Supplementary Material for "On the Coupling of Depth and Egomotion Networks for Self-Supervised Structure from Motion"

Brandon Wagstaff[1], Valentin Peretroukhin[2], and Jonathan Kelly[1†]

I. Network and Training Details

a) Egomotion network: Table III describes the layers of the egomotion network that were used. Following each convolutional layer, weight standardization (WS) [1] and group normalization (GN) [2] are applied prior to the ReLU activation except the last, which must be able to produce unrestricted values.

b) Depth network: The depth network is a U-Net [3] encoder-decoder, with the encoder being a ResNet18 network [4]. Table I describes the layers of our depth network decoder, which is based on Monodepth2 [5]. Here, the output from the Resnet18 encoder (enc5) is upsampled, and passed through multiple convolution and upsampling layers, while being merged with additional skip connections (enc4, enc3, enc2). Rather than directly predicting the disparity (inverse depth) from these layers at multiple scales, we adopt the "dense connected prediction" (DCP) convolutional layers from DNet [6] to merge features from all of the resolution scales prior to regressing the inverse depth. Table II describes how the various layers from the depth decoder (i.e., iconv3, iconv2, iconv2) are passed into DCP convolution layers, which have sigmoid activations used to output disparity predictions whose values are within the range of $[0, 1]$. Note the final layer concatenates the dcp3, dcp2, dcp1 layer outputs to produce a multiscale disparity prediction.

The inverse depth output $\mathbf{o}_t$ is then inverted, and constrained to be within the range $[D_{min}, D_{max}]$:

$$\mathbf{D}_t^{-1} = \frac{1}{D_{min}} + \left(\frac{1}{D_{max}} - \frac{1}{D_{min}}\right)\mathbf{o}_t.$$

For our experiments, we set these hyperparameters specifically for the dataset, and note that in the future this parameter can be learned [7]. We use $[0.06, 2.67]$ for KITTI odometry, $[0.1, 2.67]$ for KITTI Eigen, and $[0.03, 3]$ for ScanNet.

c) More training details: Our iterative egomotion network is trained with minimal applied changes to the baseline training procedure. To ensure stability, we only apply one iteration in the first epoch, and then increase to the specified number of iterations for the rest of the training. When training using multiple egomotion iterations, we only compute

[1]Brandon Wagstaff and Jonathan Kelly are with the Space & Terrestrial Autonomous Robotic Systems (STARS) Laboratory at the University of Toronto Institute for Aerospace Studies (UTIAS), Toronto, Ontario, Canada, M3H 5T6. Email: <first name>.<last name>@robotics.utias.utoronto.ca

[2]Valentin Peretroukhin is with the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology.

[†]Jonathan Kelly is a Vector Institute Faculty Affiliate. This research was supported in part by the Canada Research Chairs program.

TABLE I: The network layers for our depth decoder network. To merge the current layer with the ResNet18 layers, we upsampled (via nearest neighbour interpolation) them prior to applying the `upconv` convolution. Upsampling is indicated by ↑.

Depth Decoder					
Layer	**k**	**s**	**chns**	**input**	**activation**
upconv5	3	1	512/256	↑enc5	ELU
iconv5	3	1	256/256	upconv5	ELU
upconv4	3	1	256/128	↑iconv5	ELU
iconv4	3	1	128/128	upconv4 + enc4	ELU
upconv3	3	1	128/64	↑iconv4	ELU
iconv3	3	1	64/64	upconv3 + enc3	ELU
upconv2	3	1	64/64	↑iconv3	ELU
iconv2	3	1	64/64	upconv2 + enc2	ELU
upconv1	3	1	64/32	↑iconv2	ELU
iconv1	3	1	32/32	upconv1	ELU

TABLE II: The DCP components of our decoder, that merge features from all scales to produce the disp1 (full resolution) prediction.

DCP Layers					
Layer	**k**	**s**	**chns**	**input**	**activation**
dcp3	3	1	64/8	iconv3	ELU
disp3	3	1	8/1	dcp3	sigmoid
dcp2	3	1	64/8	iconv2	ELU
disp2	3	1	16/1	↑dcp3, dcp2	sigmoid
dcp1	3	1	32/8	iconv1	ELU
disp1	3	1	24/1	dcp3, ↑dcp2, ↑dcp1	sigmoid

TABLE III: The network layers for our egomotion network. **k** refers to the kernel size, **s** refers to the stride, **chns** refers to the input/output channels.

Egomotion Network					
Layer	**k**	**s**	**chns**	**input**	**Activation**
conv1	3	1	6/16	stacked-img-pair	WS+GN+ReLU
conv2	3	2	16/32	conv1	WS+GN+ReLU
conv3	3	3	32/64	conv2	WS+GN+ReLU
conv4	3	2	64/128	conv3	WS+GN+ReLU
conv5	3	2	128/256	conv4	WS+GN+ReLU
conv6	3	2	256/256	conv5	WS+GN+ReLU
conv7	3	2	256/256	conv6	WS+GN+ReLU
avgpool				conv7	-
poseconv	1	1	256/6	avgpool	-

Method	Seq. 09		Seq. 10	
	t_{err} (%)	r_{err} (°/100m)	t_{err} (%)	r_{err} (°/100m)
Pretrained	1.19	0.30	1.34	0.37
No Pretraining	1.17	0.29	1.81	0.54

TABLE IV: Demonstrating the impact of using Oxford RobotCar pretraining for our KITTI odometry model. There is a small difference in the overall accuracy that comes from additionally using more data.

the loss once using the *final* egomotion prediction, rather than for the egomotion prediction after every iteration. This speeds up training, and allows the iterative approach to be easily incorporated into the standard self-supervised depth and egomotion pipeline. We note that multiple backpropagation steps (for every forward pass through the egomotion network) would better enforce accuracy across all iterations, rather than only enforcing accuracy for the final prediction, but was less feasible to implement. A simpler way to enforce accuracy across all iterations would be to randomize the number of egomotion iterations per minibatch.

d) Oxford Robotcar pretraining: We initialize the depth and egomotion network weights via self-supervised training on the Oxford Robotcar dataset. Here, following the same training procedure (i.e., same loss function, same network structure, same hyperparameters), we train for 15 epochs on sequences `2014-11-18-13-20-12`, `2015-07-08-13-37-17`, `2015-07-10-10-01-59`, `2015-08-12-15-04-18`; we use the first and second subsequences of `2014-11-18-13-20-12` for validation and testing respectively. Note that this pretraining does not significantly affect our results, but simply ensures stability during training, as initializing from random weights is not always guaranteed to converge. Starting from an initialized model (that we can use for all of our KITTI and ScanNet experiments) simplifies our training scheme. We demonstrate that the added Oxford pretraining data does not result in a significant performance increase compared with solely using training data (see Table IV)

II. Additional results

We include additional details and results for our experiments. First, we discuss how the 1D loss curves were generated, and provide additional examples of such curves. Then, we include more examples of predictions from our depth network. Next, we provide further details and results for our pose perturbation experiment, and for our depth scaling experiment. Finally, we include another ablation study for the KITTI depth results on the Eigen test split.

A. Generation of 1D loss curves

For any source/target image pair, given a fixed target depth prediction, we visualize the photometric reconstruction loss surface by varying one of the six degrees of freedom of the egomotion prediction via a grid search. We demonstrate that the loss curve has a clear minimum that our egomotion network identifies by iteratively updating its prediction until the error is minimized. For image datasets generated with a forward facing camera on a ground vehicle (e.g., KITTI), there are two primary degrees of freedom: the forward translation (z-axis), and the yaw angle (about the y-axis). Thus, our experiments focus on these two degrees of freedom. To sample egomotion values, we generate the initial depth and egomotion predictions from our trained networks, and use a grid search to sample poses within a range centered around the initial prediction. For visualization purposes, we only vary a single degree of freedom of the egomotion at a time, while keeping the other values fixed with the initial prediction value. Note that we use DNet rescaling [6] to align the scale factor of the predictions with ground truth prior to generating these curves.

Concretely, one variable within the inter-frame pose prediction $p_0 \in \{t_x, t_y, t_z, \theta_x, \theta_y, \theta_z\}$ is varied using a grid search to produce N values in the range $[p_0 - y, p_0 + y]$. For each value we compute the photometric reconstruction loss; all loss values are plotted to construct the 1D loss curve (as a function of the selected egomotion parameter). For the forward translation t_z, we generate $N = 200$ uniformly sampled values in the range $[t_z - 3t_z, t_z + 3t_z]$. For the yaw (heading) angle, we generate $N = 100$ uniformly sampled values in the range $[\theta_y - 0.02, \theta_y + 0.02]$ radians. See Figure 1 for additional example loss surfaces. Here, we include illustrations of how the loss curves shift during the inference-time PFT. To generate these curves, we repeat this 1D loss curve generation procedure after every optimization epoch.

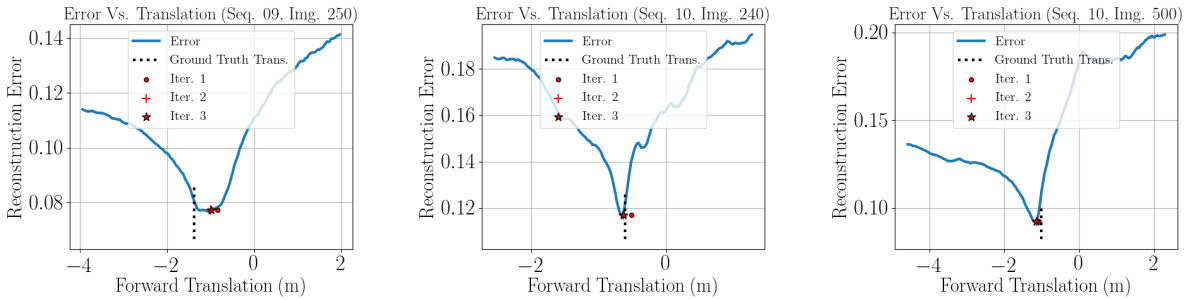
B. Additional figures

We provide several more examples of our depth predictions, and the resulting error images in Figures 2 and 3. Additionally, we include some "failure modes" in Figure 4. Specifically, our optimization method occasionally removes thin objects (like poles or trees) and will shift moving objects in such a way that minimizes the loss, but does not represent reality. We aim to address these drawbacks in future work.

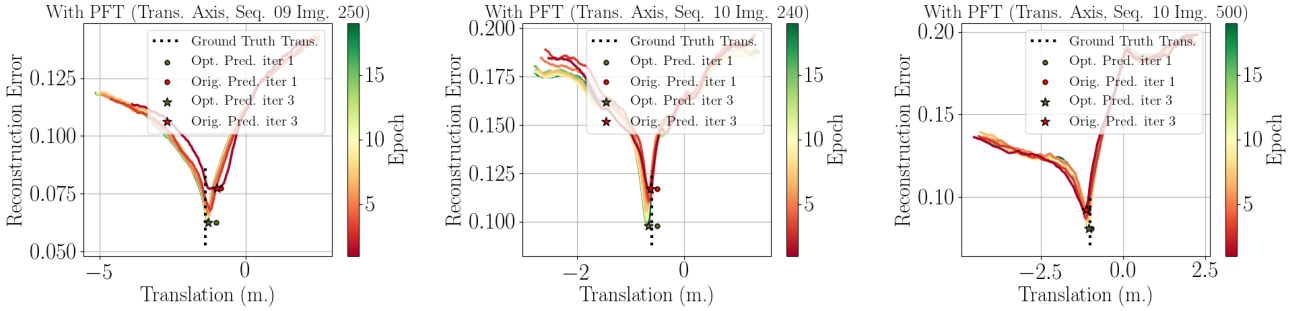
C. Additional pose perturbation experiment details

In this experiment, we perturb the first iteration egomotion prediction by adding uniformly distributed noise to the translation and yaw values, and then proceed with additional iterations that take as input the (perturbed) warped source image. The initial pose perturbation causes the reconstruction image to become further away from the target image, which the subsequent iterations must correct for in order to re-align the warped source image with the target image. We show that despite the initial erroneous operating point, the subsequent forward passes through our network produce corrections that bring the egomotion prediction back to the minimum.

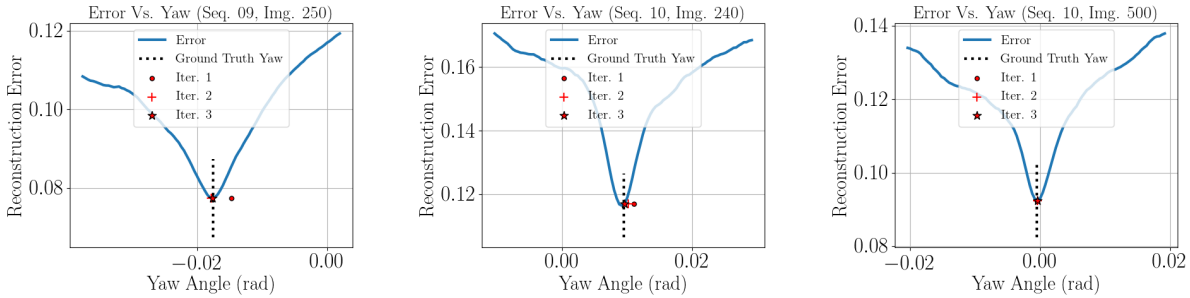
Concretely, we apply pose perturbations (in the forward translation direction, and along the yaw axis) to the initial egomotion prediction. The translation perturbation is sampled from a uniform distribution in the range $[-\alpha, \alpha]$, where $\alpha \in \{0, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5\}$ meters, and the yaw



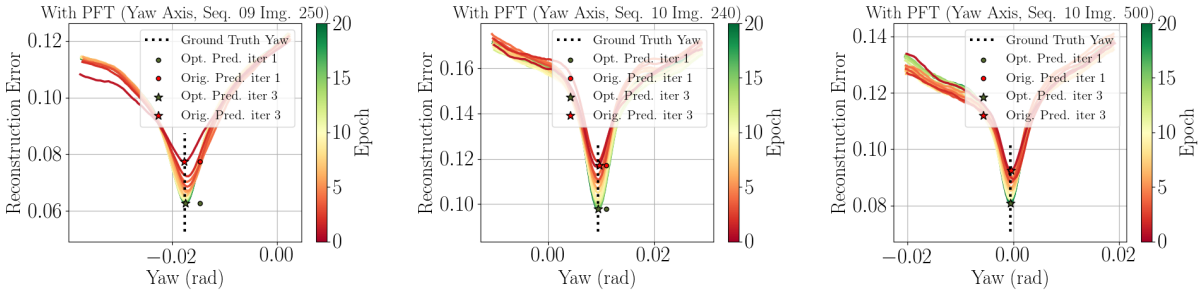
(a) 1D loss curves (in the forward translation axis) generated using the original (non-optimized) depth network prediction



(b) 1D loss curves (in the forward translation axis) showing improved convergence due to the refined depth prediction as a result of using our PFT strategy.

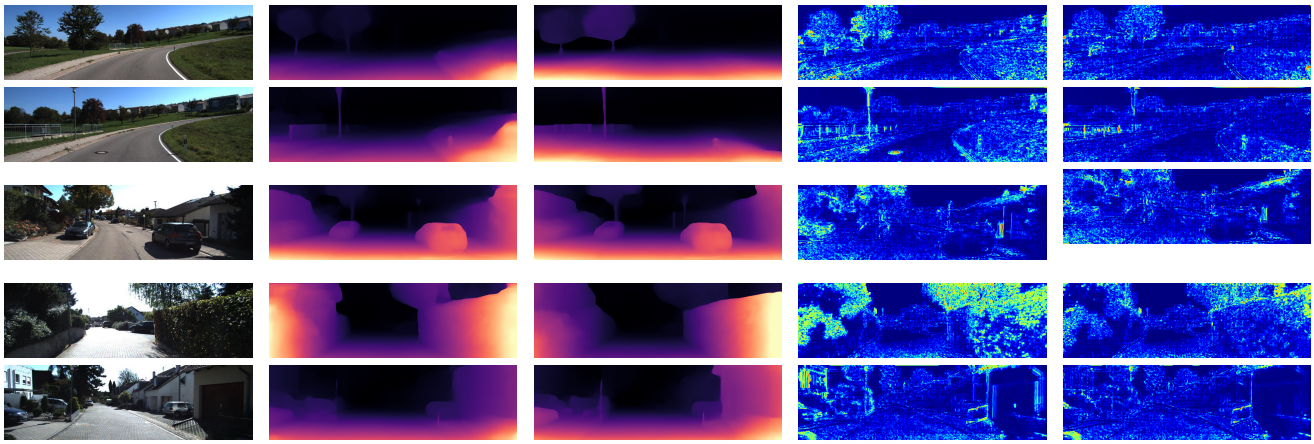


(c) 1D loss curves (in the yaw axis) generated using the original (non-optimized) depth network prediction



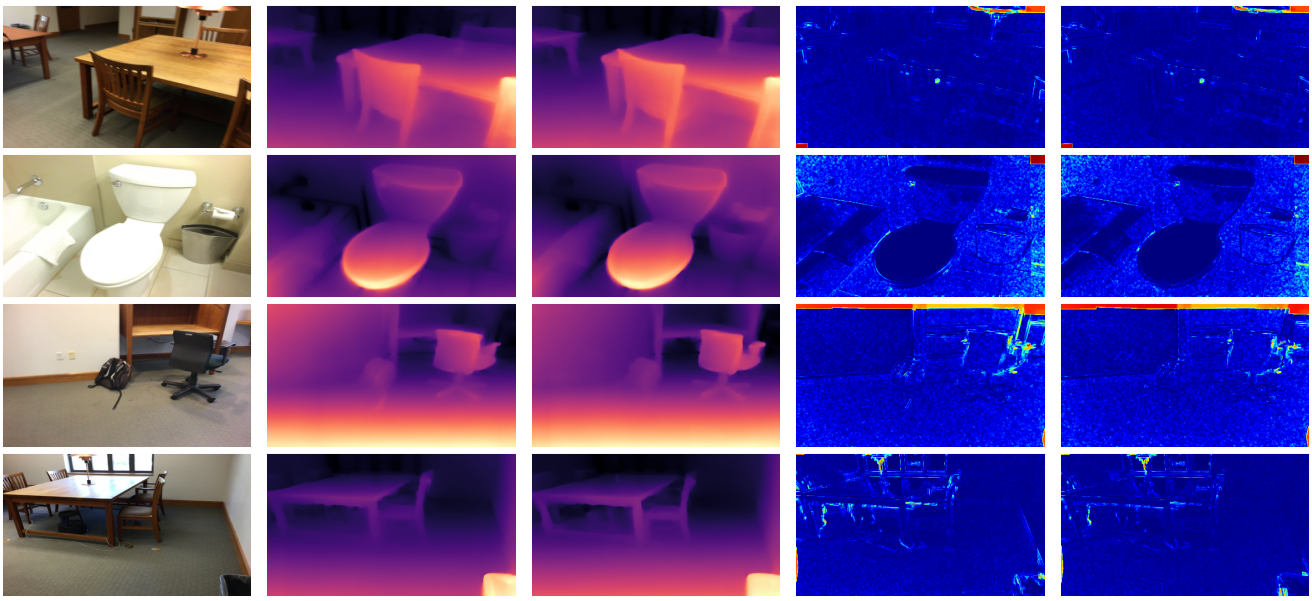
(d) 1D loss curves (in the yaw axis) showing improved convergence due to the refined depth prediction as a result of using our PFT strategy.

Fig. 1: 1D loss surface experiment demonstrating that the iterative egomotion prediction converges to minimum of the loss function, and inference-time PFT of the depth network further minimizes the overall loss curve.



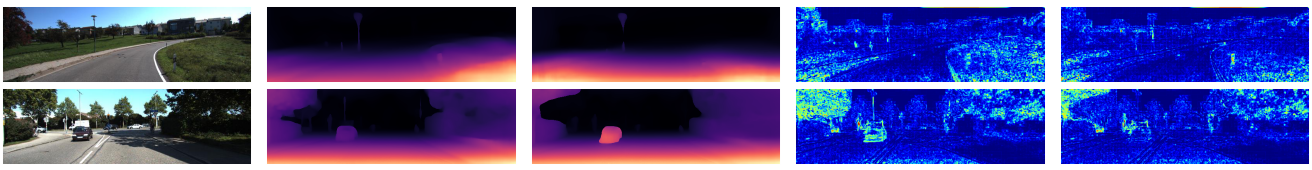
(a) Image (b) Depth (w/o PFT) (c) Depth (w PFT) (d) Error (w/o PFT) (e) Error (w PFT)

Fig. 2: Additional depth predictions for KITTI dataset images. We show the original network prediction, and the optimized result after applying our PFT strategy



(a) Image (b) Depth (w/o PFT) (c) Depth (w PFT) (d) Error (w/o PFT) (e) Error (w PFT)

Fig. 3: Additional depth predictions for ScanNet dataset images. We show the original network prediction, and the optimized result after applying our PFT strategy.



(a) Image (b) Depth (w/o PFT) (c) Depth (w PFT) (d) Error (w/o PFT) (e) Error (w PFT)

Fig. 4: failure mode examples for our PFT strategy: in the first case, some thin objects are removed; in the second case, the optimization shifts the car's depth closer to the camera to account for its forward motion

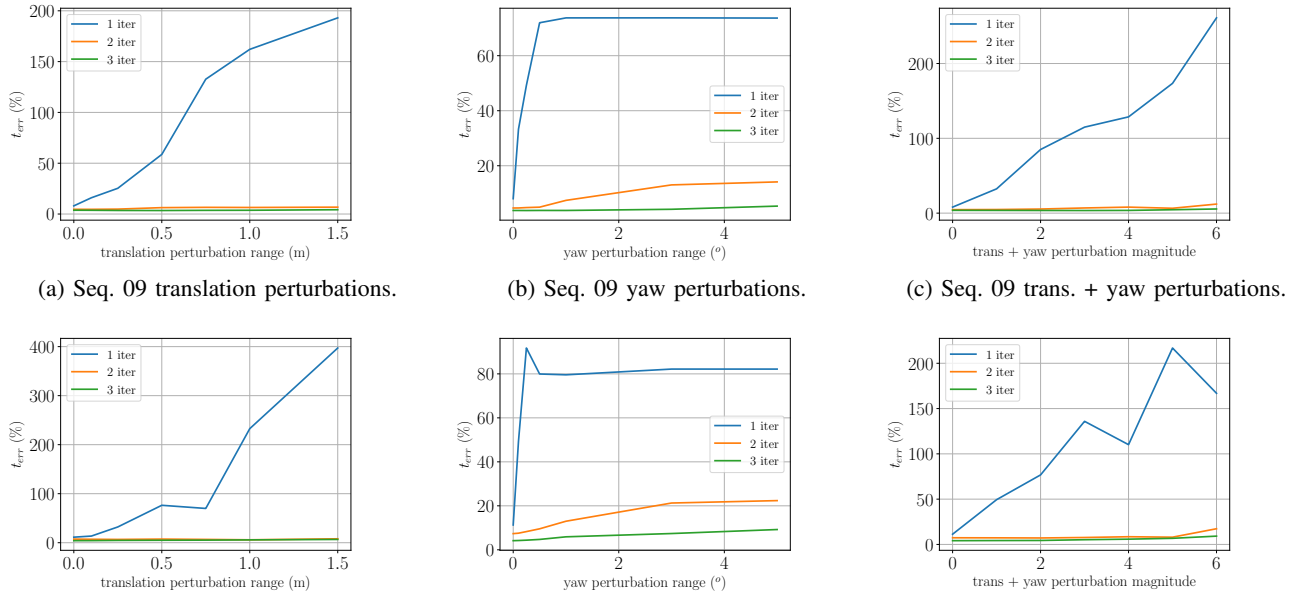
perturbation is sampled from a uniform distribution in the range $[-\beta, \beta]$, where $\beta \in \{0, 0.1, 0.25, 0.5, 1, 3, 5\}$ degrees. A random perturbation is applied to every sample within KITTI test sequences `09` and `10`, and we report the resulting errors (averaged across the sequence) in Figure 5. Here, we include the effects of applying translation and rotation perturbations independently, and then in combination. As a baseline, we report the error for our single iteration model with random noise added to its one-shot prediction (this is the error range we expect to see if our iterative network cannot correct for the added error).

D. Additional depth scaling experiment details

In this experiment, we simulate scale drift between the depth and egomotion networks by varying the scale factor s of the depth predictions ($\mathbf{D}_{scaled} = s\mathbf{D}_{pred}$). We demonstrate that through the coupling of our networks (through iterative egomotion estimation), the scale drift is mitigated. In this experiment, we apply a scaling factor $s \in \{0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$ to all depth predictions in our test sequences. We compute the change in translation norm (relative to the translation norm with an applied scale factor of 1) of our iterative egomotion network predictions. As expected, the relative change in the predicted translation norm is approximately equal to the depth scaling factor. We report this result Figures 6a and 6b. In Figures 6c and 6d we also apply the same scale factor to the ground truth position values, and report the t_{err} between our predictions and the rescaled ground truth. With our iterative method, we see that the error remains relatively constant. In contrast, without being able to account for the depth scale drift with the single iteration network, the error increases with the magnitude of the scale drift, since the egomotion prediction remains fixed while effective ground truth motion changes as the scene depth is rescaled. Note that in this experiment, we do not apply any scale alignment using ground truth, since this would account for the scale inconsistency by utilizing ground truth information. Instead, we use the online rescaling approach ([6]) to rescale the egomotion predictions based on the observed camera height (compared with the known camera height) prior to reporting the translation error.

E. KITTI Depth Ablation

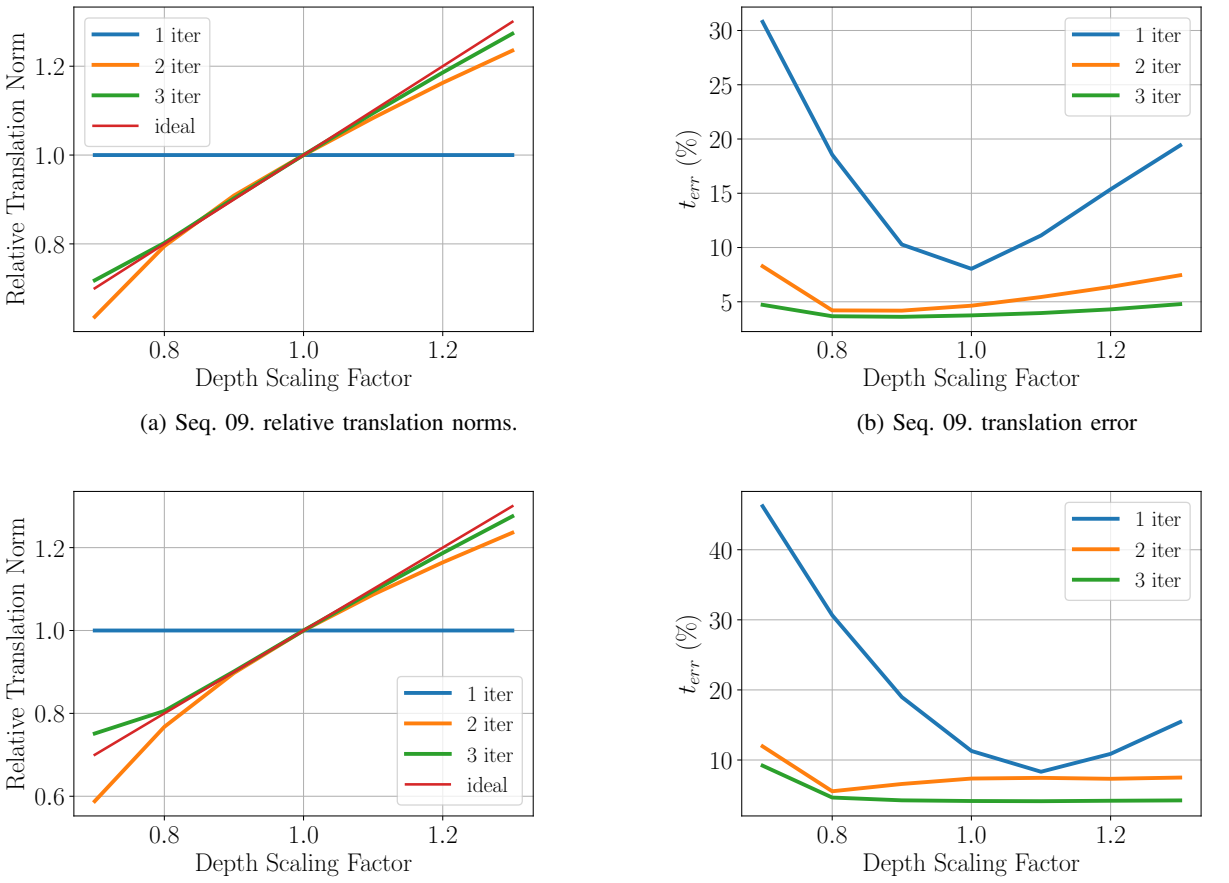
Table V provides an ablation study for the depth evaluation on the KITTI (Eigen) test split. We see that applying PFT leads to an improvement in all cases, and is more effective than applying iteration in this case. This differs from our ablation study on ScanNet, where we see that iteration is crucial for improving accuracy. We postulate that this is due to the camera motion being more constrained for KITTI compared with ScanNet; the 6-DOF camera pose changes of ScanNet are more difficult to learn than the more simple forward motion of the KITTI dataset. This simplified motion allows our one-iteration egomotion network to achieve a high level of accuracy, and as a result causes additional iterations to not have a significant amount of impact. In contrast, due to the complexity of motion within ScanNet, more iterations are required to improve generalization of the egomotion network (which is crucial to have for refining depth predictions via PFT).



(a) Seq. 09 translation perturbations. (b) Seq. 09 yaw perturbations. (c) Seq. 09 trans. + yaw perturbations.

(d) Seq. 10 translation perturbations. (e) Seq. 10 yaw perturbations. (f) Seq. 10 trans. + yaw perturbations.

Fig. 5: Full results for the pose perturbation experiments. Through coupling, our iterative egomotion network can account for large pose perturbations that would otherwise degrade the egomotion accuracy.



(a) Seq. 09. relative translation norms. (b) Seq. 09. translation error

(c) Seq. 10. relative translation norms. (d) Seq. 10. translation error

Fig. 6: Results of the depth scaling experiment. The iterative egomotion network can accurately account for the change in scale of the depth prediction.

TABLE V: Ablation study for the KITTI Eigen test split. As the number of egomotion optimizer iterations increases, the accuracy after test-time depth optimization increases.

# Egomotion Iter.	Test-time Depth Opt.	Error ↓				Accuracy ↑		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1	—	0.123	0.934	4.990	0.200	0.860	0.956	0.980
2	—	0.119	0.934	4.959	0.197	0.868	0.957	0.980
3	—	0.119	0.929	4.935	0.197	0.868	0.957	0.980
4	—	0.117	0.954	4.902	0.196	0.870	0.958	0.981
1	$\boldsymbol{\theta}_D + \boldsymbol{\theta}_E$	0.099	0.784	4.460	0.179	0.896	0.963	0.982
2	$\boldsymbol{\theta}_D + \boldsymbol{\theta}_E$	0.097	0.774	4.440	0.178	0.899	0.963	0.982
3	$\boldsymbol{\theta}_D + \boldsymbol{\theta}_E$	0.095	0.769	4.414	0.177	0.902	0.963	0.982
4	$\boldsymbol{\theta}_D + \boldsymbol{\theta}_E$	0.094	0.783	4.391	0.176	0.903	0.964	0.982
1	$\boldsymbol{\theta}_D$	0.108	0.822	4.583	0.186	0.883	0.959	0.981
2	$\boldsymbol{\theta}_D$	0.099	0.785	4.473	0.182	0.890	0.958	0.981
3	$\boldsymbol{\theta}_D$	0.098	0.775	4.410	0.180	0.894	0.960	0.981
4	$\boldsymbol{\theta}_D$	0.097	0.781	4.385	0.178	0.897	0.961	0.982

REFERENCES

[1] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," *arXiv preprint arXiv:1903.10520*, 2019.

[2] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, 2018, pp. 3–19.

[3] Ronneberger, O. and Fischer, P. and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. Int. Conf. Medical Image Computing Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[5] C. Godard, O. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2019, pp. 3828–3838.

[6] F. Xue, G. Zhuo, Z. Huang, W. Fu, Z. Wu, and M. A. Jr, "Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications," in *Proc. Conf. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2020, pp. 2330–2337.

[7] R. McCraith, L. Neumann, A. Zisserman, and A. Vedaldi, "Monocular Depth Estimation with Self-supervised Instance Adaptation," *arXiv preprint arXiv:2004.05821*, 2020.