When the Sun Goes Down: Repairing Photometric Losses for All-Day Depth Estimation
SUPPLEMENTARY MATERIAL

Abstract: In this supplementary material, we provide (i) further details about how we trained and evaluated our networks on the Oxford RobotCar dataset; (ii) a comparison of the requirements of our method vs. those of existing state-of-the-art approaches; (iii) additional experiments and discussion relating to our estimation of ego-motion and lighting changes, and our method's ability to generalise to another dataset; and (iv) compute time analysis for our method.

1 Training and Evaluation Details

1.1 Network Architecture Details

DepthNet: Our DepthNet architecture is borrowed from our baseline, Monodepth2 [1]. ResNet-18, after removing the max-pooling and fully connected layers, is used as an encoder, and a sequence of up-convolution layers is used to decode depth information. Skip connections from the encoder are used to enrich the local context information in the estimated depth maps. We output single-channel depth at 4 different scales, having used sigmoid activation functions in the output layers.

MotionNet: Our MotionNet uses a ResNet-18 encoder that is similar to the above, but takes two images as input. There are three separate decoders attached to this encoder:

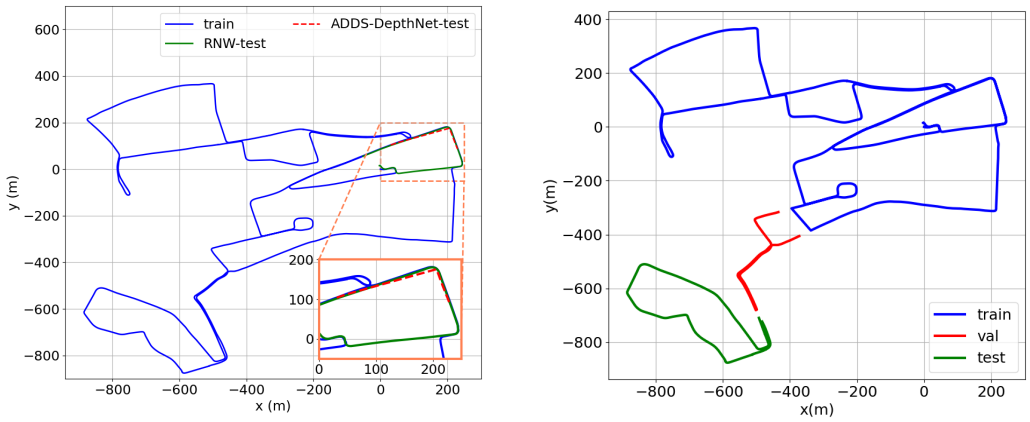
1. The MotionNet decoder inputs the features from the last convolutional layer and predicts the ego-motion of the camera. The MotionNet encoder and decoder are directly borrowed from Monodepth2 [1].
2. The lighting change decoder is similar to the DepthNet decoder, but without any skip connections. It outputs 2-channel images. We estimate light changes at 4 different scales. No activation functions are used in the output layers of this decoder.
3. The residual flow decoder is also similar to the DepthNet decoder, but it has skip connections from the MotionNet encoder and outputs 2-channel images at 4 different scales. Again, no activation functions are used in the output layers.

1.2 Dataset Processing

We evaluated our method's performance on both day and night sequences from the Oxford RobotCar dataset [2]. This dataset was collected over a one-year period by traversing the same route multiple times, so as to include a variety of different weather and lighting conditions. We used the six sequences in the `2014-12-09-13-21-02` traversal for our daytime experiments, and the six sequences in the `2014-12-16-18-44-24` traversal for our nighttime ones. We cropped the bottom 20% of each image to remove the car hood, and adjusted the camera intrinsics accordingly. We then filtered each sequence to produce a sub-sequence of keyframes such that each consecutive pair of keyframes was at least 0.5m apart. We then constructed our training, validation and testing splits from triplets of images in the original sequence, each centred on one keyframe (at this stage with a stride of 1). This process had the beneficial side-effect of filtering out most of the images that were captured when the ego-vehicle was stationary.

There are unfortunately no standard splits for the RobotCar dataset, and moreover the splits chosen by other methods [3, 4] overlap geographically (see Figure 1(a)). For these reasons, we constructed our own splits by geographically dividing our triplets (constructed as above) into three daytime sets

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.



(a) Splits of RNW [3], ADDS-DepthNet [4]

(b) Our splits

Figure 1: A geographic visualisation of the training and testing splits used by (a) RNW [3] and ADDS-DepthNet [4], and (b) our method, based on the GPS locations provided with the Oxford RobotCar dataset [2]. The splits of the methods in (a) clearly suffer from geographical overlap, whereas ours are explicitly constructed to be geographically disjoint, enabling fairer comparisons between methods on this dataset going forwards. (Best viewed in colour.)

τ'	*Method*	*Abs. Rel.*	*Sq. Rel.*	*RMSE*	*Log RMSE*	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
50m	RNW [3]	0.180	1.525	6.183	0.259	0.740	0.913	0.962
	Ours	0.171	1.481	6.009	0.242	0.758	0.917	0.965
100m	RNW [3]	0.185	1.710	6.549	0.262	0.733	0.910	0.960
	Ours	0.174	1.637	6.302	0.245	0.754	0.915	0.964

Table 1: The effects of truncating the depths estimated by different methods to different thresholds, for a maximum evaluation distance of 50m, evaluated over our test set. The methods' performance appears significantly better when the estimated depths are truncated to 50m rather than 100m.

and three nighttime sets (see Figure 1(b)). The nighttime splits are comprised of $19,612$ triplets for training, $2,629$ triplets for validation and $4,559$ triplets for testing. The daytime splits comprise $17,790$ triplets for training, $6,693$ triplets for testing and $2,629$ triplets for validation. We use the GPS locations provided with the dataset to ensure that there is no overlap between the training and testing locations in the city. Our hope in constructing these new splits is that they will help enable fairer comparisons between methods on this dataset going forwards.

1.3 Training

The networks were trained for 15 epochs using PyTorch. The learning rate was set to 10^{-4}. The loss function weights λ_r and λ_g (see main paper) were both set to 10^{-3}. We used the popular Adam optimiser with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to train our network. The residual sparsity regularisation weight λ_r was chosen by ablating three different weight values: 0.05, 0.005, and 0.001. We found 0.001 to work the best in terms of RMSE metrics as evaluated on the validation set. The gradient smoothening weight λ_g was directly borrowed from the baseline method Monodepth2 [1].

1.4 Evaluation

For depth evaluation, to avoid redundant computation due to the high camera frame rate, we sub-sampled the test set to have a 2m frame separation resulting in 709 test images for nighttime and 702 images for daytime. We used the (sparse) LiDAR data provided with the dataset as ground truth for

Metric	Definition
Absolute relative difference	$\frac{1}{\lvert\zeta_t\rvert}\sum_{\mathbf{u}\in\zeta_t}\frac{\lvert D_t(\mathbf{u})-D_t^\star(\mathbf{u})\rvert}{D_t^\star(\mathbf{u})}$
Squared relative difference	$\frac{1}{\lvert\zeta_t\rvert}\sum_{\mathbf{u}\in\zeta_t}\frac{(D_t(\mathbf{u})-D_t^\star(\mathbf{u}))^2}{D_t^\star(\mathbf{u})}$
RMSE	$\left(\frac{1}{\lvert\zeta_t\rvert}\sum_{\mathbf{u}\in\zeta_t}\lVert D_t(\mathbf{u})-D_t^\star(\mathbf{u})\rVert^2\right)^{0.5}$
Log RMSE	$\left(\frac{1}{\lvert\zeta_t\rvert}\sum_{\mathbf{u}\in\zeta_t}\lVert\log D_t(\mathbf{u})-\log D_t^\star(\mathbf{u})\rVert^2\right)^{0.5}$
Accuracy	$\frac{1}{\lvert\zeta_t\rvert}\lvert\{\mathbf{u}\in\zeta_t:\delta_t(\mathbf{u})<\eta\}\rvert$, in which $\delta_t(\mathbf{u})=\max\left(\frac{D_t^\star(\mathbf{u})}{D_t(\mathbf{u})},\frac{D_t(\mathbf{u})}{D_t^\star(\mathbf{u})}\right)$

Table 2: The error and accuracy metrics from [6] that are used for our depth evaluation in §4.1 of the main paper. In this, ζ_t denotes the set of pixels at frame t for which ground truth depth is available, D_t denotes the estimated depth image at frame t, and $D_t^\star$ denotes the corresponding ground-truth depth image. The RMSE metrics are reported in metres (m). For the accuracy, we consider three different thresholds η, namely 1.25, 1.25^2 and 1.25^3, in line with our baseline method [1]. Note that we report these as $\delta < \eta$ for the different thresholds η in the main results table.

the evaluation, as no official per-pixel ground truth depth is available. As per common practice, we used a script from the Oxford RobotCar SDK to reproject LiDAR points from several nearby frames onto the current test frame to increase the number of points available for evaluation.

Scale: Like other methods trained using only monocular images, our method can only estimate depth maps up to a scale factor. When reporting results, we scale the predictions of all the methods using the median value approach suggested in [5].

Metrics: We use the error and accuracy metrics from [6] to evaluate our depth estimation performance. The definitions of these metrics are shown in Table 2.

Depth Truncation: Typically, a depth estimation benchmark will specify one or more maximum distance thresholds, and then, for each such threshold τ, include only pixels whose ground-truth depth is $\leq \tau$ in the corresponding depth evaluation. This makes sense, since the intention is to determine the ability of a method to predict depths up to certain maximum distances.

However, for each τ, most (if not all) existing unsupervised depth estimation methods also truncate their *estimated* depth values to τ (as opposed to e.g. counting pixels whose depth is estimated as being $> \tau$ as outliers). Unfortunately, this makes far less sense, and can have the unintended side-effect of biasing the evaluation in their favour. For example, consider a method that estimates a depth $d \gg \tau$ for a pixel $\mathbf{u}$ whose corresponding ground-truth depth is τ. Without truncation, the error for this pixel will be $|d - \tau| \gg 0$, but with truncation, it will be $|\min(d, \tau) - \tau| = 0$. In other words, this practice can lead to low errors wrongly being ascribed to mispredicted pixels, which can cause the overall error that is reported to be much lower than it otherwise would have been.

To avoid this problem, and as an alternative to explicitly counting outliers, we truncate all estimated depth values in this paper to a threshold $\tau' \gg \tau$, thus limiting the maximum penalty for each outlier to at most $\tau' - \tau$ (in practice, we set $\tau' = 100$m, which is much larger than $\tau = 50$m, the largest maximum distance for which we report results). The difference this makes to the evaluation can be seen in Table 1.

2 Ease-of-Deployment Comparison with State-of-the-Art Approaches

To better understand the relative deployability of our method, we compare it to three state-of-the-art approaches, namely DeFeat-Net [7], ADDS-DepthNet [4] and RNW [3]. Besides outperforming the existing models, our method also has several other advantages over them, including easier data and training requirements, as shown in Table 3.

DeFeat-Net [7] is trained with an additional feature space with pixel-wise contrastive learning, which is assumed to be illumination and weather invariant. This feature network is simultaneously

Requirement	*DeFeat-Net* [7]	*ADDS-DepthNet* [4]	*RNW* [3]	*Ours*
No additional features	✗	✓	✓	✓
No daytime images	✓	✗	✗	✓
No paired day-night images	✓	✗	✓	✓
No GAN losses	✓	✓	✗	✓
Night-only training	✓	✗	✗	✓
Works for day and night	✓	✓	✗	✓

Table 3: Ease-of-deployment comparison of our method with three state-of-the-art approaches

Method	$t_{ate}\ (m) \downarrow$	$r_{ate}\ (rad) \downarrow$
Monodepth2 [1]	0.01317 ± 0.01517	$\mathbf{0.00036} \pm 0.00040$
DeFeat-Net [7]	0.03760 ± 0.01990	0.00115 ± 0.00129
ADDS-DepthNet [4]	0.01312 ± 0.01386	0.00083 ± 0.00061
Ours	$\mathbf{0.01310} \pm 0.01348$	0.00041 ± 0.00039

Table 4: A quantitative comparison of our ATE errors to those of Monodepth2 [1], DeFeat-Net [7] and ADDS-DepthNet [4] on our nighttime test sequence. (The results are in the form mean $\pm$ std.)

learned with the depth and pose networks. We used their pre-trained model to compare against our method.

ADDS-DepthNet [4] is compared against our method using the pre-trained models released alongside their code on GitHub. The orthogonality loss proposed in their method requires corresponding daytime data. It is not possible to train this model with night-only data.

RNW [3] did not release their pre-trained models alongside their code. We retrained their model with our training data and reported the results. This method uses a GAN-based regularisation loss, through which they achieve an $\approx 93\%$ improvement over the baseline method. To achieve this, they needed to carefully train a model using daytime images, as the nighttime performance directly depends on how well the daytime model is trained.

The results reported in their paper are trained with a ResNet-50 model, whereas the model we use is ResNet-18. When we tried retraining their approach with ResNet-18, we observed a divergence of training after a few epochs. Training instability is a well-known issue that must be overcome when working with GANs, but this does highlight the relative difficulties of making even simple modifications to a GAN-based system such as RNW in comparison to our own approach. Notably, in spite of such training difficulties, RNW is the only approach that is very close to our method in terms of its error and accuracy metrics. However, unlike our method, their model can only be used for nighttime depth estimation.

3 Ego-Motion Estimation

To evaluate the ability of our method to estimate the ego-motion of the camera, we compare the absolute trajectory error (ATE) [5, 8] it can achieve on our nighttime test sequence to the ATEs of three state-of-the-art methods, namely Monodepth2 [1], DeFeat-Net [7] and ADDS-DepthNet [4] (see Table 4). We also visualise the overall trajectory we estimate and compare it to those of other methods, as well as the ground truth (see Figure 2). Note that we estimate the ego-motion only up to scale, as our method uses monocular images during training. Moreover, the scale can drift over the course of the sequence, as there is no external constraint enforced to keep it constant.

We used the rescaling approach described in [5] for both our quantitative and qualitative pose estimation results. In both cases, our proposed method achieves results that are competitive with the state-of-the-art.

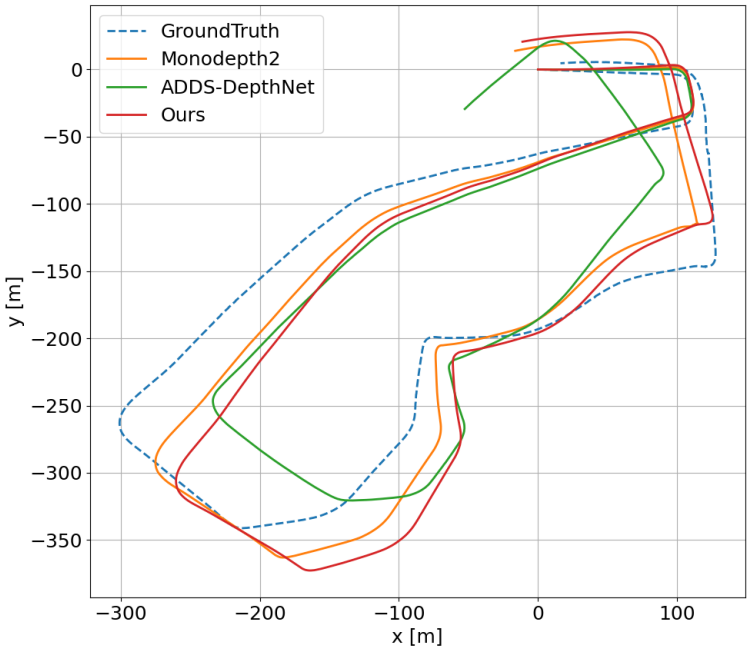


Figure 2: Comparing the trajectory estimated by our method for our nighttime test sequence to the trajectories estimated by Monodepth2 [1] and ADDS-DepthNet [4], and to the ground truth.

4 Lighting Change Compensation

In this section, we aim to provide a greater understanding of the lighting change compensation technique we propose in the main paper by motivating it from first principles.

Recall that at frame t, our approach first predicts two per-pixel change images, C_t and B_t, that aim to capture the per-pixel changes in contrast (scale) and brightness (shift) that occur between frame t and frame $t+1$. It then uses these change images to transform the reconstructed image I'_t via $\tilde{I}_t = C_t \odot I'_t + B_t$ before it is compared with image I_t.

A natural question to ask about this is: 'why is this linear transformation an appropriate model of the lighting changes between frames t and $t+1$?' In this section, we will attempt to provide an answer.

4.1 Theoretical Background

Image Formation: Recall the basic Phong illumination model from Computer Graphics, which calculates the brightness at a 3D point $\mathbf{p}$ in world space when viewed from a direction $-\hat{\mathbf{V}}(\mathbf{p})$:

$$B(\mathbf{p}) = B_a k_a + k_d \sum_{l \in lights} B_l \left((\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l(\mathbf{p})) + k_s (\hat{\mathbf{R}}_l(\mathbf{p}) \cdot \hat{\mathbf{V}}(\mathbf{p}))^{k_n} \right). \tag{1}$$

In this, as usual, B_a denotes the ambient illumination, B_l denotes the brightness of light l, $\{k_a, k_d, k_s\}$ denote the {ambient,diffuse,specular} reflection coefficients and k_n the specular exponent associated with the surface material, $\hat{\mathbf{N}}(\mathbf{p})$ denotes the unit normal to the surface at $\mathbf{p}$, $\hat{\mathbf{L}}_l(\mathbf{p})$ denotes a unit vector from $\mathbf{p}$ towards $\mathbf{p}_l$, the position of light l, $\hat{\mathbf{R}}_l(\mathbf{p})$ denotes the result of reflecting $\hat{\mathbf{L}}_l(\mathbf{p})$ across $\mathbf{N}(\mathbf{p})$, and $\hat{\mathbf{V}}(\mathbf{p})$ denotes a unit vector from $\mathbf{p}$ towards the viewer.

The image formation process can then be modelled as

$$I(\mathbf{u}) = f(e\mathcal{V}(\mathbf{u})B(\mathbf{p})), \tag{2}$$

in which $\mathbf{u}$ is the pixel corresponding to $\mathbf{p}$ on the image plane, f denotes the camera response function, e denotes the exposure time, and $\mathcal{V}$ denotes the Vignetting function. This is commonly simplified (e.g. [9, 10]) by assuming that f is the identity function and that there is no Vignetting effect, in which case the model simplifies to

$$I(\mathbf{u}) = eB(\mathbf{p}). \tag{3}$$

Further Simplification: For our purposes here, we will simplify this model still further by assuming that the only illumination in the scene is diffuse, which allows us to approximate $B(\mathbf{p})$ as

$$B(\mathbf{p}) \approx k_d \sum_{l \in lights} B_l(\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l(\mathbf{p})). \tag{4}$$

Whilst this assumption is not really true in practice (though the *ambient* illumination at night is likely to be negligible), it will significantly simplify our derivation in what follows.

Lighting Change Model: To model the change in lighting between frames t and $t+1$ for a 3D world-space point $\mathbf{p}$ that is imaged at pixels $\mathbf{u}$ in I_t and $\mathbf{u}'$ in I_{t+1}, we can first approximate $I_t(\mathbf{u})$ and $I_{t+1}(\mathbf{u}')$ using our image formation model:

$$\begin{aligned} I_t(\mathbf{u}) &\approx e^{(t)}\ k_d \sum_{l \in lights} B_l^{(t)}(\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l^{(t)}(\mathbf{p})) \\ I_{t+1}(\mathbf{u}') &\approx e^{(t+1)}\ k_d \sum_{l \in lights} B_l^{(t+1)}(\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l^{(t+1)}(\mathbf{p})). \end{aligned} \tag{5}$$

In this, $e^{(t)}$ denotes the exposure time at frame t, $B_l^{(t)}$ denotes the brightness of light l at frame t and $\hat{\mathbf{L}}_l^{(t)}(\mathbf{p})$ denotes a unit vector from $\mathbf{p}$ towards $\mathbf{p}_l^{(t)}$, the position of light l at frame t. Note that both the brightness and the position of each light in the scene are now assumed to potentially vary with time (e.g. a car headlight will move with the car).

We next observe that both I_t and I_{t+1} are actually 3-channel RGB images, not single-channel intensity images as in the simplest version of the Phong illumination model. In Computer Graphics, this is commonly handled by assigning a 3-channel RGB colour to each light, thus making each $B_l^{(t)}$ a 3-channel vector. However, it is more physically correct to think of the diffuse reflection coefficient k_d as a 3-channel vector that captures the extent to which different components of white light get reflected/absorbed by the surface material. Moreover, it will turn out to be possible to eliminate k_d in this case, and so for our purposes here, we will take this latter view.

To eliminate k_d, it suffices to divide $I_t(\mathbf{u})$ by $I_{t+1}(\mathbf{u}')$ via

$$\frac{I_t(\mathbf{u})}{I_{t+1}(\mathbf{u}')} \approx \frac{e^{(t)} \sum_{l \in lights} B_l^{(t)}(\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l^{(t)}(\mathbf{p}))}{e^{(t+1)} \sum_{l \in lights} B_l^{(t+1)}(\hat{\mathbf{N}}(\mathbf{p}) \cdot \hat{\mathbf{L}}_l^{(t+1)}(\mathbf{p}))} \mathbb{1} = C_t^\star(\mathbf{u})\mathbb{1}, \tag{6}$$

in which $\mathbb{1} = (1,1,1)^\top \in \mathbb{R}^3$. This yields a scalar, $C_t^\star(\mathbf{u})$, that can clearly be multiplied by $I_{t+1}(\mathbf{u}')$ to approximate $I_t(\mathbf{u})$ via $C_t^\star(\mathbf{u}) I_{t+1}(\mathbf{u}') \approx I_t(\mathbf{u})$. Interestingly, since $I_t'(\mathbf{u}) \approx I_{t+1}(\mathbf{u}')$ as per the definition in the main paper, this means that $C_t^\star(\mathbf{u}) I_t'(\mathbf{u}) \approx I_t(\mathbf{u})$, which we can also write as

$$I_t \approx C_t^\star \odot I_t'. \tag{7}$$

This corresponds to a single-channel lighting change model in which we will aim to compensate for the lighting changes in I_t' by simply multiplying the value of each pixel in I_t' by its corresponding scaling factor.

4.2 Scale-only Model

Based on the theoretical formulation in the previous sub-section, it makes sense to try to predict a single per-pixel change image C_t for each frame t (using a single-channel variant of the lighting change decoder described in the main paper) that can take the place of $C_t^\star$ in Equation 7 and be used to multiply I_t' to compensate for lighting changes. The effects of applying such a scale-only

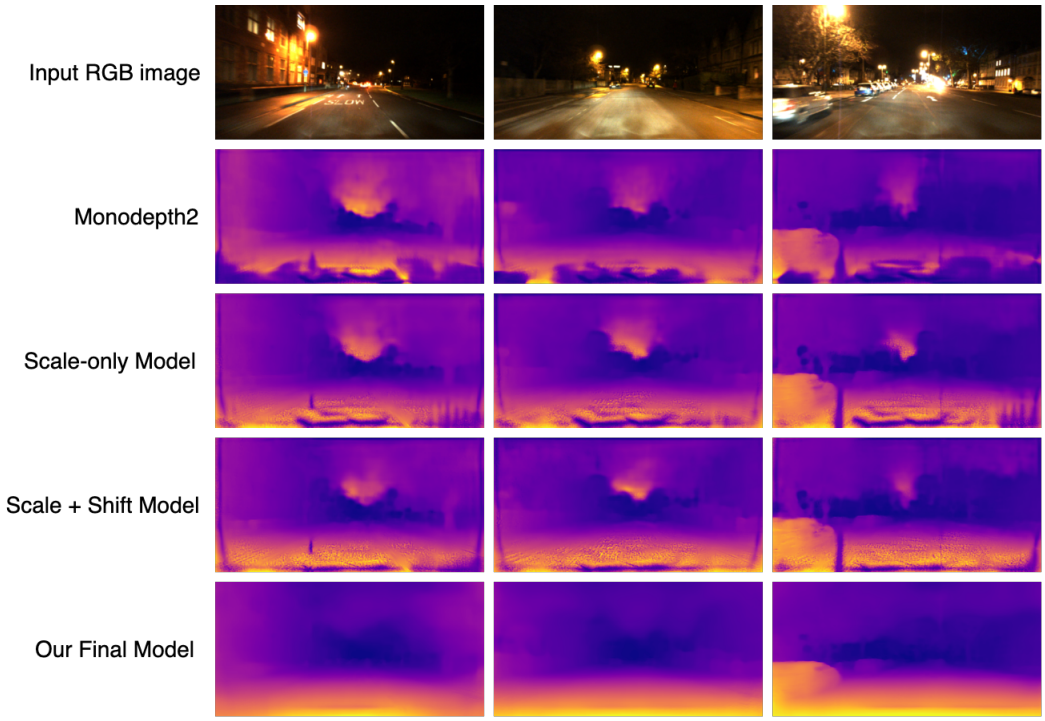


Figure 3: A qualitative comparison showing the differences in depth estimation performance between Monodepth2 [1], our scale-only lighting change model (see §4.2) and our scale + shift lighting change model (see §4.3). Our final model (which additionally includes the motion compensation and denoising components) achieves the best overall performance.

lighting change compensation technique to Monodepth2 [1] can be seen in Figure 3, in which it can be clearly seen that e.g. the sizes of the holes directly in front of the car have been significantly reduced. Noticeably, however, our scale-only model still struggles to cope with the strong intensity discontinuities caused by the car headlights in the close-range image region.

4.3 Scale + Shift Model

The intensity transforms are purely linear when using the scale-only model, but they can be straightforwardly changed into affine ones by introducing an additional (shift) channel B_t. This provides an extra degree of freedom to the network to learn a suitable intensity transform for each pixel, and, as shown in Figure 3, results in significant improvements in the estimated depth for the affected areas. We observed that our affine (scale+shift) model specifically improved depth estimation for pixels close to the camera that represent textureless road patches illuminated non-uniformly by the ego-vehicle's headlights. We hypothesise that given the specific nature of this image region, the affine model benefits from additional learnable parameters.

5 Generalisation to Another Dataset

To examine our method's ability to generalise to datasets other than Oxford RobotCar [2], we performed a qualitative evaluation of our model on a number of images from the Once dataset [11]. The images in question were extracted from sequence `000340` of the *Unlabeled medium split* subset of the dataset. It is important to note here that the lighting conditions and geographic location of the Once dataset are very different from those of Oxford RobotCar. Nevertheless, despite these differences, our model manages to estimate visually plausible depth maps with sharp structural details for these images, as shown in Figure 4.

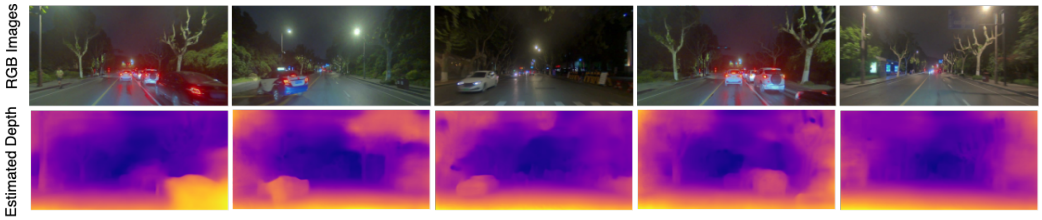


Figure 4: Qualitative examples showing the performance of our model trained on the Oxford RobotCar dataset [2] when tested on images from the Once dataset [11].

6 Compute Time Analysis

Our inference network (DepthNet) is composed of a ResNet-18 encoder and a U-Net-style decoder. It has 14.84 million parameters, and its computational complexity, as measured in GMACs (Giga Multiply Accumulate per second), is 8.58. For a batch size of 1, the model runs at 231 FPS (i.e. each frame takes roughly 0.00423s) on an NVIDIA A10 GPU, and at 34 FPS (i.e. each frame takes roughly 0.02887s) on an Intel Xeon Silver 4314 CPU @ 2.40GHz. These numbers were estimated by averaging over 300 runs. We used the `ptflops`[1] library to estimate the GMACs.

References

[1] C. Godard, O. Mac Aodha, M. Firman, and G. Brostow. Digging Into Self-Supervised Monocular Depth Estimation. In *ICCV*, pages 3828–3838, 2019.

[2] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1):3–15, 2017.

[3] K. Wang*, Z. Zhang*, Z. Yan, X. Li, B. Xu, J. Li, and J. Yang. Regularizing Nighttime Weirdness: Efficient Self-supervised Monocular Depth Estimation in the Dark. In *ICCV*, pages 16055–16064, 2021.

[4] L. Liu, X. Song, M. Wang, Y. Liu, and L. Zhang. Self-supervised Monocular Depth Estimation for All Day Images using Domain Separation. In *ICCV*, pages 12737–12746, 2021.

[5] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*, pages 1851–1860, 2017.

[6] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *NeurIPS*, pages 2366–2374, 2014.

[7] J. Spencer, R. Bowden, and S. Hadfield. DeFeat-Net: General Monocular Depth via Simultaneous Unsupervised Representation Learning. In *CVPR*, pages 14402–14413, 2020.

[8] M. Vankadari, S. Kumar, A. Majumder, and K. Das. Unsupervised Learning of Monocular Depth and Ego-Motion using Conditional PatchGANs. In *IJCAI*, pages 5677–5684, 2019.

[9] H. Jin, P. Favaro, and S. Soatto. Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination. In *ICCV*, pages 684–689, 2001.

[10] N. Yang, L. von Stumberg, R. Wang, and D. Cremers. D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry. In *CVPR*, pages 1281–1292, 2020.

[11] J. Mao*, M. Niu*, C. Jiang, H. Liang, J. Chen, X. Liang, Y. Li, C. Ye, W. Zhang, Z. Li, J. Yu, H. Xu, and C. Xu. One Million Scenes for Autonomous Driving: ONCE Dataset. *arXiv preprint arXiv:2106.11037v3*, 2021.

[1] `https://github.com/sovrasov/flops-counter.pytorch`