracking with single-view online enrollment (SVOE)* studies the scenario where object instances of interest are defined on-the-fly, i.e., objects are specified with a 2D bounding box in the frame where they *first* become fully visible to the user. (2) *Tracking with multi-view pre-enrollment (MVPE)* assumes objects of interest are specified by multiple photos of the object of interest from different viewpoints before the tracking system starts. As detailed in Section 5.3.1, we evaluate performance with standard precision/recall metrics as well as geometric L2 and angular errors used in the Ego4D VQ3D evaluation [88].

Contribution 3: Technical explorations. Since our benchmark task is novel and underexplored in the literature, it is natural to re-purpose and evaluate existing approaches (e.g., SOT methods). We also explore an alternative *piecewise constant velocity* method that utilizes the Kalman filter [112] with instance proposals from SAM [118] and encoded by DINOv2 [175], resulting in drastic performance improvement over state-of-the-art SOT methods. Section 5.4 and 5.5 provide details regarding baselines and benchmark results, respectively. From the experimental results, we provide the following insights: Tracking object instances in egocentric videos is easier in 3D scenes leveraging camera poses and depth maps. Intuitively, an object not being interacted with has the same 3D position in a predefined world coordinate but the positions in 2D frames can change drastically due to the head motion. As a result, existing state-of-the-art 2D SOT approaches perform poorly

Table 5.1: **Comparisons of egocentric datasets that explore tracking-related problem.** Existing egocentric datasets only explore the tracking problem in 2D or predicting discrete 3D locations. Some mention the tracking problem in 3D but only consider limited sensor data (RGB) or synthetic environments. Our benchmark dataset supports the study of instance tracking in 3D real-world scenarios (RWS in the table) from egocentric videos.

Dataset	Modality	Device	Avg. Length	Annot. FPS	RWS	Camera Trajectory	3D Tracking	Year
TREK-150 [59]	RGB	GoPro	10s	60	✓	Natural	✗	2021
EK-VISOR [45]	RGB	GoPro	12s	0.9	✓	Natural	✗	2022
Ego4D-VQ3D [88]	RGB	GoPro	-	-	✓	Natural	✗	2022
EMQA [48]	RGB-D+IMU	-	-	-	✗	Simulated	✗	2022
EgoPAT3D [142]	RGB-D+IMU	Kinect	4min	30	✗	Object-Centric	✗	2022
DigitalTwin [178]	RGB-D+IMU	Aria	2min	-	✗	Natural	✓	2022
EgoTrack [227]	RGB	GoPro	6min	5	✓	Natural	✗	2022
Ours	RGB-D+IMU	HoloLens	>5min	6	✓	Natural	✓	2023

on egocentric data. Future work should address the problem of re-identifying objects by leveraging the camera poses and accurately identifying and updating object motion changes.

5.2 Related Work

Egocentric video datasets have been developed to study different problems over the last decade [135, 185, 67, 223, 45, 88]. Traditionally, egocentric video understanding has focused on tasks such as activity recognition [199, 187, 113, 186], human-object interactions [46, 150, 151], and inferring the camera wearer's body pose [196, 111, 169, 242]. Recently, more tasks have emerged due to the increasing interest in egocentric videos, such as action anticipation [195, 72, 68], privacy protection [198, 53, 229], and estimating social interactions [268, 139, 173]. However, object tracking in egocentric videos is largely underexplored in the literature until the introduction of recent datasets[60, 227]. These existing tracking datasets only support 2D tracking, which motivates us to collect and setup a new benchmark to evaluate real-world 3D instance tracking.

Tracking in 3D scenes aims to identify objects of interest in 3D space from a sequence of frames. The prediction output format depends on the downstream tasks, including 3D bounding boxes [115, 247], 3D object centers [298, 266], or 6DOF poses [76, 4]. State-of-the art 3D tracking models [295, 153, 37] have focused on well-established third-person perspective benchmark datasets [79, 29, 44]. The recent large-scale Ego4D dataset starts to address the problem of querying the 3D positions of objects from a first-person perspective. However, the raw sensor data in Ego4D only includes RGB images and no other 3D information, such as depth and camera poses [288, 287]. However, contemporary AR/VR headsets come with additional cameras, depth, and IMU sensors that allow for richer geometric reasoning [236, 178]. Therefore, we believe it is realistic to leverage diverse sensor streams and explore the egocentric tracking problem in 3D. Our benchmark dataset thus includes multiple raw sensors and derived data streams to support the study tracking in 3D scenes with modern hardware platforms.

Object instance detection and tracking is a long-standing problem in computer vision and robotics [81, 61, 103, 241, 210, 208]. Instead of predicting labels from a predefined set of object categories, instance-level predictions treat every object instance as a separate category. Instance-level tracking aims to locate given object instances in a sequence of frames, commonly using a tracking-by-detection paradigm. One common formulation is person re-identification [293, 262], which aims to track and associate individual people as they enter and leave multiple cameras' fields of view. Our setting is closely related but is dominated by the motion of the (egocentric) camera rather than the dynamics of object motion.

5.3 IT3DEgo: Protocol and Dataset

The problem of IT3DEgo is motivated by real-world assistive agents running on AR/VR devices. Given an object instance specified by the end user, developed models are required to track it in the 3D environment, i.e., recording its 3D location over time (cf. Fig. 5.2). In this section, we introduce our benchmarking protocol and dataset.

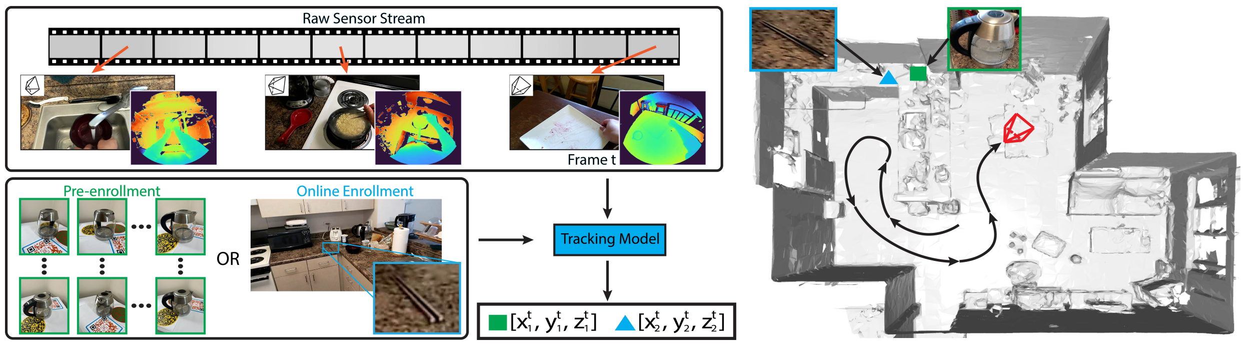


Figure 5.2: **Illustration of input and output of our benchmark task**. Given a raw RGB-D video sequence with camera poses and object instances of interest, i.e., either by online enrollment (SVOE) or pre-enrollment (MVPE), the goal of our benchmark task is to output the object instance 3D centers in a predefined world coordinate at each timestamp. Please check Section 5.3.1 for more details.

5.3.1 Benchmarking Protocol

Because object instances of interest are naturally diverse and may fall outside of the vocabulary of existing detectors, we set up a benchmarking protocol that focuses on evaluation without a separate training set. In other words, models should be pretrained on other data sources and cannot see objects in our dataset. This aligns with the contemporary foundation models (e.g., CLIP [188] and SAM [118]) pretrained on open-world data.

Instances enrollment. We consider two distinct setups to specify object instances of interest. The first is *single-view online enrollment (SVOE)*, similar to single object tracking (SOT) where an object is specified on-the-fly by the end users. For example, the user can specify an object of interest by interacting or pointing to it, after which the system should track it in the 3D world. The second is *multi-view pre-enrollment (MVPE)*, which defines (or pre-enrolls) concerned objects with a set of object-centric images captured from multiple angles. The two setups present different challenges. SVOE provides a bounding box of the object (similar to specifying an object in SOT), but the visual quality is generally lower in resolution as the objects can be far from the camera. MVPE provides 25 high-resolution (2124×2832) object-centric images of the instances captured from different angles. However, the object instance is captured under different lighting conditions than the tracking environment, and can be posed differently (e.g., keys can be deformed over time).

Evaluation protocols. Following the literature on object tracking and detection, we use the metrics below in our benchmarking protocol.

- **Precision and recall** at different L2 distance thresholds. Given N specific thresholds τ_i with $i \in \{1, 2, ..N\}$, specifically 0.25, 0.5, 0.75, 1.0, and 1.5 meters, a ground-truth object location $\mathbf{o_{gt}} \in \mathbb{R}^3$ and a predicted location $\mathbf{o_{pred}} \in \mathbb{R}^3$, we count a true positive (TP_i) when $||\mathbf{o_{gt}} - \mathbf{o_{pred}}||_2 \leq \tau_i$. At each timestamp, each ground-truth is matched to the prediction with the smallest L2 distance below the threshold. Unmatched predictions and ground-truth at threshold τ_i are counted as false positives (FP_i) and false negatives (FN_i), respectively. TP_i, FP_i, and FN_i are computed over all object instances in every frame. The precision and recall at threshold τ_i is computed as $\sum \mathrm{TP}_i$ / ($\sum \mathrm{TP}_i$ + $\sum \mathrm{FP}_i$) and $\sum \mathrm{TP}_i$ / ($\sum \mathrm{TP}_i$ + $\sum \mathrm{FN}_i$), respectively [189, 265].
- **L2 and angular error.** Following VQ3D in Ego4D [88], we also compute the L2 distance between the ground-truth and predictions in the world coordinates in meters. We also report the angular error in radians in the current camera coordinate system. Unlike threshold-aware 3D precision and recall, these metrics are computed only on frames where both ground-truth and prediction of the object instance location are available.

To make 3D annotation tractable, we only evaluate predictions during time intervals when target objects are stationary (i.e., not being handled by the camera wearer).

5.3.2 Dataset

We present additional details of the datasets, such as collection details and annotations, to help others better understand and utilize the benchmark dataset. Note that the data collection protocol was registered with the appropriate institutional review board (IRB).

Raw video collection. The raw IT3DEgo data was recorded by three individuals in ten diverse indoor scenes, e.g., kitchen, garage, office, labs, etc. The participants perform naturalistic tasks with different object instances in the scene, e.g., cooking, repairing, writing, etc. The raw data includes 50 recordings in total. Each recording contains five or more object instances, each of which appears at three different 3D locations on average. The average length of each recording is 10K frames or >5min. We capture the raw data with HoloLens2 which includes an RGB camera, four grayscale cameras and a depth sensor operating in 2 different modes, shown in Figure 5.3. Considering the downstream application scenarios of our benchmark task, we choose to capture our benchmark dataset in 10 different indoor scenes. To capture the real-time geometry information, we capture all videos with high fps AHAT depth mode in HoloLens2 [236]. Note that AHAT depth maps come with phase wrapping [91] at 1 meter but they can be unwrapped using rendered depth from mesh or exploring existing unwrapping algorithms [57, 56]. Before capturing in a new environment, we have a warm-up phase to make the device familiar with the surrounding environment in order to output accurate camera poses when capturing the video. In the warm-up phase, we walk around in the environment with the HoloLens2 turned on and make sure the device has seen all visible surfaces. In practice, we spend around 20 minutes for the warm-up phase when we move to a new environment and around 5 minutes every time before we capture the new video. The resolutions of RBG, grayscale and depth sensors are 720×1280, 480×640, 512×512, respectively. Raw sensors operate at different frequencies, we sync all other sensors to the frequency of the RGB camera (30 fps). We also provide a coarse resolution scene mesh of each environment reconstructed by the Hololense OS. We include additional 2D and 3D

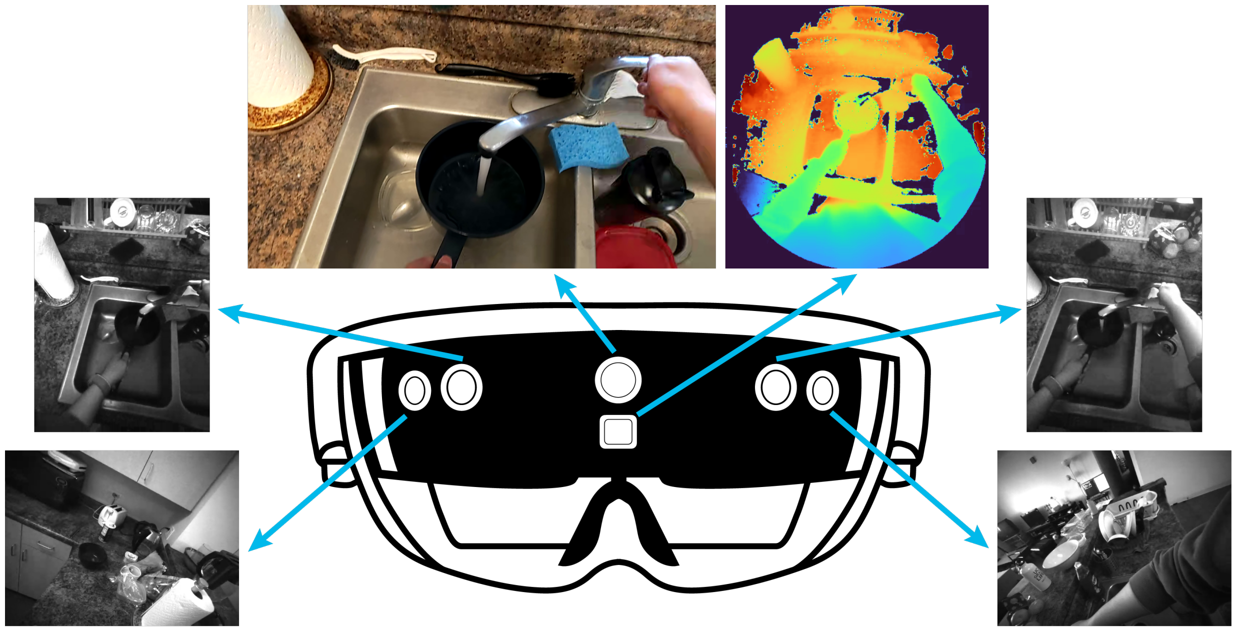


Figure 5.3: **Illustration of our benchmark dataset.** It is collected with HoloLens2 which captures RGB, depth, and four grayscale side views at 30 fps. Additionally, the device also captures per-frame camera poses allowing coarse reconstruction of the surroundings.

visualizations of our benchmark dataset in Figure 5.11.

Object instance collection. The entire videos come with 220 unique object instances, which cover a wide range of object instances for naturalistic daily tasks, such as cooking, writing, and repairing. To support the SVOE setup, annotators identify the first RGB frame where a given object is fully visible and close enough to specify a 2D bounding box which is at least 500 pixels in area. For MVPE, we take 25 high-resolution images on a rotary table with the QR code (c.f. Figure 5.4 for visual examples). Specifically, the photos are taken by hand-held iPhone 13 Pro approximately 45 cm away from the object center. Each object was placed on a rotary table with QR codes. As illustrated in Figure 5.5, we took 12 photos of each object evenly from 360° while keeping the camera at about 30° elevation, 12 more at 60° elevation and 1 top-down view. We zoom in 2.5 times for objects whose diameter is lower than 20 cm to ensure the object instance is large enough in the image and use the normal scale (no zoom) for the rest of the case.

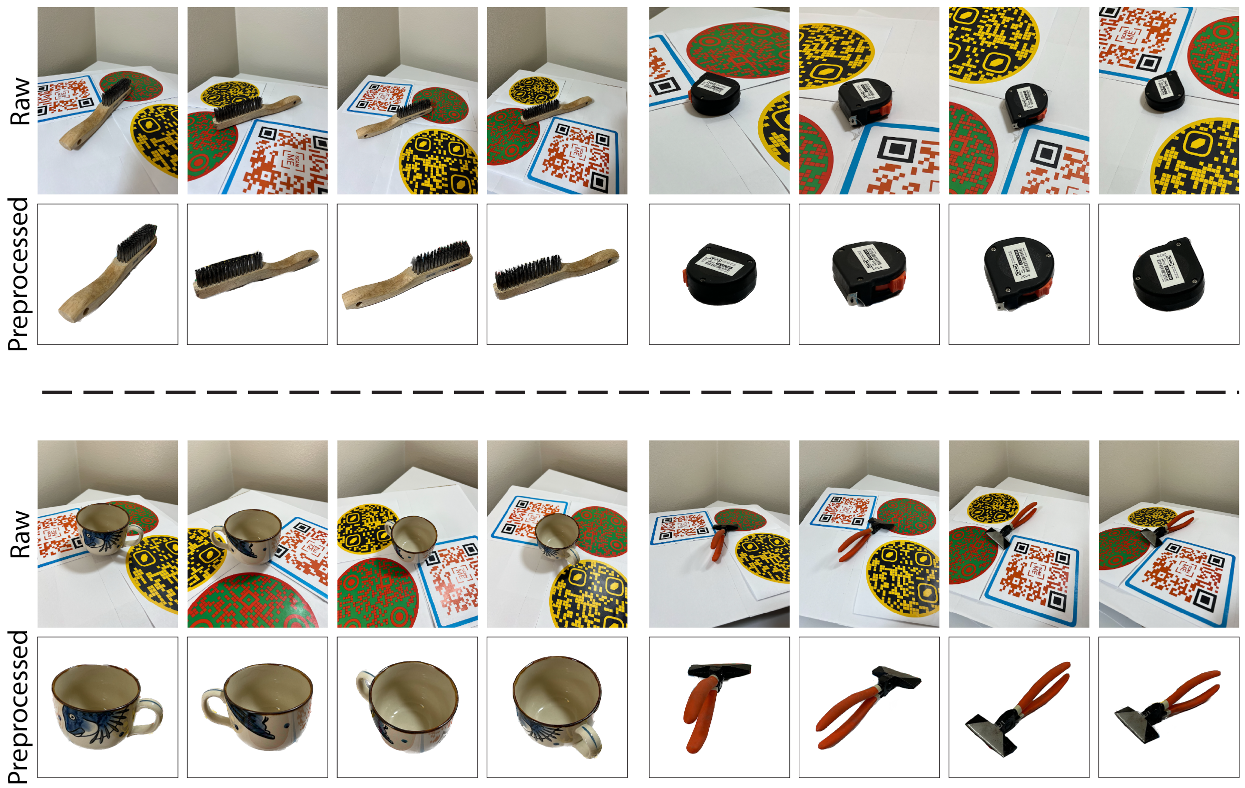


Figure 5.4: **Visualization of raw and preprocessed multi-view images**. Raw images represent the images directly output from the capture device, i.e., iPhone 13 Pro. We process raw images with segmentation and cropping before feeding them into the models.

Annotations. Our dataset includes three types of manual annotations: (1) *Object instance 3D centers* describe the 3D positions of each object instance center in a world coordinate frame. We annotate the 3D center by first averaging 3D points computed from camera poses and depth maps of different views of the object instance. The annotator is first asked to draw boxes on depth maps from $\geq$5 diverse views if possible. Each 2D bounding box is lifted to the 3D space with camera poses. The 3D centers of each object instance in a stationary period are averaged to get the initial estimation. The annotators then examine the adjust the annotated 3D points based on the RGB frames from the video sequence and captured mesh. (2) *2D bounding box annotations* are axis-aligned 2D bounding boxes of the instance every five frames starting from the beginning of the video. Specifically, we ask the annotators to go through the entire video first. We provide one video frame with a 2D bounding box to specify each object instance to the annotators. We ask annotators to draw *amodal* bounding

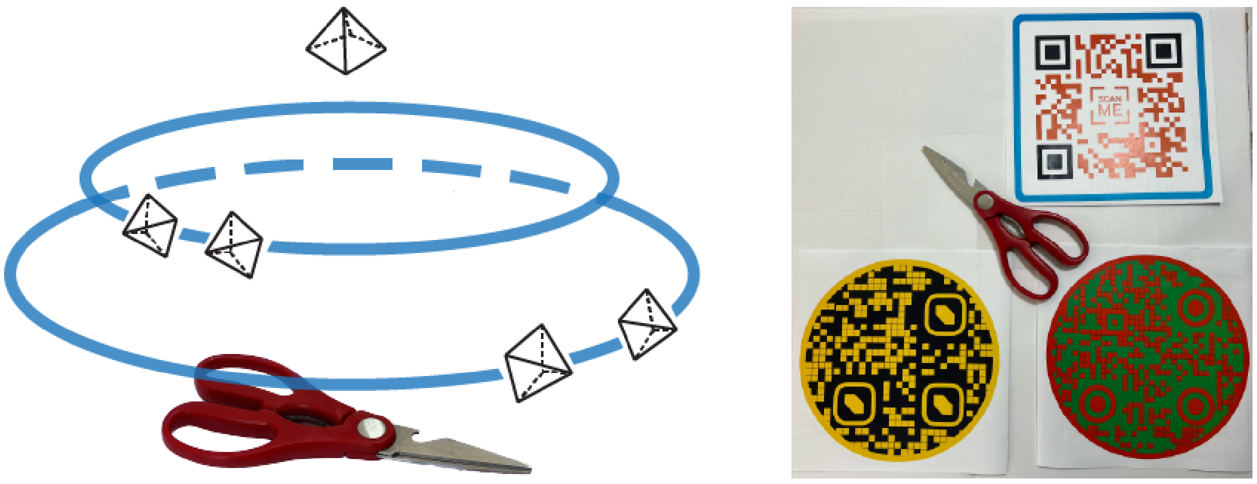


Figure 5.5: **Illustration of our multi-view capture setup.** The left panel shows our camera positions when taking 25 images to support the pre-enrollment study. Specifically, we take 12 object-centric photos evenly from 360° while keeping the camera 30° elevation. Another 12 images are taken in a similar fashion while keeping the camera 60° elevation. Lastly, we take one top-down view. An example of the top-down view with the QR code is shown on the right.

boxes of each object instance and do not annotate the object instances with heavy occlusions (i.e., when less than 25% of the object is visible). (3) *Object motion state annotations* are a per-frame annotation of whether the object is stationary or dynamic. For the data we collected, dynamic implies the camera wearer is interacting with the object. All annotations are first labeled by a group of annotators and checked by other independent annotators to ensure the quality.

5.4 Methodology

5.4.1 Baseline: Re-purposed SOT Trackers

To approach the problem of IT3DEgo, we first explore a simple *unified pipeline* as the baseline approach based on single object tracking (SOT). It allows instance-level 2D tracking by providing the visual appearance of object instances to track [47, 18, 259], which enables us to re-purpose them for our benchmark task. In the unified pipeline, we first compute the 2D trajectories of each object instance with SOT. The final 3D trajectories are computed by lifting the center of 2D bounding boxes with depth maps and camera poses. Lastly, we adopt a simple memory mechanism that stores the previous locations of each object instance to handle the case where the instance moves out of sight, i.e., frames without valid predictions.

Lifting 2D trajectories to 3D. With the 2D trajectory predicted from SOT, each valid 2D detection is then lifted into 3D space with the equation: $\mathbf{o}_t^i = \mathbf{T}_t z \mathbf{K}^{-1}\mathbf{c}_t^i$, where $\mathbf{c}_t^i$ is the 2D coordinate of the center of the bounding box of instance i at timestamp t, $\mathbf{o}_t^i$ is the 3D position of instance i at timestamp t in world coordinate. z is the corresponding depth value of $\mathbf{c}_t^i$ on the depth map. $\mathbf{T}_t$ is the camera pose at timestamp t that specifies the camera rotation and translation w.r.t to a predefined world coordinate. $\mathbf{K}$ is the intrinsic matrix. A frame may lack a valid 3D prediction because either there is no 2D location from SOT (e.g., the object is outside the field-of-view) or the depth map is missing the depth value at $\mathbf{c}_t^i$.

Completing 3D trajectories with memory. Any given frame may lack a valid 3D prediction, either because there is no 2D location from SOT (e.g., the object is outside the field-of-view) or the depth map has missing depth values at $\mathbf{c}_t^i$. To address this we implement a simple memory mechanism that stores only the most recent 3D location for each tracked instance (memory size=1). We update the memory whenever there is a new valid prediction. We note that this heuristic is a good match for the prior that object locations change only when they are being interacted with, in which case they should also be visible to the camera.

5.4.2 Improved Baseline

We also explore the approach that leverages the recent foundation model SAM [118] and state-of-the-art feature encoder DINOv2 [175] for IT3DEgo. Following a tracking-by-detection pipeline, we first compute the per-frame 2D detections of each object instance by comparing the cosine similarity of DINOv2 encoded features between candidate proposals from SAM

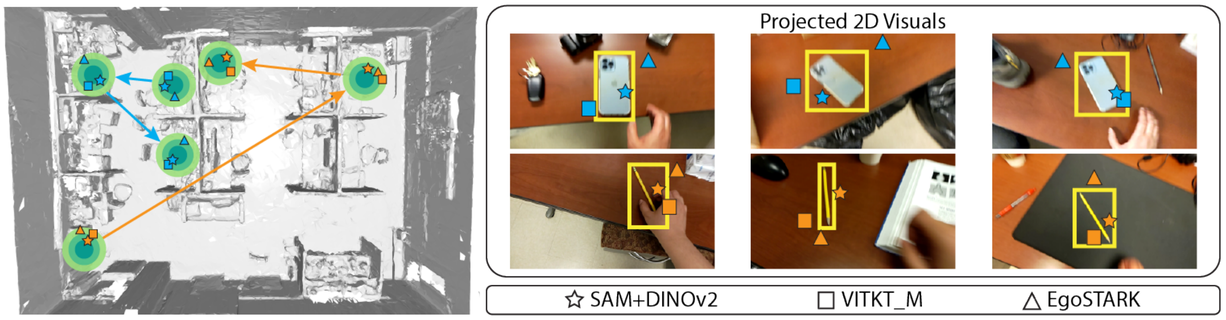


Figure 5.6: **Qualitative visualizations of tracking with SVOE in both 3D space (left) and projected 2D view (right).** We visualize three top-performing trackers from different categories, i.e., EgoSTARK, VITKT_M, and SAM+DINOv2. For projected 2D visualization, we compare the projected 3D points of each model w.r.t to the ground-truth annotated 2D bounding boxes. In the 3D view, we show 3 concentric circles at each ground-truth position representing 0.25, 0.5 and 0.75 meter thresholds. In both 2D and 3D visualizations, we find SAM+DINOv2 outperforms others as the predictions are closer to the center of object instances.

and a visual feature template. Together with the depth and camera pose information, we convert the 2D detection of each object into a 3D point in a predefined world coordinate. A simple memory with size 1 is also adopted to handle the frames without valid predictions.

Exploring motion prior with Kalman filters. Currently, the naive update mechanism, i.e., always updating the memory for all incoming predictions, does not exploit the temporal information in video sequences. Inspired by the Kalman filter [246, 182, 247] that is widely adopted in the tracking literature, we simply model the stationary position of each object instance as *piecewise constant velocity motion*, leveraging the prior information that an object without being interacted with has the same 3D coordinate. Mathematically, the motion update with Kalman filter in each stationary position: $\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t + \mathbf{K}_t(\mathbf{z_t} - \mathbf{H}\hat{\mathbf{x}}_\mathbf{t})$, where $\hat{\mathbf{x}}_t$ is a 6DOF estimated state vector including position and velocity at time step t, $\mathbf{K}_t$ is the Kalman gain, $\mathbf{z_t}$ is a 3DOF the measurement vector, $\mathbf{H}$ is the observation matrix. Please refer to Kalman [112] for more details. Moving from one stationary position to the next one, we introduce an L2 distance heuristic to model the period where objects are being interacted. Specifically, we compute the L2 distance between incoming 3D positions and the state predictions from the Kalman filter. If the L2 distance is above the threshold, we reset the Kalman filter with the current 3D predictions as the initialization.

5.5 Experiments

In this section, we first describe the implementation details of benchmark results. Then, we show the quantitative results of both setups and the visualizations of tracking results. Lastly, we demonstrate the importance of exploiting camera pose for tracking in 3D and perform ablation studies of the trackers. Note that we split our benchmark dataset into validation and test sets. All experiments are conducted on the validation set; the test set is used for future work.

Baseline SOT trackers. We choose top-ranked trackers from well-established SOT literature and VOT challenges with open-source code for both tracking setups. Specifically, we benchmark three short-term trackers ToMP [167], MixFormer [42], and ARTrack [245]; and three top-performing trackers from VOT long-term tracking challenges 2021 [124] and 2022 [123], mixLT, mlpLT and VITKT_M. We also evaluate trackers that utilize additional depth information as part of the input, including SAMF and MixForRGBD from VOT RGB-D tracking challenge 2022 [123], and ViPT [300]. Lastly, we benchmark the recent egocentric specific finetuned trackers, EgoSTARK [227]. Note that SOT trackers require initial bounding boxes to track, which are not available in MVPE. When re-purposing to MVPE setup, we explore two different initializations: (1) *detection-based initialization:* use multi-view pre-enrollment images to search for the initial bounding boxes where object instances first appear in the video and initialize SOT trackers with the predicted 2D boxes. (2) *template-based initialization:* directly adopt multi-view pre-enrollment images as visual templates in the tracker and set the initial tracking search region to the entire frame.

Table 5.2: **Benchmark results of tracking with SVOE.** From the results, we draw three salient conclusions: (1) The ability of re-identifying object instances after they disappear is important, as long-term and egocentric specific trackers outperform short-term trackers, i.e., RGB-ST and RGB-D. (2) Currently, encoding depth maps as auxiliary information cannot improve performance since depth maps are sparse and not always perfectly aligned with RGB frames due to distortions. (3) The Kalman filter smoothing yields marginal improvements over the simple memory heuristic. The method with KF subscript indicates it applies the Kalman filter.

Model	Modality	Precision(%)↑					Recall(%)↑					L2↓	Angle↓
		0.25	0.5	0.75	1.0	1.5	0.25	0.5	0.75	1.0	1.5	(m)	(rad)
ToMP	RGB-ST	5.6	10.1	17.2	25.3	39.0	6.1	11.0	18.8	27.7	42.6	2.11	1.32
MixFormer	RGB-ST	8.3	12.2	18.7	27.0	43.0	9.0	13.4	20.4	29.5	47.0	1.97	1.15
ARTrack	RGB-ST	9.1	13.9	21.5	30.3	45.1	10.1	15.3	23.7	32.4	47.2	1.92	1.10
SAMF	RGB-D	7.0	11.5	15.7	24.0	40.8	7.7	12.5	17.2	26.3	44.7	1.90	1.00
MixForRGBD	RGB-D	7.5	12.1	16.8	25.3	41.0	8.3	13.4	20.1	28.5	45.0	2.11	1.32
ViPT	RGB-D	8.9	13.6	20.6	28.1	41.4	9.7	14.9	22.5	30.7	45.3	2.02	1.21
mixLT	RGB-LT	14.4	17.5	23.9	31.8	47.2	15.8	19.2	26.1	34.8	51.6	1.85	1.02
mlpLT	RGB-LT	16.0	20.0	25.5	35.2	48.2	16.7	20.8	26.5	36.7	50.1	1.77	0.97
VITKT_M	RGB-LT	21.5	24.2	29.7	37.5	50.6	23.0	25.9	31.8	40.2	54.2	1.55	0.83
EgoSTARK	RGB-Ego	17.5	21.2	26.8	36.3	49.1	17.6	22.0	27.4	38.0	51.2	1.70	0.91
SAM + DINOv2	RGB	23.3	26.4	33.1	43.3	59.4	24.9	28.1	35.3	46.3	63.4	1.35	0.81
SAM + DINOv2$_{\mathrm{KF}}$	RGB	**23.7**	**27.1**	**33.9**	**44.5**	**61.2**	**25.5**	**29.0**	**36.8**	**48.0**	**64.9**	**1.32**	**0.79**

Implementation details. The cosine similarity threshold in the SAM+DINOv2 approach is 0.6, i.e., the object is considered not visible if the cosine similarity is smaller than the threshold. For a fair comparison, we add additional 2D prediction filtering when re-purposing SOT trackers. We discard 2D predictions from SOT trackers whose prediction scores are lower than 75% of the maximum prediction score. When tracking with MVPE, we first preprocess the captured multi-view images by segmenting and cropping the foreground object using [140]. Many transformer-based SOT trackers only encode a limited number of templates, therefore, we choose 5 images from 0°, 90°, 180°, 270° and top-down for all models in MVPE experiments. To keep the comparison fair, all detection-based trackers in MVPE use SAM+DINOv2 with the same cosine thresholds to locate the initial bounding boxes. In terms of benchmarking RGB-D trackers in MVPE, we utilize the estimated sparse depth maps using COLMAP [206]. The L2 distance threshold of resetting Kalman filters is 0.15m. All experiments are implemented with PyTorch and run on Nvidia 2080Ti GPUs.

5.5.1 Benchmark Results

Tracking with SVOE. From the results shown in Table 5.2, we have the following salient insights: (1) *Re-identifying object instances is important.* Trackers designed with strong re-identify ability, i.e., long-term and egocentric specific types, outperform short-term trackers. Similar findings are shown in recent 2D egocentric tracking work [60, 227]. Surprisingly, SAM+DINOv2, the non-learned approach which does not exploit temporal information beyond the memory heuristic, performs the best among all baselines. We believe the exhaustive proposals on every frame and high quality features provide the model strong, generic re-identification ability. (2) *Depth information is not fully leveraged.* Current RGB-D trackers show similar or slightly worse performance compared to RGB-ST trackers (c.f. MixFormer and MixforRGBD). The main reason is that RGB-D trackers only encode depth maps as auxiliary visual features, which cannot fully exploit the geometric information from depth maps. Additionally, the depth maps are sparse and not always perfectly aligned with RGB images due to camera distortions. (3) *Simple Kalman filter brings marginal benefits.* The Kalman filter does not improve over the simple "most recent" memory heuristic for stationary objects. The naive filter is also not sufficient for modeling the switching between stationary and dynamic motions needed to capture user-object interactions.

Tracking with MVPE. We benchmark top-performing trackers in each category in Table 5.2 for MVPE setup. From the results shown in Table 5.3, we find: (1) *SOT trackers cannot fully exploit pre-enrollment information.* SOT methods rely on the initial position defined by 2D boxes on the frame to perform well. Comparing detection-based initializations and SVOE results, e.g., ARTrack$^{\mathrm{D}}$ and ARTrack in Table 5.2, the model performance drops since the initial boxes are not as accurate as ground-truth initialization. VITKT_M adopts many complicated modules that all rely on the initial bounding boxes and degrades more significantly, compared to other types of trackers. (2) *Encoding rich visual information generally helps.* From the results of template-based initializations, VITKT_M for the same

Table 5.3: **Benchmark results of tracking with MVPE.** We evaluate top-performing trackers in each category in Table 5.2 for MVPE setup. From the results, we have the following summaries: (1) *SOT trackers cannot fully exploit pre-enrollment information.* Detection-initialized versions perform less well compared to SVOE due to the inaccurate estimated initial bounding boxes. VITKT_M, which uses many modules that rely heavily on the initialization, degrades more significantly. (2) *Encoding rich visual information generally helps.* SAM+DINOv2 shows an even larger performance boost because it is more robust to the inaccurate initialization. The D and T superscripts indicate the detection- and template-based initializations, respectively.

Model	Modality	Precision(%)↑					Recall(%)↑					L2↓	Angle↓
		0.25	0.5	0.75	1.0	1.5	0.25	0.5	0.75	1.0	1.5	(m)	(rad)
ARTrack^D	RGB-ST	6.8	12.1	18.5	25.8	41.0	7.1	12.3	18.8	26.8	42.1	1.98	1.16
ARTrack^T	RGB-ST	11.2	18.1	25.2	28.7	38.5	12.7	20.9	28.5	33.0	45.1	1.91	1.07
ViPT^D	RGB-D	6.3	11.7	17.9	24.9	40.2	6.9	11.8	17.9	25.9	41.0	2.01	1.21
ViPT^T	RGB-D	10.5	17.4	24.0	27.1	36.3	11.9	20.1	27.1	31.3	44.0	1.93	1.10
VITKT_M^D	RGB-LT	13.8	18.0	25.0	33.0	46.6	14.3	18.6	25.8	34.1	48.2	1.77	0.98
VITKT_M^T	RGB-LT	9.2	12.8	20.7	28.5	44.0	9.7	14.2	22.4	31.3	46.5	1.95	1.08
EgoSTARK^D	RGB-Ego	13.2	17.0	23.1	30.5	47.0	14.7	18.5	25.9	34.4	49.7	1.82	1.01
EgoSTARK^T	RGB-Ego	18.9	23.1	28.3	37.1	49.6	19.1	23.1	29.3	39.1	52.9	1.67	0.88
SAM + DINOv2	RGB	56.0	59.0	61.8	67.5	74.3	50.0	52.7	55.2	60.3	66.4	0.67	0.40
SAM + DINOv2 $_{KF}$	RGB	**56.2**	**59.4**	**62.2**	**68.1**	**74.8**	**50.3**	**53.1**	**55.7**	**61.1**	**67.1**	**0.65**	**0.39**

reason mentioned before, we find trackers benefit from the high-resolution multi-view images. SAM+DINOv2 shows a significant performance boost because it is more robust to inaccurate initialization without relying on temporal information.

Qualitative results. Figure 5.6 shows predictions of top-performing trackers from three different categories, i.e., best tracker in long-term and egocentric specific, and SAM+DINOv2. Clearly, SAM+DINOv2 predictions are closer to the object center in both 3D and projected 2D space.

5.5.2 Further Analysis and Ablation Study

We further compare tracking object instances in both 2D and 3D settings, demonstrating tracking object instances is much easier in 3D space. We also include an ablation study regarding the cosine similarity thresholds. All studies shown in this section are using SAM+DINOv2 unless otherwise specified.

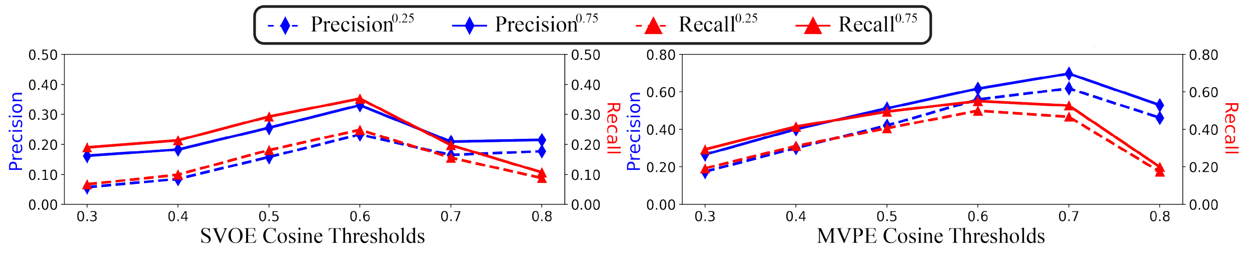


Figure 5.7: **Performance comparisons of SAM+DINOv2 with different cosine thresholds.** By increasing the threshold, we find the model performance first improves and then gradually decreases. Intuitively, increasing the threshold will initially filter noisy predictions but when the threshold is too large the model will miss correct object 3D location updates.

Table 5.4: **Quantitative comparisons of 2D tracking results w/ and w/o 3D guidance.** With 3D guidance means the 2D results are computed by finding the bounding box proposal with the smallest L2 distance from projected 3D trajectories. Without 3D guidance means proposals are selected purely based on the visual feature cosine similarity. Please refer to Section 5.5.2 for more details. From the results, we find the tracking results are significantly improved with the 3D guidance, indicating that tracking in 3D in egocentric videos is much easier than in 2D by leveraging camera pose and depth sensors.

	3D Guid.	AUC(%)↑	N. Prec.(%)↑	Prec.(%)↑
SVOE	✗	20.7	14.9	8.9
	✓	**27.6**	**21.7**	**11.5**
MVPE	✗	14.1	7.4	3.0
	✓	**39.1**	**35.2**	**18.7**

Tracking in 2D with 3D guidance. We experimentally demonstrate the importance of leveraging 3D information in egocentric instance tracking by comparing 2D tracking results w/ and w/o 3D guidance. With 3D guidance means the 2D tracking results are computed with *predicted* 3D trajectories as the guidance. For each object instance, the per-frame 2D detection results are computed by selecting the (above threshold) proposal with the smallest L2 distance between projected 3D points and the center of bounding boxes proposals. Without 3D guidance means the 2D tracking results are produced by selecting the proposal with the highest cosine feature similarity. To keep a fair comparison, the cosine similarity threshold is the same when computing the 3D and 2D trajectories. We evaluate the 2D tracking performance using widely adopted precision, normalized precision metrics and AUC in SOT literature [253]. As shown in Table 5.4, the model with 3D guidance performs significantly better in both SVOE and MVPE, demonstrating that leveraging the 3D information, such as

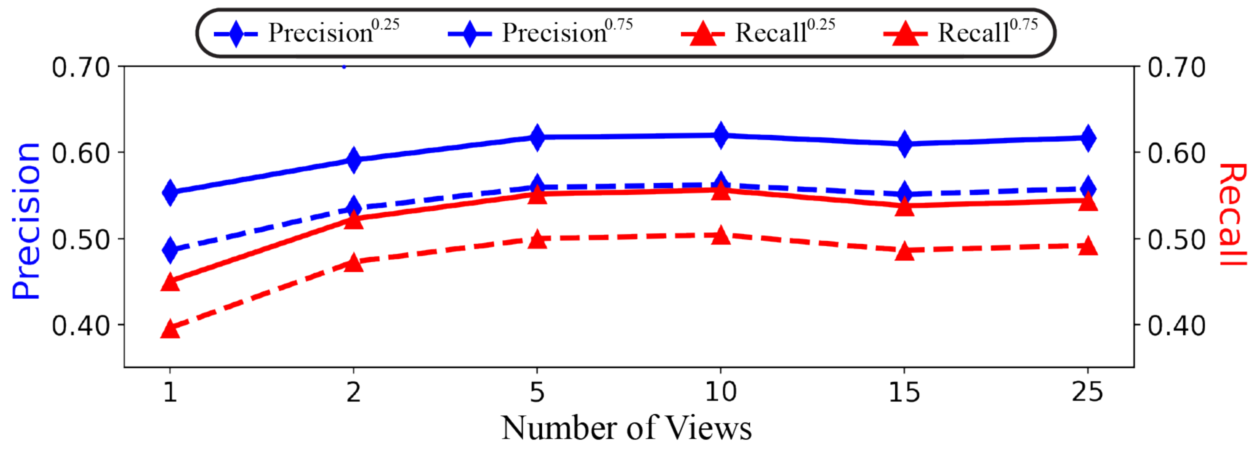


Figure 5.8: **Performance w.r.t number views in MVPE.** We run SAM+DINOv2 with different numbers of views while keeping everything else the same for a fair comparison. We find the performance saturates after using 5 views. This suggests that simply encode and average features benefit from a higher number of views (i.e., number of views from 1 to 5) but still cannot fully exploit the visual information from different views (i.e., after using 5 views).

camera pose and depth map, makes the tracking problem much easier.

Performance w.r.t cosine similarity thresholds. Feature cosine similarity threshold is adopted to determine whether the object instance is present in the current frame, which is crucial for the memory updating mechanism. To characterize the relationship between tracking performance and cosine similarity thresholds, we run the experiment with different cosine similarity thresholds but keep everything else the same. As shown in Figure 5.7, both models show improved performance at first and then a gradual decrease. Higher cosine thresholds result in fewer predictions so the model must increasingly rely on previous confident predictions stored in the memory. Models with large cosine similarity thresholds have a higher chance of missing valid location updates, which leads to a drop in both precision and recall.

Performance w.r.t number of views. Due to the architecture design of many transformer-based trackers, we only use 5 views in the benchmark experiment. In this section, we further study the relationship between the number of views and the tracking performance. Specifically, we compare the performance of SAM+DINOv2 with 1, 2, 5, 10, 15, 25 images while keeping

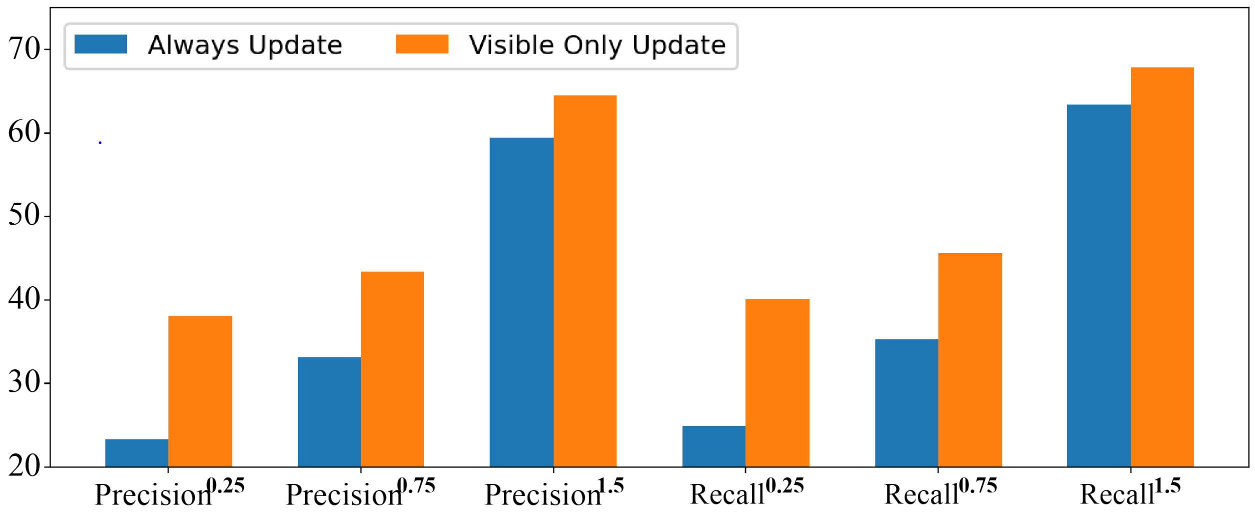


Figure 5.9: **Performance improvement by updating on visible only frames.** We control the memory update of SAM+DINOv2 by updating the memory only when the object instance is visible. We find the performance is significantly improved, indicating one of the major challenges of the baseline is to correctly update the memory with high quality predictions.

all other parameters the same. As shown in Figure 5.8, the performance improves from 1 view to 5 views but quickly saturates after using 5 views. This suggests that naively encode and average features benefit from a higher number of views but still cannot fully exploit the visual information from different views.

Performance improvement with visible update only. From the results shown in Table 5.2 and Table 5.3, we find identifying high quality predictions and updating the memory is the main challenge in the proposed baseline pipeline. To further validate this idea, we control the update of memory in SAM+DINOv2 model by only updating on the visible frames. We extract the visible information from the 2D annotations. In other words, the memory for each instance is only updated on the frame where the 2D bounding box is annotated. As shown in Figure 5.9, updating the memory only when object instances are visible significantly improves the performance. Although the update timing is correct, errors from 2D predictions, depth maps and camera poses prevent the model from improving further.

Performance w.r.t different feature encoders. The top-performing baseline, i.e., SAM+DINOv2 adopts DINOv2 as the pretrained feature encoder. To further explore

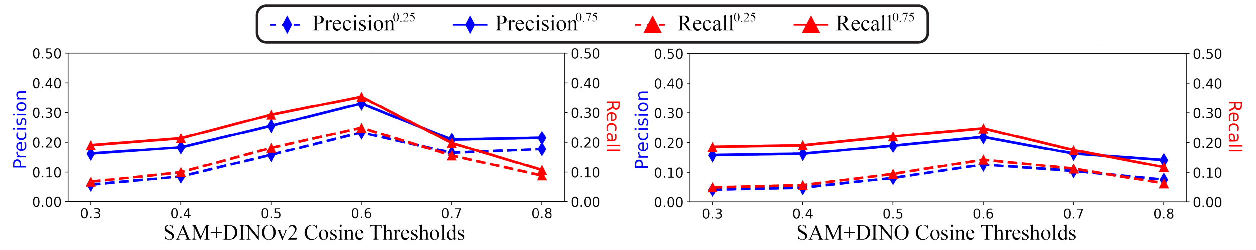


Figure 5.10: **Performance comparisons of different encoders at various cosine thresholds**. From the results, we find: (1) *Stronger encoder improve the performance.* The best performance of SAM+DINOv2 is stronger than SAM+DINO where both models have the peak performance when the cosine threshold equals 0.6. (2) *Similar performance trend w.r.t cosine similarity changes.* The performance of both models first improves and then gradually decreases when increasing the cosine threshold from 0.3 to 0.8.

Table 5.5: **Quantitative comparisons of different proposal generators.** We compare the performance of SAM+DINOv2 and YOLOv7+DINOv2. To keep the comparison fair, the only differences between these models are the proposal generators. From the results, we find adopting YOLOv7 makes the performance slightly worse. The proposal quality from YOLOv7 is lower but runs faster.

Proposals	Precision(%)↑			Recall(%)↑			L2↓
	0.25	0.75	1.5	0.25	0.75	1.5	(m)
YOLOv7	20.3	28.1	50.2	21.5	30.7	53.9	1.72
SAM	**23.3**	**33.1**	**59.4**	**24.9**	**35.3**	**63.4**	**1.35**

the performance w.r.t different large-scale feature encoders, we experiment with another state-of-the-art feature encoder, i.e., DINO [31]. We plot the results using DINO and DINOv2 at different cosine thresholds in Figure 5.10. From the results, we find: (1) *Stronger encoder improves the performance.* The best performance of SAM+DINOv2 is stronger than SAM+DINO where both models have the peak performance when the cosine threshold equals 0.6. (2) *Similar performance trend w.r.t cosine similarity changes.* The performance of both models first improves and then gradually decreases when increasing the cosine threshold from 0.3 to 0.8.

Comparisons of different proposal generators. Currently, the improved baseline utilizes SAM as the proposal generator. In this part, we replace SAM with the proposals from YOLOv7, i.e., the output before the final classification layer. The results are shown in Table 5.5. Although the performance of YOLOv7+DINOv2 is lower compared to SAM+DINOv2, which is not surprising. The proposal quality from YOLOv7 is lower but runs faster. However, the current baseline approaches are not able to run in real time due to the following encoding and lifting steps. One promising direction for future work is to improve the speed of the tracking models.

5.6 Discussion

Limitations and future work. We point out that the current benchmark dataset has limited geographic and demographic diversity and captures only a small range of objects and activities. As such it is not appropriate for training large models and only serves as a diagnostic test to identify some limitations of existing approaches. Our hope is that it serves as a starting point for the research community to explore and eventually grow into a more comprehensive challenge. Currently, the studied baseline approaches follow the same paradigm, i.e., lifting predicted 2D trajectories into 3D space. We found empirically that the simplest memory mechanism performed best but it seems very likely there are more nuanced state-update models which can integrate multiple observations effectively.

Finally, we highlight two opportunities for future work. First, advanced models to detect object 3D motion changes. Our experiments demonstrate that tracking in 3D world coordinates effectively narrows the problem to that of accurately predicting the object motion status, i.e., finding all stationary periods for each object instance. However, accurately predicting object state changes is still a non-trivial problem to solve. Second, better utilization of object instance information. Currently, the object instances enrollments, i.e., SVOE and MVPE are naively encoded as visual features. Future work should explore the approaches of fusing the additional scene 3D information with object instances for better tracking performance.

Broader impact. We believe the broader impact of our work is two-fold. First, we hope our benchmark brings more attention to the problem of tracking object instances in 3D from the egocentric perspective and contributes towards building future task-aware assistive agents. Second, our multi-modal benchmark dataset is beneficial to the study of other 3D scene understanding related problems from the egocentric perspective, such as SLAM, camera localization, 3D reconstruction, and depth estimation.

Potential negative impacts. Tracking in 3D from egocentric videos requires the geometric data of surrounding environments and the sensor streams that continuously capture their workplace or daily lives. There are obvious privacy concerns when deploying such hardware and algorithms. Similar to other apps running on personal devices, the simple solution is to keep all user data locally or (in the context of research) develop techniques for anonymizing video [229].

5.7 Conclusion

We introduce a new *IT3DEgo* benchmark that allows us to study the problem of tracking object instances in 3D from egocentric videos. The object instances to be tracked are either determined in advance or enrolled online during user interactions with the environment. To support the study, we collect and annotate a new dataset that features RGB-D videos and per-frame camera poses, along with instance-level annotations in both 2D camera and 3D world coordinate frames. We re-purpose and evaluate state-of-the-art single object trackers and develop a strong baseline using large pretrained recognition models and Kalman filtering. We hope our benchmark brings more attention to this challenge and contributes to the development of perceptually-aware assistive agents.

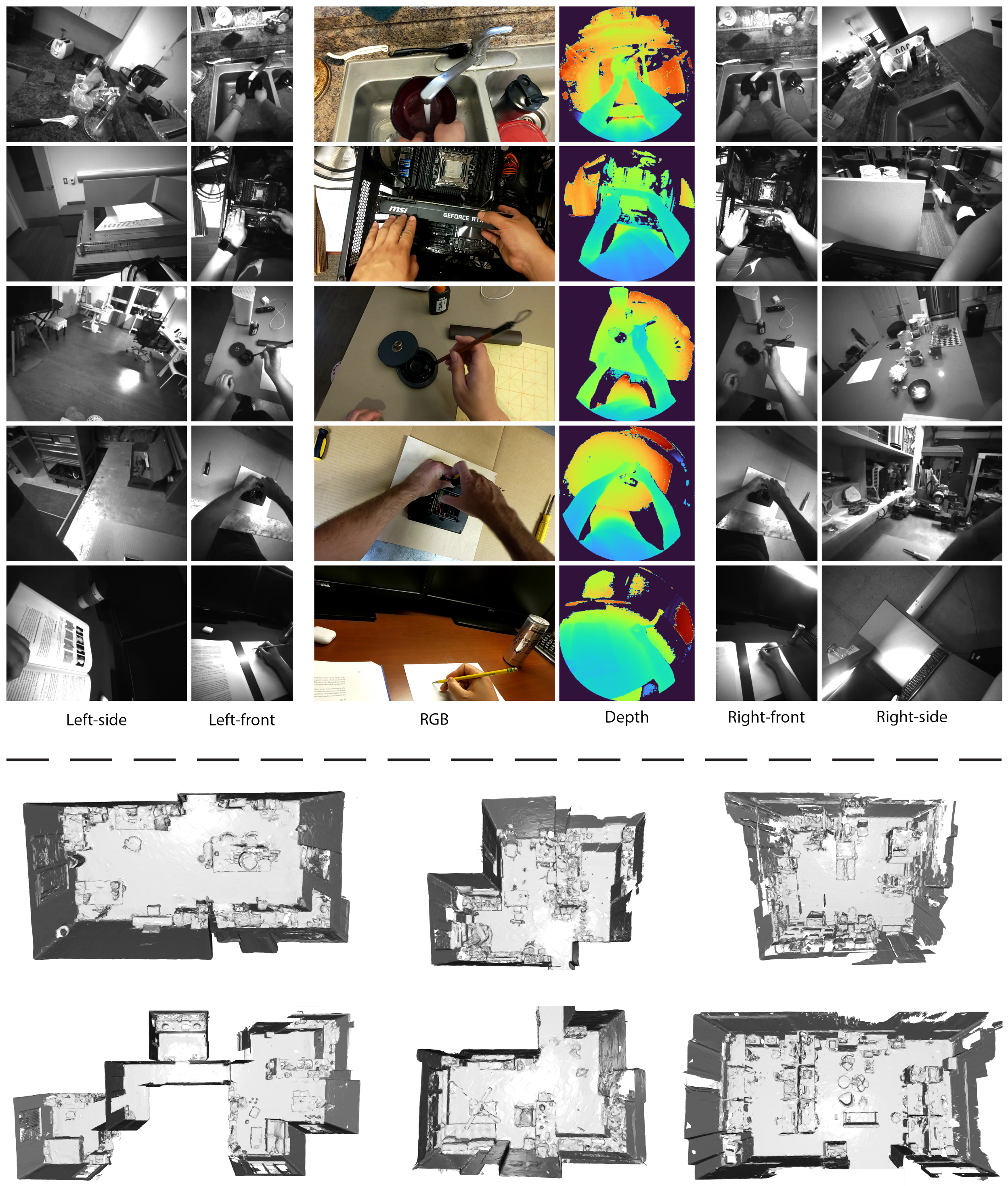


Figure 5.11: **2D visualizations of frames from raw video sequences (upper panel) and 3D visualizations of the capture environments (lower panel)**. The benchmark videos record camera wearers perform naturalistic tasks in real-world scenarios, such as cooking and repairing. Please refer to Figure 5.3 for the layout of each sensor on the HoloLens2.

Chapter 6

Concluding Remarks

6.1 Summary of Contributions

In Chapter 2, we reveal the high-level domain gaps in learning monocular depth estimators. Specifically, high-level domain gaps refer to the difference between real and synthetic images, *i.e.* the cluttered scene and novel objects. State-of-the-art models focus on closing the low-level gaps, such as colors and textures. We propose the attend-remove-complete (ARC) approach that learns to remove the regions that are challenging and detrimental to overall depth prediction performance. ARC outperforms state-of-the-art methods on both indoor and outdoor datasets, suggesting the effectiveness of our proposed solution.

In Chapter 3, we describe the camera pose distribution shift between training and testing data. State-of-the-art depth estimators rely on large-scale data for end-to-end training. However, we find that trained predictors fail to make reliable depth predictions for testing examples captured under uncommon camera poses. To mitigate this problem, we propose two novel solutions: perspective-aware data augmentation and camera pose prior encoding. The former one allows us to augment RGB-D data in a geometrically consistent way. The latter one encodes the camera pose to learn a pose conditional depth predictor. Experimental results demonstrate that both of the proposed approaches successfully mitigate the bias and outperform state-of-the-art depth estimators.

In Chapter 4, we explore the reference-based image inpainting task that aims to inpaint the hole region in the target image leveraging the texture from another reference image. This task requires understanding the 3D geometry that relates pixels between two views to accurately paste the texture for inpainting. We propose GeoFill that leverages an explicit non-planar 3D scene representation from depth maps given only limited two-view RGB images. Specifically, we utilize pretrained sparse correspondence models and monocular depth estimators to reconstruct the scene geometry. A joint optimization step is introduced to adjust the pose and relative pose. Experimental results show that GeoFill achieves state-of-the-art performance. It also better handles large camera movements and complicated geometric structures.

In Chapter 5, we introduce a new benchmark problem to explore instance tracking in 3D from egocentric videos. This task is motivated by building task-assistive agents running on AR/VR devices. Tracking from the egocentric perspective is different as the camera is constantly moving while the object instance is stationary unless being interacted with. Therefore, we formulate the problem in the 3D scene to leverage the camera poses and depth maps. We explore this problem by collecting and annotating a new benchmark dataset. We also re-purpose and evaluate state-of-the-art single object trackers on our benchmark test and propose a novel method leveraging recent foundation models.

6.2 Future Directions

Based on the chapters discussed above, there are many exciting directions to explore in the future.

Robust depth estimation. A major problem of current state-of-the-art monocular depth estimators is that the model only performs well in certain scenarios, such as indoor scenes only. This problem is largely due to the significant depth range difference between indoor and outdoor scenarios, and the limited scale of RGB-D data. Recent works already start to address this problem by predicting the inverse depth and training models on diverse datasets [192, 190]. Future work could try to leverage the powerful foundation models or inject more object-level knowledge, such as object size to achieve more robust depth predictions.

Self-supervised depth estimation. Self-supervised learning paradigm does not need the ground truth annotation, which enables the potential to train on a significantly larger scale data. Current approaches exploit the relationship between frames and use the photometric loss to supervise the depth training [84, 21]. However, the estimated pose and depth, can only be estimated up to an unknown scale, which makes the model struggle to predict consistent scales between frames. The model also struggles with dynamic objects and texture-less regions. It is worth investigating these issues by incorporating semantic information and stronger geometric constraints.

3D-aware image/video generation. Recent image/video generation models are capable of generating high fidelity textures, such as diffusion models [51, 102]. One major problem that remains is to generate controllable consistent textures across different views or in long-term videos. State-of-the-art models incorporate 3D representation into the generation step or add explicit geometric constraints [33, 32]. It is worth for future work to explore different 3D representations for long-term video generation.

Task-assistive agents on egocentric videos. The strong demand of building task-assistive agents motivates the future work to further explore in this direction [201, 88]. In terms of the IT3DEgo benchmark, we suggest further work in the following two directions. First, advanced models to detect object 3D motion changes. Second, better utilization of object instance information. In terms of task-assistive agents in general, future works should explore combining IT3DEgo with other tasks to enable agents to execute a series of actions or finish more complicated tasks.

Bibliography

[1] Hololens 2 sensor streaming. `https://github.com/jdibenes/hl2ss`, 2023.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[4] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7822–7831, 2021.

[5] I. Alhashim and P. Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018.

[6] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.

[7] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[8] A. Atapour-Abarghouei and T. P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2800–2810, 2018.

[9] S.-H. Baek, I. Choi, and M. H. Kim. Multiview image completion with space structure propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 488–496, 2016.

[10] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 1090–1099, 2022.

[11] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 2006.

[12] M. Baradad and A. Torralba. Height and uprightness invariance for 3d prediction from a single view. In *CVPR*, 2020.

[13] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[14] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[15] E. Barshan and P. Fieguth. Stage-wise training: An improved feature learning strategy for deep models. In *Feature Extraction: Modern Questions and Challenges*, pages 49–59, 2015.

[16] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[17] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[18] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6182–6191, 2019.

[19] A. V. Bhavsar and A. N. Rajagopalan. Inpainting in multi-image stereo. In *Joint Pattern Recognition Symposium*, pages 172–181. Springer, 2010.

[20] H. Bi, R. Zhang, T. Mao, Z. Deng, and Z. Wang. How can i see my future? fvtraj: Using first-person view for pedestrian trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 576–593. Springer, 2020.

[21] J. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems*, 2019.

[22] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, pages 536–551. Springer, 2014.

[23] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017.

[24] E. Brachmann and C. Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4322–4331, 2019.

[25] M. Brady. *Robot motion: Planning and control.* MIT press, 1982.

[26] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[27] P. Buyssens, O. Le Meur, M. Daisy, D. Tschumperlé, and O. Lézoray. Depth-guided disocclusion inpainting of synthesized rgb-d images. *IEEE Transactions on Image Processing*, 2016.

[28] C. Cadena, A. R. Dick, and I. D. Reid. Multi-modal auto-encoders as joint estimators for robotics scene understanding. In *Robotics: Science and systems*, volume 5, 2016.

[29] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[30] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[31] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[32] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022.

[33] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.

[34] K. Chen, N. Snavely, and A. Makadia. Wide-baseline relative camera pose estimation with directional learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3258–3268, 2021.

[35] X. Chen, X. Chen, and Z.-J. Zha. Structure-aware residual pyramid network for monocular depth estimation. *arXiv preprint arXiv:1907.06023*, 2019.

[36] Y. Chen, Y. Li, X. Zhang, J. Sun, and J. Jia. Focal sparse convolutional networks for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5428–5437, 2022.

[37] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21674–21683, 2023.

[38] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang. Crdoco: Pixel-level domain transfer with cross-domain consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1791–1800, 2019.

[39] Z. Chen, Z. Yuan, J. Yi, B. Zhou, E. Chen, and T. Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. *arXiv preprint arXiv:1808.06296*, 2018.

[40] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[41] S. Cova, A. Longoni, and A. Andreoni. Towards picosecond resolution with single-photon avalanche diodes. *Review of Scientific Instruments*, 52(3):408–412, 1981.

[42] Y. Cui, C. Jiang, L. Wang, and G. Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13608–13618, 2022.

[43] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[44] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[45] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[46] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. W. Mayol-Cuevas. You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *BMVC*, volume 2, page 3, 2014.

[47] M. Danelljan, L. V. Gool, and R. Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7183–7192, 2020.

[48] S. Datta, S. Dharur, V. Cartillier, R. Desai, M. Khanna, D. Batra, and D. Parikh. Episodic memory question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19119–19128, 2022.

[49] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deep slam. *arXiv preprint arXiv:1707.07410*, 2017.

[50] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[51] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[52] T. v. Dijk and G. d. Croon. How do neural networks see depth in single images? In *ICCV*, 2019.

[53] M. Dimiccoli, J. Marín, and E. Thomaz. Mitigating bystander privacy concerns in egocentric activity recognition with deep learning and intentional image degradation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–18, 2018.

[54] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[55] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[56] D. Droeschel, D. Holz, and S. Behnke. Multi-frequency phase unwrapping for time-of-flight cameras. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1463–1469. IEEE, 2010.

[57] D. Droeschel, D. Holz, and S. Behnke. Probabilistic phase unwrapping for time-of-flight cameras. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–7. VDE, 2010.

[58] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. B. Chaudhuri. diffgrad: An optimization method for convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 31(11):4500–4511, 2019.

[59] M. Dunnhofer, A. Furnari, G. M. Farinella, and C. Micheloni. Is first person vision challenging for object tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2698–2710, 2021.

[60] M. Dunnhofer, A. Furnari, G. M. Farinella, and C. Micheloni. Visual object tracking in first person vision. *International Journal of Computer Vision*, 131(1):259–283, 2023.

[61] D. Dwibedi, I. Misra, and M. Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1301–1310, 2017.

[62] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[63] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, 2014.

[64] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera. Cam-convs: camera-aware multi-scale convolutions for single-view depth. In *CVPR*, 2019.

[65] J. Fan, P. Zheng, and S. Li. Vision-based holistic scene understanding towards proactive human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, 75:102304, 2022.

[66] Z. Fang, S. Kong, C. Fowlkes, and Y. Yang. Modularized textual grounding for counterfactual resilience. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6378–6388, 2019.

[67] A. Fathi, J. K. Hodgins, and J. M. Rehg. Social interactions: A first-person perspective. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1226–1233. IEEE, 2012.

[68] B. Fernando and S. Herath. Anticipating human actions by correlating past with the future with jaccard similarity measures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13224–13233, 2021.

[69] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[70] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[71] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[72] A. Furnari and G. M. Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6252–6261, 2019.

[73] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

[74] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[75] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[76] M. Garon and J.-F. Lalonde. Deep 6-dof tracking. *IEEE transactions on visualization and computer graphics*, 23(11):2410–2418, 2017.

[77] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. Iii, and K. Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[78] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

[79] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[80] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[81] G. Georgakis, M. A. Reza, A. Mousavian, P.-H. Le, and J. Kovsecká. Multiview rgb-d dataset for object instance detection. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 426–434. IEEE, 2016.

[82] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018.

[83] B. Girod and S. Scherock. Depth from defocus of structured light. In *Optics, Illumination, and Image Sensing for Machine Vision IV*, volume 1194, pages 209–215. SPIE, 1990.

[84] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3838, 2019.

[85] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[86] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[87] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background inpainting for videos with dynamic objects and a free-moving camera. In *European Conference on Computer Vision*, pages 682–695. Springer, 2012.

[88] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. 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.

[89] E. J. Gumbel. *Statistics of extremes.* Courier Corporation, 2012.

[90] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang. Learning monocular depth by distilling cross-domain stereo networks. In *ECCV*, 2018.

[91] M. Hansard, S. Lee, O. Choi, and R. P. Horaud. *Time-of-flight cameras: principles, methods and applications.* Springer Science & Business Media, 2012.

[92] Z. Hao, Y. Li, S. You, and F. Lu. Detail preserving depth estimation from a single image using attention guided networks. In *2018 International Conference on 3D Vision (3DV)*, pages 304–313. IEEE, 2018.

[93] K. G. Harding. High accuracy structured light profiler, Jan. 8 1991. US Patent 4,983,043.

[94] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[95] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593, 1997.

[96] R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[97] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 2009.

[98] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[99] L. He, G. Wang, and Z. Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 2018.

[100] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[101] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[102] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[103] T. Hodavn, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter. Photorealistic image synthesis for object instance detection. In *2019 IEEE international conference on image processing (ICIP)*, pages 66–70. IEEE, 2019.

[104] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[105] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH 2005 Papers*, pages 577–584. 2005.

[106] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[107] L. Huynh, P. Nguyen-Ha, J. Matas, E. Rahtu, and J. Heikkilä. Guiding monocular depth estimation using depth-attention volume. In *European Conference on Computer Vision*, pages 581–597. Springer, 2020.

[108] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[109] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[110] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

[111] H. Jiang and K. Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3509. IEEE, 2017.

[112] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[113] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5492–5501, 2019.

[114] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[115] A. Kim, A. O**v**sep, and L. Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11315–11321. IEEE, 2021.

[116] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[117] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[118] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[119] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[120] S. Kong and C. Fowlkes. Pixel-wise attentional gating for scene parsing. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1024–1033. IEEE, 2019.

[121] S. Kong and C. C. Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *CVPR*, 2018.

[122] P. Krähenbühl. Free supervision from video games. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2955–2964, 2018.

[123] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, H. J. Chang, M. Danelljan, L. **v**Cehovin Zajc, A. Luke**v**zi**v**c, O. Drbohlav, J. Bjorklund, Y. Zhang, Z. Zhang, S. Yan, W. Yang, D. Cai, C. Mayer, and G. Fernandez. The tenth visual object tracking vot2022 challenge results, 2022.

[124] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, H. J. Chang, M. Danelljan, L. **v**Cehovin Zajc, A. Luke**v**zi**v**c, O. Drbohlav, J. Kapyla, G. Hager, S. Yan, J. Yang, Z. Zhang, G. Fernandez, and et. al. The ninth visual object tracking vot2021 challenge results, 2021.

[125] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[126] J. N. Kundu, P. K. Uppala, A. Pahuja, and R. V. Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2656–2665, 2018.

[127] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014.

[128] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, 2016.

[129] R. Lange and P. Seitz. Solid-state time-of-flight range camera. *IEEE Journal of quantum electronics*, 37(3):390–397, 2001.

[130] K. Lasinger, R. Ranftl, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

[131] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[132] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. Roomnet: End-to-end room layout estimation. In *ICCV*, 2017.

[133] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[134] K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[135] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1346–1353. IEEE, 2012.

[136] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE intelligent vehicles symposium (IV)*, pages 163–168. IEEE, 2011.

[137] C. Levy and T. Roosendaal. Sintel. In *ACM SIGGRAPH ASIA 2010 Computer Animation Festival*, page 82. ACM, 2010.

[138] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1119–1127, 2015.

[139] H. Li, Y. Cai, and W.-S. Zheng. Deep dual relation modeling for egocentric interaction recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7932–7941, 2019.

[140] J. Li, J. Jain, and H. Shi. Matting anything. *arXiv: 2306.05399*, 2023.

[141] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. *arXiv preprint arXiv:1809.00716*, 2018.

[142] Y. Li, Z. Cao, A. Liang, B. Liang, L. Chen, H. Zhao, and C. Feng. Egocentric prediction of action target in 3d. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20971–20980. IEEE, 2022.

[143] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. Learning the depths of moving people by watching frozen people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4521–4530, 2019.

[144] Z. Li and N. Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–387, 2018.

[145] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.

[146] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[147] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh. Guidance and evaluation: Semantic-aware image inpainting for mixed scenes. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 683–700. Springer, 2020.

[148] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[149] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[150] M. Liu, S. Tang, Y. Li, and J. M. Rehg. Forecasting human-object interaction: joint prediction of motor attention and actions in first person video. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 704–721. Springer, 2020.

[151] Y. Liu, Y. Liu, C. Jiang, K. Lyu, W. Wan, H. Shen, B. Liang, Z. Fu, H. Wang, and L. Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022.

[152] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019.

[153] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2774–2781. IEEE, 2023.

[154] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[155] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu. Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 407–414, 2013.

[156] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[157] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[158] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[159] G. Luo, Y. Zhu, Z. Weng, and Z. Li. A disocclusion inpainting framework for depth-based view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[160] H. Ma, L. Chen, D. Kong, Z. Wang, X. Liu, H. Tang, X. Yan, Y. Xie, S.-Y. Lin, and X. Xie. Transfusion: Cross-view fusion with transformer for 3d human pose estimation. *arXiv preprint arXiv:2110.09554*, 2021.

[161] L. Ma, S. Georgoulis, X. Jia, and L. Van Gool. Fov-net: Field-of-view extrapolation using self-attention and uncertainty. *IEEE Robotics and Automation Letters*, 6(3):4321–4328, 2021.

[162] W. Ma, M. Zheng, W. Ma, S. Xu, and X. Zhang. Learning across views for stereo image completion. *IET Computer Vision*, 14(7):482–492, 2020.

[163] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[164] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[165] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3164–3173, 2021.

[166] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Proceedings 1998 International Conference on Image Processing*, 1998.

[167] C. Mayer, M. Danelljan, G. Bhat, M. Paul, D. P. Paudel, F. Yu, and L. Van Gool. Transforming model prediction for tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8731–8740, 2022.

[168] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019.

[169] E. Ng, D. Xiang, H. Joo, and K. Grauman. You2me: Inferring body pose in egocentric video via first and second person interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9890–9900, 2020.

[170] K. Nguyen and H. Daumé III. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. *arXiv preprint arXiv:1909.01871*, 2019.

[171] P. Nguyen, T. Liu, G. Prasad, and B. Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6752–6761, 2018.

[172] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[173] C. Northcutt, S. Zha, S. Lovegrove, and R. Newcombe. Egocom: A multi-person multi-modal egocentric communications dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[174] S. W. Oh, S. Lee, J.-Y. Lee, and S. J. Kim. Onion-peel networks for deep video completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4403–4412, 2019.

[175] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[176] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13503–13513, 2022.

[177] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[178] X. Pan, N. Charron, Y. Yang, S. Peters, T. Whelan, C. Kong, O. Parkhi, R. Newcombe, and Y. C. Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20133–20143, 2023.

[179] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *CVPR*, 2017.

[180] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[181] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[182] H. A. Patel and D. G. Thakore. Moving object tracking using kalman filter. *International Journal of Computer Science and Mobile Computing*, 2(4):326–332, 2013.

[183] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[184] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–69. IEEE, 2005.

[185] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2847–2854. IEEE, 2012.

[186] C. Plizzari, M. Planamente, G. Goletto, M. Cannici, E. Gusso, M. Matteucci, and B. Caputo. E2 (go) motion: Motion augmented event stream for egocentric action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19935–19947, 2022.

[187] R. Possas, S. P. Caceres, and F. Ramos. Egocentric activity recognition on a budget. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5967–5976, 2018.

[188] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[189] V. Raghavan, P. Bollmann, and G. S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

[190] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[191] R. Ranftl and V. Koltun. Deep fundamental matrix estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 284–299, 2018.

[192] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[193] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[194] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li. Structureflow: Image inpainting via structure-aware appearance flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 181–190, 2019.

[195] N. Rhinehart and K. M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3696–3705, 2017.

[196] G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4325–4333, 2015.

[197] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[198] M. Ryoo, B. Rothrock, C. Fleming, and H. J. Yang. Privacy-preserving human activity recognition from extreme low resolution. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[199] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2730–2737, 2013.

[200] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[201] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[202] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[203] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[204] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47:7–42, 2002.

[205] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[206] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[207] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. High-quality scanning using time-of-flight depth superresolution. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE, 2008.

[208] S. Shafique, B. Kong, S. Kong, and C. Fowlkes. Creating a forensic database of shoeprints from online shoe-tread photos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 858–868, 2023.

[209] S. Shafique, S. Kong, and C. Fowlkes. Crisp: Leveraging tread depth maps for enhanced crime-scene shoeprint matching. *arXiv preprint arXiv:2404.16972*, 2024.

[210] Q. Shen, Y. Zhao, N. Kwon, J. Kim, Y. Li, and S. Kong. A high-resolution dataset for instance detection with multi-view object capture. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[211] R. R. Shetty, M. Fritz, and B. Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In *Advances in Neural Information Processing Systems*, pages 7706–7716, 2018.

[212] H. Shi, L. Ball, G. Thattai, D. Zhang, L. Hu, Q. Gao, S. Shakiah, X. Gao, A. Padmakumar, B. Yang, et al. Alexa, play with robot: Introducing the first alexa prize simbot challenge on embodied ai. *arXiv preprint arXiv:2308.05221*, 2023.

[213] D. Shin, Z. Ren, E. B. Sudderth, and C. C. Fowlkes. 3d scene reconstruction with multi-layer depth and epipolar transformers. In *ICCV*, 2019.

[214] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[215] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[216] I. Skorokhodov, A. Siarohin, Y. Xu, J. Ren, H.-Y. Lee, P. Wonka, and S. Tulyakov. 3d generation on imagenet. In *International Conference on Learning Representations*, 2023.

[217] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Siggraph 2006 Papers*, pages 835–846. 2006.

[218] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 808–816, 2016.

[219] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[220] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[221] Y. Song, C. Yang, Y. Shen, P. Wang, Q. Huang, and C.-C. J. Kuo. Spg-net: Segmentation prediction and guidance network for image inpainting. *arXiv preprint arXiv:1805.03356*, 2018.

[222] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[223] Y.-C. Su and K. Grauman. Detecting engagement in egocentric video. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 454–471. Springer, 2016.

[224] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[225] Y. Sun and R. Fisher. Object-based visual attention for computer vision. *Artificial intelligence*, 146(1):77–123, 2003.

[226] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[227] H. Tang, K. Liang, K. Grauman, M. Feiszli, and W. Wang. Egotracks: A long-term egocentric visual object tracking dataset. *arXiv preprint arXiv:2301.03213*, 2023.

[228] H. Tao, H. S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 532–539. IEEE, 2001.

[229] D. Thapar, A. Nigam, and C. Arora. Anonymizing egocentric videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2320–2329, 2021.

[230] T. Thonat, E. Shechtman, S. Paris, and G. Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 351–359. IEEE, 2016.

[231] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7472–7481, 2018.

[232] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

[233] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[234] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[235] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017.

[236] D. Ungureanu, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger, et al. Hololens 2 research mode as a tool for computer vision research. *arXiv preprint arXiv:2008.11239*, 2020.

[237] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117, 2017.

[238] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[239] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[240] C. Wang, H. Huang, X. Han, and J. Wang. Video inpainting by jointly learning temporal structure and spatial details. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5232–5239, 2019.

[241] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6288–6297, 2020.

[242] J. Wang, L. Liu, W. Xu, K. Sarkar, D. Luvizon, and C. Theobalt. Estimating egocentric 3d human pose in the wild with external weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13157–13166, 2022.

[243] J. Wang, Y. Zhong, Y. Dai, S. Birchfield, K. Zhang, N. Smolyanskiy, and H. Li. Deep two-view structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8953–8962, 2021.

[244] L. Wang, H. Jin, R. Yang, and M. Gong. Stereoscopic inpainting: Joint color and depth completion from stereo images. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[245] X. Wei, Y. Bai, Y. Zheng, D. Shi, and Y. Gong. Autoregressive visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9697–9706, 2023.

[246] S.-K. Weng, C.-M. Kuo, and S.-K. Tu. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, 2006.

[247] X. Weng, J. Wang, D. Held, and K. Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[248] L. Westover. Footprint evaluation for volume rendering. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, 1990.

[249] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[250] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007.

[251] S. Workman, M. Zhai, and N. Jacobs. Horizon lines in the wild. *arXiv preprint arXiv:1604.02129*, 2016.

[252] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[253] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[254] W. Xian, Z. Li, M. Fisher, J. Eisenmann, E. Shechtman, and N. Snavely. Uprightnet: geometry-aware camera orientation estimation from single images. In *ICCV*, 2019.

[255] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

[256] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo. Foreground-aware image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2019.

[257] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *CVPR*, 2017.

[258] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3917–3925, 2018.

[259] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10448–10457, 2021.

[260] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6721–6729, 2017.

[261] N. Yang, R. Wang, J. Stückler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018.

[262] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):2872–2893, 2021.

[263] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European conference on computer vision*, pages 467–483. Springer, 2016.

[264] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7508–7517, 2020.

[265] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006.

[266] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[267] W. Yin, Y. Liu, C. Shen, and Y. Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *ICCV*, 2019.

[268] R. Yonetani, K. M. Kitani, and Y. Sato. Recognizing micro-actions and reactions from paired egocentric videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2629–2638, 2016.

[269] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019.

[270] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[271] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4471–4480, 2019.

[272] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[273] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[274] Y. Zeng, J. Fu, and H. Chao. Learning joint spatial-temporal transformations for video inpainting. In *The Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[275] Y. Zeng, Z. Lin, J. Yang, J. Zhang, E. Shechtman, and H. Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.

[276] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao. Learning two-view correspondences and geometry using order-aware network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5845–5854, 2019.

[277] J. Zhang, K. Sunkavalli, Y. Hold-Geoffroy, S. Hadap, J. Eisenman, and J.-F. Lalonde. All-weather deep outdoor lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10158–10166, 2019.

[278] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[279] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[280] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[281] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017.

[282] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *CVPR*, 2019.

[283] S. Zhao, J. Cui, Y. Sheng, Y. Dong, X. Liang, E. I. Chang, and Y. Xu. Large scale image completion via co-modulated generative adversarial networks. *arXiv preprint arXiv:2103.10428*, 2021.

[284] S. Zhao, H. Fu, M. Gong, and D. Tao. Geometry-aware symmetric domain adaptation for monocular depth estimation. In *CVPR*, 2019.

[285] W. Zhao, S. Liu, Y. Shu, and Y.-J. Liu. Towards better generalization: Joint depth-pose learning without posenet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9151–9161, 2020.

[286] Y. Zhao, C. Barnes, Y. Zhou, E. Shechtman, S. Amirghodsi, and C. Fowlkes. Geofill: Reference-based image inpainting with better geometric understanding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1776–1786, 2023.

[287] Y. Zhao, S. Kong, and C. Fowlkes. Camera pose matters: Improving depth prediction by mitigating pose distribution bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15759–15768, 2021.

[288] Y. Zhao, S. Kong, D. Shin, and C. Fowlkes. Domain decluttering: Simplifying images to mitigate synthetic-real domain shift and improve depth estimation. In *CVPR*, 2020.

[289] Y. Zhao, H. Ma, S. Kong, and C. Fowlkes. Instance tracking in 3d scenes from egocentric videos. *arXiv preprint arXiv:2312.04117*, 2023.

[290] Y. Zhao, Y. Tian, C. Fowlkes, W. Shen, and A. Yuille. Resisting large data variations via introspective transformation network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3080–3089, 2020.

[291] C. Zheng, T.-J. Cham, and J. Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *ECCV*, 2018.

[292] C. Zheng, T.-J. Cham, and J. Cai. Pluralistic image completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1438–1447, 2019.

[293] L. Zheng, Y. Yang, and A. G. Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, 2016.

[294] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[295] C. Zhou, Z. Luo, Y. Luo, T. Liu, L. Pan, Z. Cai, H. Zhao, and S. Lu. Pttr: Relational 3d point cloud object tracking with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8531–8540, 2022.

[296] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[297] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[298] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[299] Y. Zhou, C. Barnes, E. Shechtman, and S. Amirghodsi. Transfill: Reference-guided image inpainting by merging multiple color and spatial transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2266–2276, 2021.

[300] J. Zhu, S. Lai, X. Chen, D. Wang, and H. Lu. Visual prompt multi-modal tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9516–9526, 2023.

[301] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[302] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

[303] S. Zhu, C. Li, C. Change Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4998–5006, 2015.

[304] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.

[305] C. Zou, A. Colburn, Q. Shan, and D. Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *CVPR*, 2018.

Appendix A

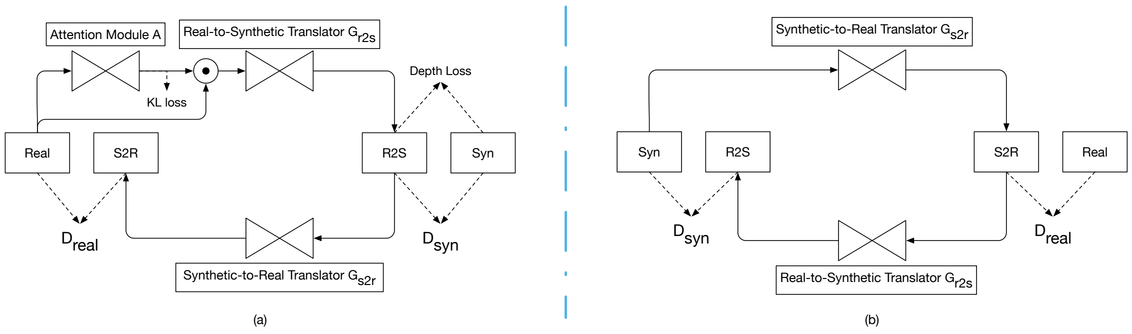
Chapter 2 Supplemental Material

This document supplements Chapter 2 with detailed training diagrams of how we initialize/pretrain the inpainting module $\mathcal{I}$ and attention module $\mathcal{A}$ in our modular coordinate descent algorithm.

A.1 Detailed Training Diagrams of $\mathcal{A}$ and $\mathcal{I}$

To provide a clear idea of how we (pre)train our modules, we present two diagrams, the attention module $\mathcal{A}$ and inpainting module $\mathcal{I}$. For others, we train the real-to-synthetic style translator $\mathcal{T}$ by simply using the CycleGAN pipeline [301]. To train the depth predictor module $\mathcal{D}$, we train it simply using depth regression loss.

The training diagram of our attention module $\mathcal{A}$ is presented in Fig. A.1. Note that $\mathcal{A}$ only appears in the left panel Fig. A.1 (a), which means $\mathcal{A}$ only learns where to mask out in real images. We do not apply this to synthetic data, as synthetically rendered images are clean without clutters.



Attention Module A
Real-to-Synthetic Translator G_{r2s}
KL loss
Depth Loss
Real
S2R
R2S
Syn
D_{real}
D_{syn}
Synthetic-to-Real Translator G_{s2r}
(a)
Synthetic-to-Real Translator G_{s2r}
Syn
R2S
S2R
Real
D_{syn}
D_{real}
Real-to-Synthetic Translator G_{r2s}
(b)

Figure A.1: Detailed training diagram of our attention module $\mathcal{A}$. Note that $\mathcal{A}$ only shows in the real-to-synthetic cycle, e.g., the part (a) in the diagram. The intuition behind two asymmetric cycles is that $\mathcal{A}$ should remove clutters in real samples instead of clean synthetic images.

Our detailed training diagram of module $\mathcal{I}$ is shown in Fig. A.2. The attention module $\mathcal{A}$ and the style translator $\mathcal{T}$ are pretrained models and we color them in red for the purpose of indication. Note that the output of $\mathcal{I}$ is the intermediate inpainting results and our final reconstructed images still follow Eqn. 2.3 in Chapter 2.

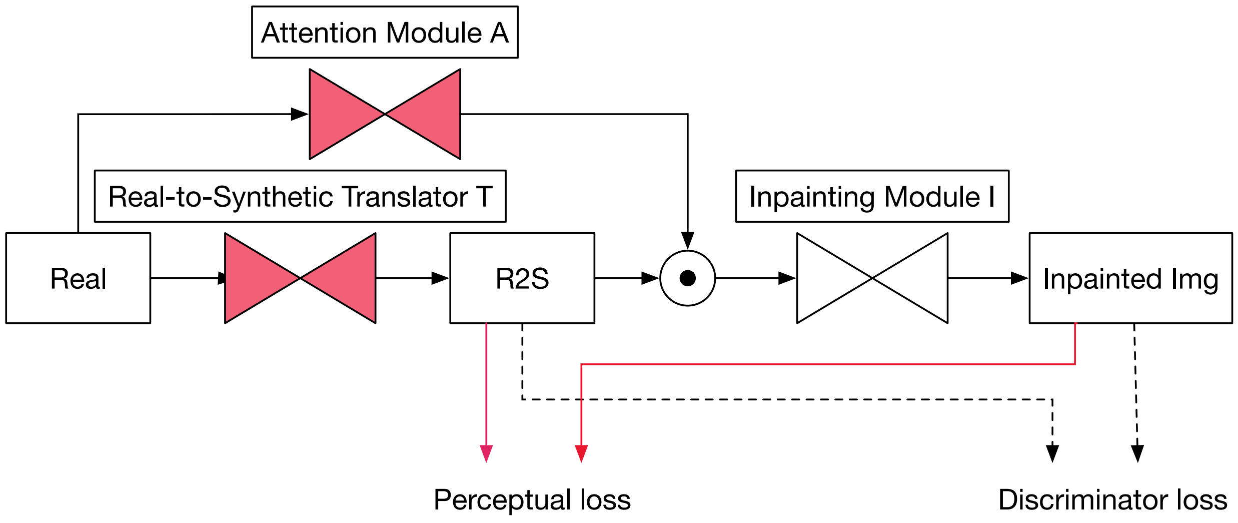


Figure A.2: Detailed training diagram of our inpainting module $\mathcal{I}$. Red blocks in the figure indicates that they are pretrained modules, *i.e.*, the attention module $\mathcal{A}$ and style translator $\mathcal{T}$.

Appendix B

Chapter 3 Supplemental Material

This document supplements Chapter 3 with additional details in experiments, including RGB and depth preprocessing steps, training the camera pose prediction models, our evaluation protocol, and the ScanNet camera pose distribution.

B.1 Additional Details in Experiments

B.1.1 Image and Depth Preprocessing

All input RGB images are first normalized to the range of $[-1.0, 1.0]$ and then resized to 240×320 before feeding into CNNs. Note that resizing images to 240×320 does not change their original aspect ratios. For better training, as a preprocessing step on the depth [21, 85], we apply the following operation to rescale depth maps y to get a normalized map y':

$$y' = (\frac{y - E_{min}}{E_{max} - E_{min}} - 0.5) * 2.0, \tag{B.1}$$

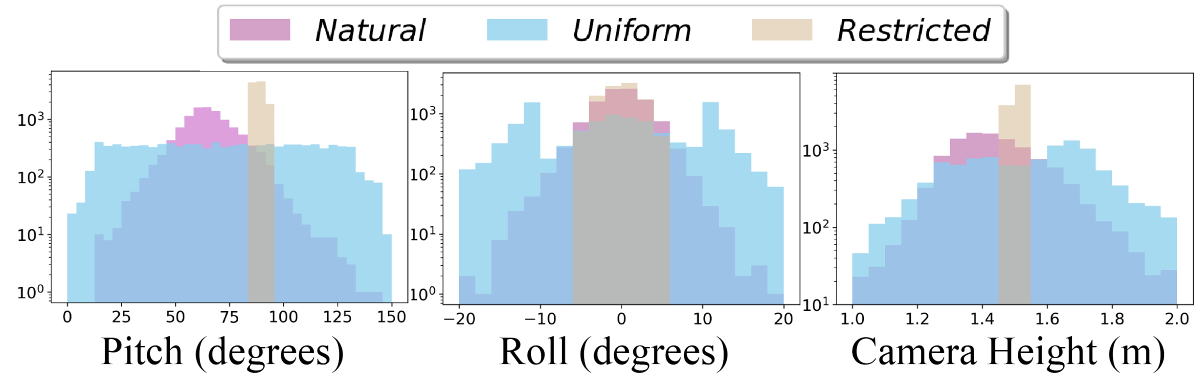


Figure B.1: Distribution of pitch, roll and camera height for three subsets of images from ScanNet. From the *Natural* subset, we observe the ScanNet dataset also has a naturally biased distribution in both pitch, roll and camera height. Please refer to Section 3.5 in Chapter 3 on how we construct these three subsets.

where $E_{min} = 1.0$ and $E_{max} = 10.0$ are the minimum and maximum evaluation values, respective. The above operation is a map from $[1.0, 10.0]$ to $[-1.0, 1.0]$. In the literature, it is reported the model can be trained better in this scale range [291, 284]. We only compute the loss for pixels that have depth values between 1.0 and 10.0 meters. We evaluate the depth prediction on the original depth scale. To do so, we apply an inverse operation of Eq. B.1 to the predicted depth maps. Moreover, we also only evaluate the depth that lies in [1, 10] meters.

B.1.2 Pose Prediction Network

When camera poses are not available during testing, we train a camera pose predictor that predicts camera pitch θ, roll ω and height h for CPP encoding (i.e., the CPP$_{pred}$ model). We build the pose predictor over ResNet18 structure with a new top layer that outputs a 3-dim vector to regress pitch, roll, and camera height. During training, we load the ImageNet pretrained weights and finetune the weights for pose predictions with L1 loss.

B.1.3 Evaluation Protocol

The depth evaluation range in this work is from 1.0m to 10.0m for both InteriorNet and ScanNet. For each method, we save a checkpoint every 10 epochs and select the checkpoint that produces the smallest average L1 loss on the validation set to report the performance.

B.1.4 ScanNet Camera Pose Distribution

The camera pose distribution of subsets in ScanNet is shown in Fig. B.1. While it is hard to sample a subset with exactly uniform distribution w.r.t to all attributes (i.e., pitch, roll, and camera height), we sample the *Uniform* subset with the priority of pitch, roll, height from high to low. As these subsets differ a lot in terms of camera pose distribution, they serve our study w.r.t camera distribution bias.

Appendix C

Chapter 4 Supplemental Material

This document supplements Chapter 4 with additional details, including convergence criteria of our optimization and average running time of each step in GeoFill.

C.1 Convergence Criteria

The convergence criteria define when the optimization should stop. Our optimization halts the loop at a given scale and continues to the next scale if the following condition is met or the predefined maximum number of iteration is achieved. The formula below measures the objective function value changes within the last m iterations.

$$\epsilon_i = \frac{|\sum_{i-(m/2)-1}^{i} l_i - \sum_{i-(m/2)}^{i-m-1} l_i|}{\sum_{i-(m/2)-1}^{i} l_i}, \tag{C.1}$$

where i represents the i^{th} iteration. If ϵ_i is smaller than a predefined ϵ_{opt}, we assume the objective function has converged. Since we adopt a coarse-to-fine optimization strategy, we check the same condition at every level of the pyramid. In other words, we move to the finer scale level only if Eqn. C.1 is met or maximum number of iterations at the current level is reached. We also keep track of the optimal parameters at each level and use them as the initialization in the next level. In practice, we set the convergence threshold ϵ_{opt} to 10^{-6} for all levels. The number of loss values to track in computing convergence criteria is $m = 10$.

C.2 Average Running Time of GeoFill

We randomly sampled 50 images at 1280x720 pixels and compute the average time of each step. Monocular depth estimation takes 3.83s, sparse correspondence estimation takes 0.596s, triangulation takes 0.0009s, initial relative pose takes 0.0052s, joint optimization takes 58.2s, mesh rendering takes 1.03s, refinement and merging step takes 2.53s. The reported time uses default parameters described in the experiment section. Although the joint optimization step takes up the vast majority of the time, its current implementation is naive and not optimized. If desired, various engineering optimizations could be made such as using a custom kernel with proper low-level optimizations such as fusion for the renderer instead of a naive pure PyTorch implementation, using FP16 mode, using only the sparse edge map pixels during optimization (these are quite sparse so significant acceleration should be possible), using second-order optimization techniques that could potentially converge in fewer steps, carefully tuning input resolution, number of pyramid levels, iteration limits, break thresholds, etc. We considered these to be lower-level engineering details that we did not focus on in our current implementation, since we were focusing more on research aspects.

Appendix D

Chapter 5 Supplemental Material

This document supplements Chapter 5 with dataset documentation and intended uses.

D.1 Datasheet

We follow the datasheet proposed in [77] for documenting our benchmark dataset.

Motivation

For what purpose was the dataset created?

This dataset was created to study the problem of instance tracking in 3D from egocentric videos. We find current egocentric sensor data from AR/VR devices cannot support the study of our benchmark problem.

Composition

What do the instances that comprise the dataset represent?

Raw egocentric video sequences, object enrollments for each object instance, and annotation files.

How many instances are there in total?

There are 50 video sequences with an average length of over 10K frames, 220 unique object instances with two types of enrollment information, and three types of annotations.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?

Yes.

What data does each instance consist of?

Please check Section 5.3.2 in Chapter 5 for details.

Is there a label or target associated with each instance?

Yes. Please check Section 5.3.2 in Chapter 5 for details.

Is any information missing from individual instances?

No.

Are relationships between individual instances made explicit?

Videos captured in the same scene share a similar surrounding environment but different activities. Object instances are related to the task performed in the video. No explicit relationships between different object instances in the same video.

Are there recommended data splits?

Yes. The entire benchmark dataset focuses on evaluation only. Models should be pretrained on other data sources. Please check Section 5.3.1 in Chapter 5 for details.

Are there any errors, sources of noise, or redundancies in the dataset?

Yes. There are noises in camera poses and depth maps. The source of camera pose noise is from the camera localization from HoloLens2, especially under large head motion. The depth map noises are from phase wrapping. But this noise can be easily recovered with rendered depth using mesh or exploring existing unwrapping algorithms.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?
Yes. The dataset is self-contained.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)?
No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?
No.

Does the dataset identify any subpopulations (e.g., by age, gender)?
No.

Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset?
No. We have carefully examined the data and ensure no personally identifiable information is included.

Does the dataset contain data that might be considered sensitive in any way (e.g., data that reveals racial or ethnic origins, sexual orientations, religious beliefs, political opinions or union memberships, or locations; financial or health data; biometric or genetic data; forms of government identification, such as social security numbers; criminal history)?
No..

Any other comments?

N/A

Collection Process

How was the data associated with each instance acquired?

The raw video sequences are collected with HoloLens2. The pre-enrollment information is captured with the iPhone 13 Pro. The rest data, i.e, annotations and online enrollment information, are acquired from human annotators.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?

The dataset is collected with open-source hl2ss [1] using HoloLens2. The pre-enrollment images are captured with the iPhone 13 Pro. For more details please check Section 5.3 in Chapter 5.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

N/A

Does the dataset relate to people?

Yes. The dataset includes video sequences of the first-person view of individuals performing the daily activity.

Were any ethical review processes conducted (e.g., by an institutional review board)?

Yes. Data collection protocol was registered with the appropriate institutional review board (IRB).

Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources (e.g., websites)?

The raw video sequences are collected when the camera wearer performs the daily task.

Were the individuals in question notified about the data collection?

Yes.

Did the individuals in question consent to the collection and use of their data?

Yes.

If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses?

No.

Has an analysis of the potential impact of the dataset and its use on data subjects (e.g., a data protection impact analysis) been conducted?

No. All annotations are on objective world states with no subjective opinions or arguments involved.

Any other comments?

N/A

Preprocessing/Cleaning/Labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)?

No.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)?

Yes. We will provide both the raw data and annotations.

Is the software used to preprocess/clean/label the instances available?

No.

Any other comments?

N/A

Uses

Has the dataset been used for any tasks already?

No.

What (other) tasks could the dataset be used for?

Our benchmark dataset also supports the study of other 3D scene understanding problems from egocentric videos, such as SLAM, depth estimation, and camera localization.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?

No.

Are there tasks for which the dataset should not be used?

The usage of this dataset should be limited to the scope of instance tracking in 3D and geometric scene understanding from egocentric videos.

Any other comments?

N/A

Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?

Yes. The dataset will be made publicly available and third parties are allowed to distribute the dataset.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)?

The dataset will be publicly available on both Github repo and the website and stored on the cloud store, e.g., Google drive or Amazon S3.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

We release our benchmark dataset and code under MIT license.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

No.

Any other comments?

N/A

Maintenance

Is there an erratum?

No. When errors are confirmed, we will announce erratum on the platform where dataset is publicly hosted, i.e., either the Github repo or the website.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances')?

Yes. We hope to bring more diversity to the dataset, such as more object instance and scenes.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)?

No.

Will older versions of the dataset continue to be supported/hosted/maintained?

Yes. All versions of the dataset will be publicly available.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

Please email us if you are interested in extending or contributing to the dataset.

Any other comments?

N/A