A. Appendix

A.1. Computing the surface normal of the road

In this section, we provide a detailed explanation of how the average normal vector,

$$\mathbf{N}_{\text{avg}}(i,j)$$

, is computed in our method. This process involves defining the central point and its neighbors, computing vectors to the neighbors, computing cross products of these vectors, and finally normalizing and averaging the results.

$$\mathbf{N}_{\text{avg}}(i,j) = \frac{1}{4}\sum_{k=1}^{4}\frac{\mathbf{V}_{k1}(i,j)\times\mathbf{V}_{k2}(i,j)}{\|\mathbf{V}_{k1}(i,j)\times\mathbf{V}_{k2}(i,j)\|} \tag{20}$$

where, V_{k1} and V_{k2} are the vectors from the central point to its neighbours, as shown in Eq. (21) and Eq. (22).

$$\mathbf{V}_{k1}(i,j) = \mathbf{P}(i+\Delta i_k, j+\Delta j_k) - \mathbf{P}(i,j) \tag{21}$$

$$\mathbf{V}_{k2}(i,j) = \mathbf{P}(i+\Delta i'_k, j+\Delta j'_k) - \mathbf{P}(i,j) \tag{22}$$

with $(\Delta i_k, \Delta j_k)$ and $(\Delta i'_k, \Delta j'_k)$ representing the offsets for the neighboring points. Then, the final surface normal for each pixel is the mean of these normals as shown in Eq. (20). For example, the computed vectors when using the nearest neighbors n are:

$$\begin{aligned}
\mathbf{V}_{11}(i,j) &= \mathbf{P}(i,j-n) - \mathbf{P}(i,j)\\
\mathbf{V}_{12}(i,j) &= \mathbf{P}(i-n,j) - \mathbf{P}(i,j)\\
\mathbf{V}_{21}(i,j) &= \mathbf{P}(i,j+n) - \mathbf{P}(i,j)\\
\mathbf{V}_{22}(i,j) &= \mathbf{P}(i+n,j) - \mathbf{P}(i,j)\\
\mathbf{V}_{31}(i,j) &= \mathbf{P}(i-n,j-n) - \mathbf{P}(i,j)\\
\mathbf{V}_{32}(i,j) &= \mathbf{P}(i+n,j-n) - \mathbf{P}(i,j)\\
\mathbf{V}_{41}(i,j) &= \mathbf{P}(i-n,j+n) - \mathbf{P}(i,j)\\
\mathbf{V}_{42}(i,j) &= \mathbf{P}(i+n,j+n) - \mathbf{P}(i,j)
\end{aligned}$$

Normalize and Average We normalize each of the computed normal vectors and then take their average. This is done using Eq. (20). Then, this is the surface normal at this pixel. These normal vectors are filtered based on their alignment with the surface normal of the road, as described in our method.

A.2. Auto-masking for re-projection loss

In the context of self-supervised monocular depth estimation, auto-masking plays a crucial role in handling occlusions. The auto-masking mechanism is integrated into the minimum reprojection loss, which is defined as:

$$L_{\text{reproj}} = \min_s \text{pe}(\hat{I}_s, I_t) \tag{23}$$

where I_s and I_t represent the source and target images respectively. The minimum operation in the loss function ensures that for each pixel in the target image, the model considers the best possible projection from the source images. This mechanism effectively serves as an automatic mask, enabling the model to be robust against occlusions. The pixels corresponding to occluded regions in the source image would have a high reprojection error, and hence, are automatically down-weighted in the loss computation. This auto-masking mechanism retains only the loss of pixels where the reprojection error of the warped image $\hat{I}_s$ is lower than that of the original, unwarped source image I_s. This can be mathematically represented as:

$$M_{\text{auto}} = [\min_s \text{pe}(I_t, \hat{I}_s) < \min_s \text{pe}(I_t, I_s)] \tag{24}$$

where pe denotes the photometric error, I_t is the target image, $\hat{I}_s$ is the image warped from s to t, and I_s is the original, unwarped source image. The function M_{auto} serves as a mask that includes only the pixels where the reprojection error of the warped image is lower than that of the original image, [ ] denotes Iverson bracket.

where L_{identity} is the identity reprojection loss, L_{reproj} is the reprojection loss. The mask M takes the value 1 for pixels where the identity reprojection loss is less than the reprojection loss, and 0 otherwise. This effectively down-weights the contribution of occluded pixels in the loss computation, thereby making the model robust to occlusions. Also this mask M_{auto} was utilized in the same fashion for masking out any invalid depth calculated by the teacher θ_{pp}.

A.3. Smoothness loss

Equation (25) is widely used in depth estimation models, which are often trainable methods. This equation encourages the disparity map to be smooth in regions where the image content is smooth, thereby reducing noise and improving the overall quality.

$$L_{smooth} = |\partial_x d^* t| \cdot e^{-|\partial_x I_t|} + |\partial_y d^* t| \cdot e^{-|\partial_y I_t|} \tag{25}$$

Where d^* is the mean-normalized disparity. The exponential term makes this a robust function, meaning it is less sensitive to large disparity changes in the presence of strong image gradients, which may be gradients due to brightness changes, or any other external factors. This is important because edges in an image often correspond to depth discontinuities in the scene, so it is desirable for the disparity map to have sharp changes at these locations. Therefore, the smoothness loss helps to preserve edge information while ensuring overall smoothness, leading to more accurate and visually pleasing disparity maps.

A.4. Scale computation for base-line comparison

In this section, we present the methodology employed to recover the scale from the depth map using surface normal vectors, which can be used for any scale-invariant model. Two different methods are used to compute the scale, both relying on the predicted height of the camera. Both methods rely on predicting the surface normal vectors from the depth map and then using those predictions to estimate the road plane and camera height.

A.4.1 Method 1: Road Plane Estimation Using RANSAC

The first method involves estimating the road plane by leveraging the surface normal vectors predicted from the depth map.

1. **Depth Map and Surface Normals:** We start with the depth map of the scene, and from this, we predict the surface normal vectors for all the pixels in the image.

2. **Identifying Flat Areas:** The flat areas in the scene (same as M_{flat}) are identified based on the surface normals, which are pointing almost in the same direction as the road surface normal.

3. **Road Plane Estimation:** Using the predicted surface normals for the flat areas, we employ RANSAC to compute an estimate of the road plane. Although this robust estimation technique helps to exclude outliers, in some cases where there are a lot of non-road flat areas, it produces wrong results.

4. **Camera Height Estimation:** Once the road plane is estimated, the inferred height of the camera can be computed from its distance to the road plane.

5. **Scale Adjustment:** The scale is then computed as the ratio between the predicted camera height and the actual known camera height.

A.4.2 Method 2: Median Height Estimation from All Pixels

The second method is computationally more straightforward, as it avoids the RANSAC optimization process. Instead, it calculates the camera height by taking the median of the estimated heights from the flat pixels, providing a direct and efficient solution.

1. **Depth Map and Surface Normals:** Similar to the first method, we start with the depth map and predict the surface normal vectors for all the pixels.

2. **Height Calculation for the flat area:** We compute the inferred height of the camera using all flat-area pixels in the image.

3. **Median Height Estimation:** Once the heights are computed for these pixels, we take the median of these inferred heights. The median serves as a robust estimate of the camera height, mitigating the influence of outliers.

4. **Scale Adjustment:** Similar to the first method, we compute the scale as the ratio between the predicted median camera height and the actual known camera height.

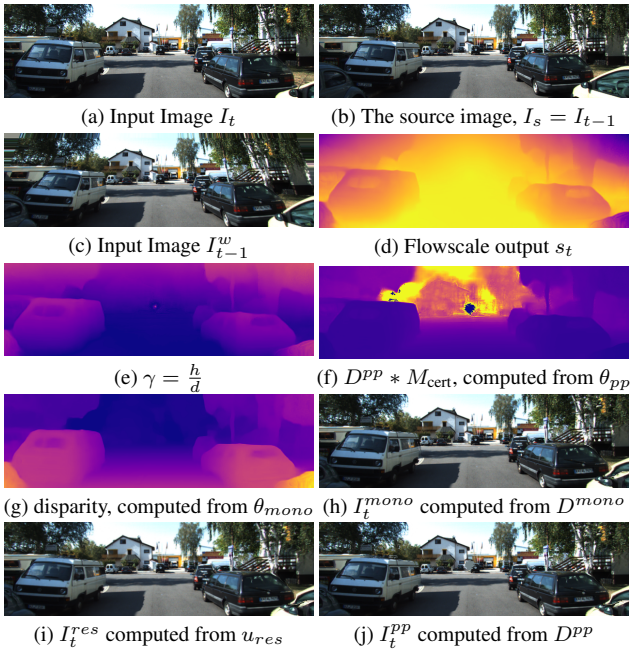
Both methods offer reliable approaches for recovering the scale based on the camera height derived from surface normals and depth data. Although these methods were not extensively tested, our results indicate that the second method is both simpler and yields more accurate scale estimates. For this reason, we adopted it in establishing our baseline using depth maps generated by Monodepth2.

A.5. Zero-shot testing

In the zero-shot testing scenario, we evaluated the model trained on KITTI using Cityscapes data, which the model had not seen during training. A key challenge in this process was the difference in scale between the datasets. To ensure a fair comparison, we implemented a straightforward module to adjust the scale by estimating the camera's height relative to the road. This adjustment was done using the same method described in the scale recovery process, ensuring consistent and accurate depth estimation across both datasets.

A.6. Qualitative results

In Fig. 6, we illustrate the entire process of the model, encompassing both the teacher and student phases, during training. This serves as a practical demonstration of how each step is executed within the model. In Figs. 8 to 10, we present successful cases where the output of M_{static} effectively maps out dynamic objects. Conversely, Figs. 11 to 13 highlight failure cases, where our masking strategy does not perform as intended, some of these failures are in masking the dynamic objects by M_{static}, as in Fig. 12, leading to completely incorrect depth as it calculates the depth based on the disparity of a moving objects. On the other hand, there are successful cases, such as in Fig. 9, where the dynamic object is not entirely masked, but only its boundaries. Despite this, the disparity output for the dynamic object is still correct.



(a) Input Image I_t (b) The source image, $I_s = I_{t-1}$

(c) Input Image I^w_{t-1} (d) Flowscale output s_t

(e) $\gamma = \frac{h}{d}$ (f) $D^{pp} * M_{\text{cert}}$, computed from θ_{pp}

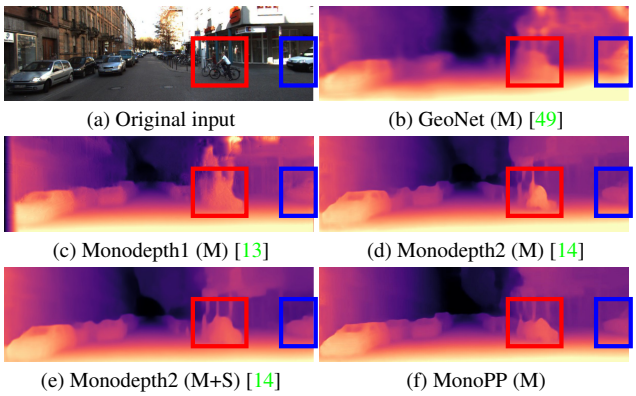
(g) disparity, computed from θ_{mono} (h) I^{mono}_t computed from D^{mono}

(i) I^{res}_t computed from u_{res} (j) I^{pp}_t computed from D^{pp}

Figure 6. Example for all the outputs as well as the intermediate outputs needed for computing the losses

As shown in Fig. 6, the training steps involved in our pipeline are outlined as follows. Starting with an input and source image, the teacher model computes the flow scale (epipolar flow scaling), followed by the calculation of the gamma parameter. The predicted depth is then masked by M_{cert}, ensuring that only reliable depth estimates are retained. Finally, novel views are synthesized using the outputs of the model.

A.6.1 Good cases



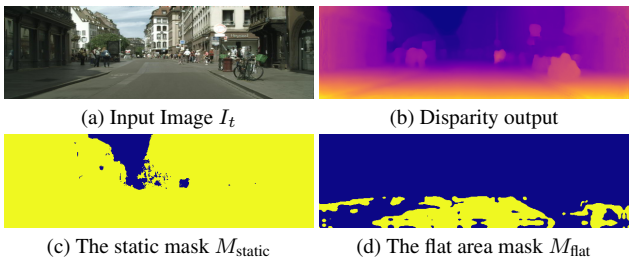
(a) Original input (b) GeoNet (M) [49]

(c) Monodepth1 (M) [13] (d) Monodepth2 (M) [14]

(e) Monodepth2 (M+S) [14] (f) MonoPP (M)

Figure 7. Qualitative results on KITTI [12], on eigen split [9] in comparison with other SOTA methods. The finer details of the bike and the vehicle are detected.

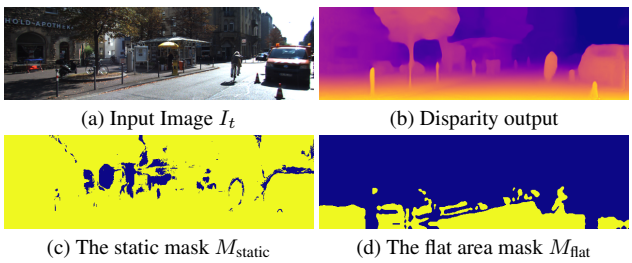
In Fig. 7, we compare our model to recent approaches that also use single-frame monocular depth estimation. Although our model predicts metric-scaled depth, it achieves qualitatively comparable results to Monodepth2 (M+S), which was trained using stereo image pairs. This demonstrates that our approach performs competitively, despite relying solely on monocular input during training.



(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

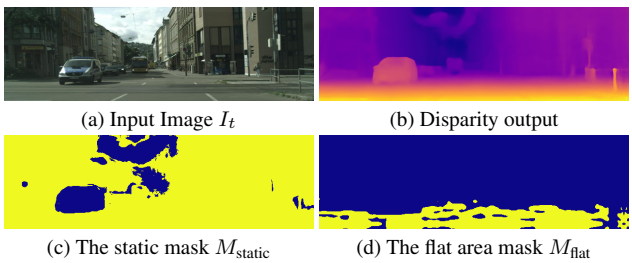
Figure 8. A qualitative example from Cityscapes, which shows that M_{static} will not affect the fully-static scene, the only masked area is the textureless sky, which is often mistaken for dynamic objects



(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

Figure 9. Qualitative results on KITTI [12], the final depth result is correct. However, it is a failure case, where M_{static} classifies some static objects as dynamic and vice versa. This is happening sometimes due to the rotational movement of the vehicle, hence some objects are wrongly classified as dynamic objects.

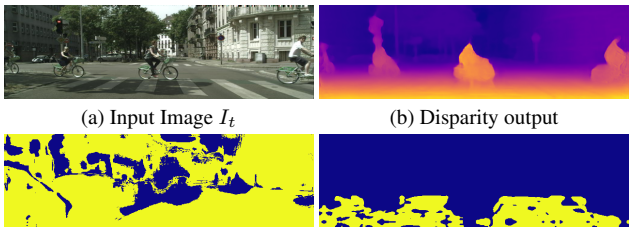


(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

Figure 10. Qualitative results on Cityscapes example, which was mentioned in the paper, and this is a good example of the usability of M_{static}.

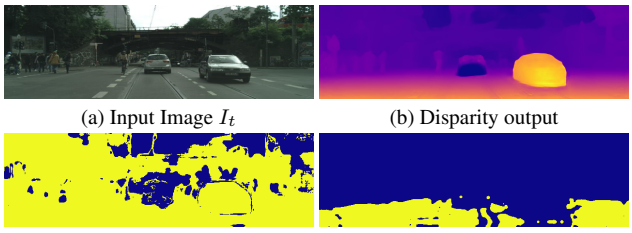
A.6.2 Failure cases



(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

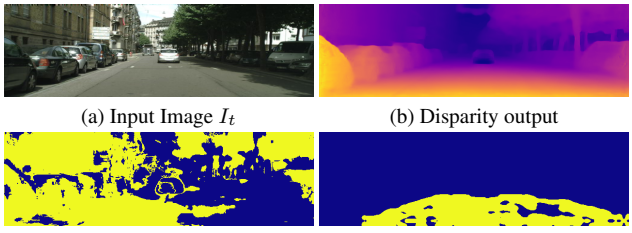
Figure 11. Qualitative results on Cityscapes, and this is one of the failure cases that M_{static} filters out this dynamic object. However, it still was perceived as a bigger object, which is due to its closeness to the camera and its speed.



(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

Figure 12. This is one of the challenging examples on cityscapes, which shows that of course our masking strategy does not filter out all dynamic objects. Hence, this will lead to hallucinated depth, which negatively affects our losses.



(a) Input Image I_t (b) Disparity output

(c) The static mask M_{static} (d) The flat area mask M_{flat}

Figure 13. This is one of the classic failures in Cityscapes, which is a moving car in front of the ego-vehicle, moving at similar speed and located around the epipole of the camera movement.

A.6.3 Rendered 3D point clouds results

All the presented figures, Figs. 14 to 18, are formatted such as the first top image is the input to the inference network for depth prediction, and then the 3D point clouds are rendered from this single image only. All the examples are samples from the evaluation Eigen-split benchmark of KITTI dataset, which means that the network was not trained on these samples.

Figure 14. Rendered a 3D point cloud for KITTI data using MonoPP, based solely on a single 2D image input. Multiple view angles were used to visualize the scene.

A.7. Quantitative results for KITTI

Tab. 4 provides a comprehensive overview of the state-of-the-art (SOTA) methods in the field of self-supervised monocular depth estimation (with and without GT median scaling). It delineates the key differences between single-frame and multi-frame methods, providing valuable insights into their respective strengths and limitations. The table serves as a useful resource for future researchers, as it underscores the general superiority of multi-frame methods in terms of performance. However, it also highlights an important caveat: in scenarios where there is no baseline available, *i.e.* only a single frame is available, single-frame methods may offer better results. This analysis can guide future research in this domain, informing the choice of methods based on the specific constraints.

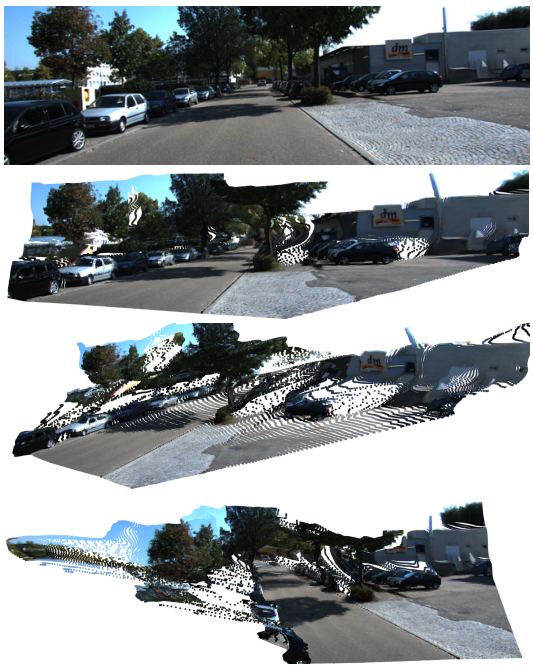


Figure 15. An additional example shows interesting faraway reconstruction of the rendered point clouds from a single image using MonoPP

Figure 16. This example shows a good quality of rendered scene from a different view angle

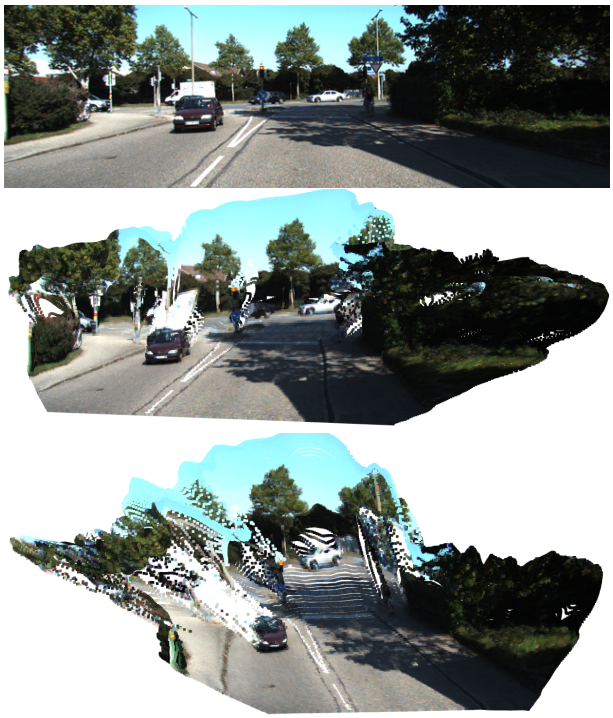


Figure 17. This is a special sample which contains a lot of moving dynamic objects, which are more prone to error. However, the rendered scenes are of good quality

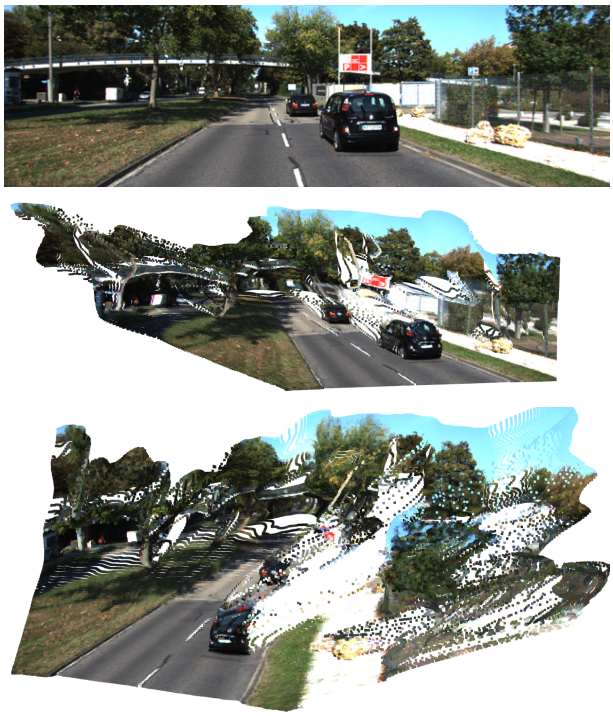


Figure 18. This is a special sample which contains a lot of moving dynamic objects, which are more prone to error. However, the rendered scenes are of good quality

	Year	Method	Test frames	Train	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
scaled by GT	2017	Monodepth1 [13]	1	M	0.148	1.344	5.927	0.247	0.803	0.922	0.964
	2018	GeoNet [49]	N	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
	2019	Monodepth2 [14]	1	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
				M+S	0.106	0.818	4.750	0.196	0.874	0.957	0.979
	2020	Patil *et al*. [29]	N	M	0.11	0.82	4.65	0.187	0.883	0.961	0.982
	2020	PackNet-SFM [16]	1	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
	2020	DNet [45]	1	M	0.113	0.864	4.812	0.191	0.877	0.960	0.981
	2021	ManyDepth [40]	N	M	0.098	0.770	4.459	0.176	0.90	0.965	0.983
			1	M	0.106	0.818	4.750	0.196	0.874	0.957	0.979
	2021	CADepth [46]	1	M	0.110	0.812	4.686	0.187	0.882	0.961	0.981
	2021	Sui *et al*. [35]	1	M	0.111	0.894	4.779	0.189	0.883	0.960	0.981
	2022	VADepth [42]	1	M	0.104	0.774	4.552	0.181	0.892	0.965	0.983
	2022	MonoFormer [3]	1	M	0.108	0.806	4.594	0.184	0.884	0.963	0.983
	2022	DepthFormer [17]	N	M	**0.090**	**0.661**	**4.149**	0.175	0.905	**0.967**	**0.984**
	2022	MonoViT [54]	1	M	0.099	0.708	4.372	0.175	0.900	**0.967**	**0.984**
	2023	Lite-Mono [51]	1	M	0.107	0.765	4.561	0.183	0.886	0.963	0.983
	2023	Lite-Mono-S [51]	1	M	0.110	0.802	4.671	0.186	0.879	0.961	0.982
	2023	TriDepth [7]	1	M	0.093	0.665	4.272	**0.172**	**0.907**	**0.967**	**0.984**
	MonoPP (ours)		1	M	0.105	0.776	4.640	0.185	0.891	0.962	0.982
w/o scaling	2019	Monodepth2** [14]	1	camH	0.126	0.973	4.880	0.198	0.864	0.957	0.980
	2020	DNet [45]	1	M+camH	0.118	0.925	4.918	0.199	0.862	0.953	0.979
	2020	Zhao *et al*. [53]	1	M+SC	0.146	1.084	5.445	0.221	0.807	0.936	0.976
	2020	PackNet [16]	1	M+V	0.111	0.829	4.788	0.199	0.864	0.954	0.980
	2021	Wagstaff *et al*. [37]	1	M+Pose	0.123	0.996	5.253	0.213	0.840	0.947	0.978
	2021	Wagstaff *et al*. [37]	1	M+camH	0.155	1.657	5.615	0.236	0.809	0.924	0.959
	2021	Sui *et al*. [35]	1	M+camH	0.128	0.936	5.063	0.214	0.847	0.951	0.978
	2022	VADepth [42]	1	M+camH	0.109	0.785	4.624	0.190	0.875	0.960	**0.982**
	2022	DynaDepth [52]	1	M+Pose	0.108	**0.761**	**4.608**	0.187	0.883	**0.962**	**0.982**
	2023	Lee *et al*. [24]	1	M+Pose	0.141	1.117	5.435	0.223	0.804	0.942	0.977
	2024†	Kinoshita & Nishino [21]	1	M+SI	0.108	0.785	4.736	0.195	0.871	0.958	0.981
	MonoPP (ours)		1	M+camH	**0.107**	0.835	4.658	**0.186**	**0.891**	**0.962**	**0.982**

Table 4. Comparison of our method to existing self-supervised approaches on the KITTI [12] Eigen split [9]. There are two separated tables, the upper one is dedicated for the comparison of scale-invariant depth, which means the predicted depth still needs to be scaled, hence all methods still need to calculate the scale from the ground-truth. The lower table focuses on comparing against the methods that predicts scaled depth. The best results in each subsection are in **bold** second best are underlined. As shown, Our method outperforms other methods in predicting scaled metric depth estimation. All comparison is done for the medium resolution (640 x 192). **M** stands for training by monocular videos, and **S** includes stereo data as well. **SC*** stands for predicting a scale consistent output, which may still need GT for scaling. **Pose** for utilizing the pose information, **V** for utilizing the vehicle's velocity, and **camH** for utilizing initial camera height from the ground, and **SI** for scraping large-dataset from the internet while training.↑ higher values are better. ↓ lower values are better. † This is an arxiv pre-print which first published in 2023, but these are their new results reported in 2024. ** is a baseline that we implemented to predict post-processed metric-scaled depth from Monodepth2, scaled by the GT camera height, as illustrated in Appendix A.4