Supplementary Material for Self-Supervised Scale Recovery for Monocular Depth and Egomotion Estimation

Brandon Wagstaff, and Jonathan Kelly†

TABLE I: Network architecture of our pose encoder network. **k** refers to the kernel size, **s** refers to the stride, **chns** refers to the input/output channels.

Pose Network					
Layer	**k**	**s**	**chns**	**input**	**Activation**
conv1	3	1	8/16	img+flow	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	-

I. Network Details

A. Pose Encoder Network

Our pose network is shown in Table I. ReLU activations are applied after every convolutional layer (except for the final poseconv layer, as this must be able to regress negative values). Following each convolution, and before the activation, weight standardization (WS) [1] and group normalization (GN) [2] is applied. The initial number of input channels is eight, as we use two RGB images, and the optical flow estimate between them. In line with [3] we found that incorporating optical flow improved the egomotion estimates. We computed the optical flow using the Gunnar-Farnebäck algorithm [4].

B. Depth Encoder Network

We use a standard Resnet18 [5] architecture to extract a feature representation from a single image.

C. Depth Decoder Network

Our decoder architecture is similar in spirit to Monodepth2 [6] and DNet [7]: the main architecture is based on the decoder from Monodepth2, but we incorporate "dense connected prediction" (DCP) layers from DNet to combine features from different scales. We only used the highest resolution scale scales for our experiments.

At each layer of the baseline decoder, the feature space is upsampled by a factor of two within each `upconv` layer, and is added to a previous encoder output via a skip connection. See Table II for more details. The extracted features from the final three layers are passed into the DCP layers; see Table III for more information. Here, the number of channels from the final three `iconv` layers are reduced to eight using a convolutional layer, and are passed through disparity prediction layers[1], which reduce the number of channels to one via a 2D convolution of stride one; a sigmoid activation is applied to constrain the network output to be within [0,1]. To produce the inverse depth estimate at each of these scales, the following transformation is applied:

$$\mathbf{D}_t^{-1} = \frac{1}{D_{min}} + \left(\frac{1}{D_{max}} - \frac{1}{D_{min}}\right) * \mathbf{out}, \qquad (1)$$

which constrains the depth estimates to be in the range $[D_{min}, D_{max}]$.

TABLE II: Network architecture of our depth encoder network. Upsampling (represented with a ↑) was done prior to applying the `upconv` convolution, and was done with nearest neighbour interpolation.

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 III: Description of the DCP layers, which merge lower level features with higher level features prior to applying a disparity 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

D. Plane Segmentation Network

Our plane segmentation network uses a pretrained Resnet18 Encoder. Following this, our decoder is described in Table IV

All authors 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

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

[1]to avoid regressing infinite depths, we output the inverse depth, or "disparity predictions", and invert this network output to produce the depth estimate.

TABLE IV: Plane Segmentation Decoder details.

Plane 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
plane1	3	1	32/1	iconv1	sigmoid

II. Training Details

For all experiments, we use an image resolution of 192x640, and whiten all input images using the ImageNet [8] statistics. All images are augmented by randomly flipping images, and modifying the brightness, contrast, saturation, and hue within the ranges ±0.2, ±0.2, ±0.2, ±0.1 respectively (the same modification is made to all the target and source images within a training sample). For training on KITTI, we use sequences `00`, `02`, `06–08`, `11`, `13–16`, `19`, and leave out `05` for validation, and `09–10` for testing.

Currently, we separate the training of our plane segmentation network from the training of our depth and egomotion networks, because an accurate depth prediction is required to train the plane segmentation network (we use it to extract 3D coordinates from each image coordinate in the image). Therefore, we initially train an *unscaled* depth and egomotion network (with the our baseline, unscaled loss function), and use the unscaled depth predictions from this network to train the ground plane segmentation network. Our final model was chosen by selecting the one that resulted in the lowest validation loss. The hyperparameters for the plane segmentation network training were $\lambda_{plane} = 25$, and $\lambda_{reg} = 0.05$. Fixing λ_{plane}, we reduced λ_{reg} until the plane segmentation masks were conservative enough to cover the majority of the road without erroneously producing high weights for off-plane pixels.

Following the training of the plane segmentation network, we train our *scaled* depth and egomotion networks with our scale recovery loss (along with the traditional photometric reconstruction losses described in the paper). We initialized our networks by training for one epoch with the unscaled (baseline) loss (eq. 7), since our scale recovery loss requires reasonable depth estimates to estimate the scale factor. After one epoch, we incorporated the scale recovery loss term (eq. 18) to begin resolving metric scale. Each training sample consists of three images: a target image, and two source images that are temporally adjacent to the target. We adopt several training improvements Monodepth2 [6] to improve the overall network training. Automasking is used to ignore pixel regions that do not change in intensity between frames, and the per-pixel minimum reprojection loss is computed to account for occluded and out-of-view pixels. Additionally, we use the same multi-scale training scheme (i.e., we evaluate our training loss at each of the depth network's three depth resolutions, but upsample each scale to full resolution before computing the reconstruction loss). In addition to warping each source frame to the target frame, we do the inverse, by computing each source image depth, and warping the target frame to the source frames (and compute the photometric reconstruction loss for each). For our depth consistency loss, we warp each source depth to the target frame, and compare these to the target depth.

We set $\alpha = 0.85, \lambda_P = 1, \lambda_S = 0.05, \lambda_{DC} = 0.14, \lambda_{PC} = 5, \lambda_{DS} = 0.02$, $D_{min} = 1.8$, $D_{max} = 60$ For λ_{TS}, we incrementally increased the loss for the first nine epochs, since we found that immediately starting training with a large weight would significantly degrade the pretrained depth predictions:

$$\lambda_{TS} = 0.6 * (1 + min(2 * epoch, 9)).$$

The ground truth camera height was $h_{gt} = 1.70$ meters.

III. Additional Experimental Results

A. KITTI Odometry Results

We include the top-down trajectory estimates of our training and validation sequences for the KITTI dataset. Figure 1 illustrates sequences `05`, `09`, `10` with our 6-DOF predictions. Additionally we have included a second set of trajectories that use ground-truth orientation (paired with learned translation) in order remove orientation error from the visualization. We see in all cases that our scaled egomotion predictions are generally more accurate than the online rescaling approach from DNet [7].

We also include Table V that features additional VO results on our KITTI training sequences. Table VI depicts additional results for the estimated scale factors of the KITTI Odometry sequences. Namely, we show that the standard deviation of the scale factor is significantly smaller for our (scaled) approach compared with a baseline unscaled approach. This indicates that our proposed scale recovery loss promotes better depth consistency than alternative approaches.

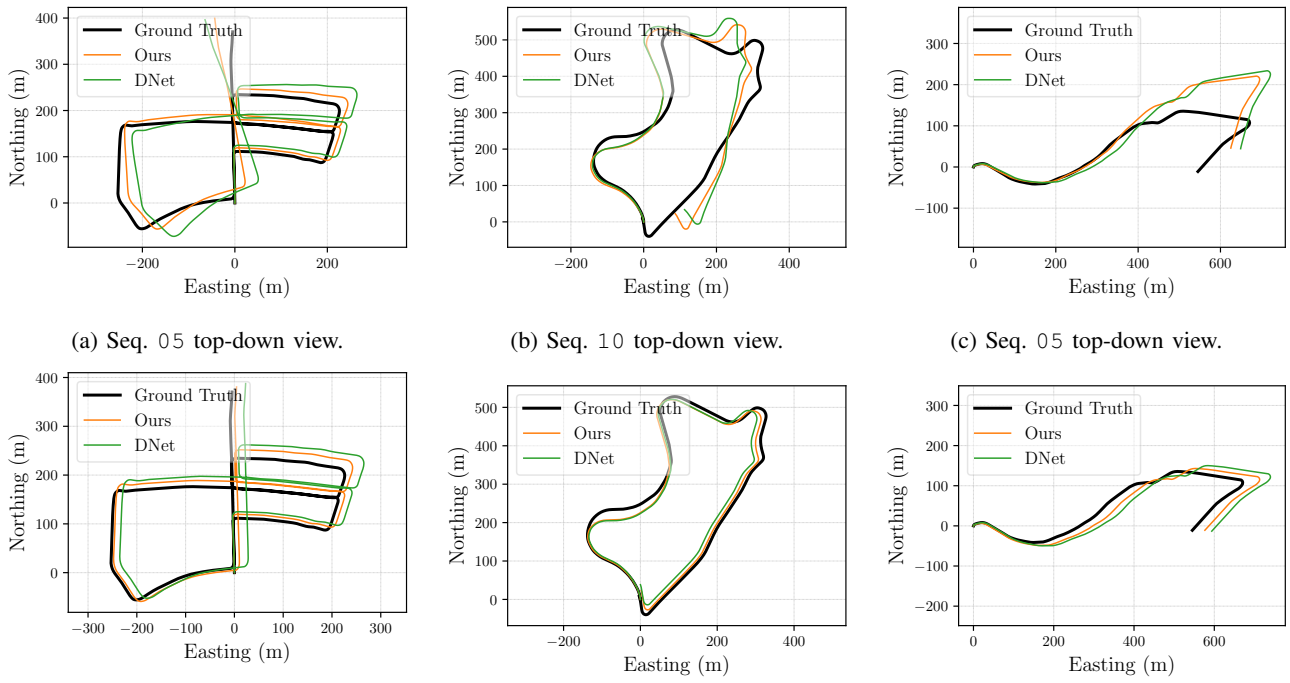
IV. Retraining Experiment

For Oxford RobotCar [10], [11] training, we kept all training conditions the same, with the exception of training for eight epochs only (due to there being a larger number of images in the training dataset), and adjusting the camera height to $h_{gt} = 1.52$ meters.

The RobotCar models were trained 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.

A. Online Rescaling Comparison

To demonstrate that our plane segmentation network can be used to accurately resolve scale, we compare our technique with two alternative methods: the DNet rescaling technique, and a RANSAC-based scale factor estimation method.



(a) Seq. 05 top-down view. (b) Seq. 10 top-down view. (c) Seq. 05 top-down view.

(d) Seq. 05 top-down view (GT Orientation). (e) Seq. 10 top-down view (GT Orientation). (f) Seq. 05 top-down view (GT Orientation).

Fig. 1: Visualization of the KITTI validation and test set trajectories and the corresponding scale factor estimates. Our scale recovery loss promotes scale-consistent depth and egomotion estimates that are metrically scaled (with a scale factor close to unity.

TABLE V: Benchmarking our method against other monocular VO estimators. Similar to [9], we include training sequences in our evaluation.

Method	Scaling Method	Mean Trans. Seg. Err. (%)								
		Train					Val.	Test		
		00	02	06	07	08	05	09	10	Mean
With Predicted Orientation										
Ours (no $\mathscr{L}_{TS} + \mathscr{L}_{DS}$)	DNet	3.71	4.32	4.69	2.36	4.21	4.70	7.23	13.98	5.65
Ours	None	**3.06**	**3.51**	**2.08**	**2.57**	**3.49**	**3.48**	**5.93**	**10.54**	**4.33**
With GT Orientation										
Ours (no $\mathscr{L}_{TS} + \mathscr{L}_{DS}$)	DNet	2.13	2.62	3.80	2.65	2.92	4.01	5.14	9.67	4.12
Ours	None	**1.83**	**2.22**	**1.60**	**1.38**	**2.37**	**2.51**	**3.63**	**6.14**	**2.71**

TABLE VI: Standard deviation of the scale factor (averaged across all frames within the specified sequence). A lower value indicates better scale consistency between independent predictions.

Method	Scale Factor Std. Dev.								
	Train					Val.	Test		
	00	02	06	07	08	05	09	10	Mean
Unscaled (no $\mathscr{L}_{TS} + \mathscr{L}_{DS}$)	0.1807	0.2624	0.1965	0.1874	0.5747	0.4517	0.0900	0.1218	0.2581
Scaled	**0.0389**	**0.0835**	**0.0423**	**0.0185**	**0.2400**	**0.1447**	**0.0279**	**0.0384**	**0.0793**

These methods all resolve scale factor by determining the camera height over a ground plane. The primary difference between methods is how each one identifies the pixels (and corresponding 3D coordinates) that belong to the ground plane. Our method uses a plane segmentation network to directly identify these pixels, while DNet extracts pixels whose surface normal is vertical, and our alternative approach uses a RANSAC-based approach which is described below.

1) RANSAC-based Scale Factor Estimation: We estimated the scale factor of the depth predictions by extracting the ground plane through a RANSAC-based [12] plane fitting procedure. First, for each image, we segmented the lower-middle region that generally corresponded to the road plane ($u \in [\frac{1}{6}W, \frac{5}{6}W]$, and $v \in [\frac{4}{7}H, H]$). Next, we determined the 3D coordinate for each ground plane pixel using

$$\mathbf{p}_t(u,v) = \hat{\mathbf{D}}_t(u,v) \begin{bmatrix} \frac{u-c_u}{f_u} & \frac{v-c_v}{f_v} & 1 \end{bmatrix}^T. \tag{2}$$

Then, for 350 iterations, we fit a plane to three sampled points and computed the number of inlier points (a point was considered an inlier if its offset was within 2% of the offset computed using the three sampled points). With the set of inlier points, we computed the offset (d) to the plane through $\mathbf{p}^T\tilde{\mathbf{n}} = d$, and considered the median offset to be the estimated camera height.

For our KITTI validation and test sequences, we use these three methods to evaluate the scale factor of each image in our validation and test sets, and report the per-sequence average scale factor. Table VII presents these results, in which the three scale estimation are consistent with each other. This indicates that our plane segmentation network can be used to accurately identify the ground plane.

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] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Sci. Robot.*, vol. 4, no. 30, 2019.

[4] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. Scandinavian Conf. Image Analysis*, 2003, pp. 363–370.

[5] 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.

[6] 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.

[7] 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," *arXiv preprint arXiv:2004.05560*, 2020.

[8] "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*.

[9] W. N. Greene and N. Roy, "Metrically-scaled monocular slam using learned scale factors."

[10] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *Int. J. Robot. Res. (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

[11] W. Maddern, G. Pascoe, M. Gadd, D. Barnes, B. Yeomans, and P. Newman, "Real-time kinematic ground truth for the oxford robotcar dataset," *arXiv preprint arXiv: 2002.10152*, 2020. [Online]. Available: https://arxiv.org/pdf/2002.10152

[12] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *J. Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

TABLE VII: Comparison of the accuracy for three online rescaling techiques.

Loss Type	Mean Scale Factor (Std. Dev.)		
	Seq. 05 (val.)	Seq. 09 (test)	Seq. 10 (test)
RANSAC-Based Plane Fitting	1.014 (0.022)	1.003 (0.027)	1.008 (0.034)
DNet	1.012 (0.018)	1.002 (0.023)	1.012 (0.034)
Plane Segmentation Network (Ours)	1.004 (0.042)	0.997 (0.028)	1.009 (0.038)