Mining Supervision for Dynamic Regions in Self-Supervised Monocular Depth Estimation
——Supplementary Material——

Hoang Chuong Nguyen[1] Tianyu Wang[1] Jose M. Alvarez[2] Miaomiao Liu[1]
[1]Australian National University [2]NVIDIA
{hoangchuong.nguyen, tianyu.wang2, miaomiao.liu}@anu.edu.au josea@nvidia.com

1. More Implementation Details

1.1. Loss Functions

Here, we provide detailed formulation of the losses we used in the paper. In particular, we follow the definition of the depth smoothness loss L_s in [3], the object-motion-sparsity loss L_g in [13] and the photometric loss L_p in [4].

Depth smoothness loss. Given the depth map $\mathbf{D}_r^{scene}$ predicted by the depth network Θ^{scene} and the input image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, the edge-aware smoothness loss [3] is computed as,

$$\begin{aligned} L_s(\mathbf{D}_r^{scene}, \mathbf{I}) &= \frac{1}{HW}\sum_{u,v}(|\partial_u \mathbf{D}_{r,\star}^{scene}(u,v)|\mathrm{e}^{-|\partial_u \mathbf{I}(u,v)|}) \\ &+ \frac{1}{HW}\sum_{u,v}(|\partial_v \mathbf{D}_{r,\star}^{scene}(u,v)|\mathrm{e}^{-|\partial_v \mathbf{I}(u,v)|}), \end{aligned}$$

$$\mathbf{D}_{r,\star}^{scene}(u,v) = \frac{(\mathbf{D}_r^{scene}(u,v))^{-1}HW}{\sum_{u',v'}(\mathbf{D}_r^{scene}(u',v'))^{-1}}. \quad (1)$$

Regarding the object depth $\mathbf{D}_r^o$ produced by Θ^{obj}, we enforce depth smoothness within the object mask $\mathbf{M} \in \{0,1\}^{H\times W\times 1}$ regardless of the object's texture.

$$\begin{aligned} L_s(\mathbf{D}_r^o, \mathbf{M}) &= \frac{1}{\sum_{u,v}\mathbf{M}(u,v)}\sum_{u,v}|\partial_u(\mathbf{M}(u,v)\mathbf{D}_{r,\star}^o(u,v))| \\ &+ \frac{1}{\sum_{u,v}\mathbf{M}(u,v)}\sum_{u,v}|\partial_v(\mathbf{M}(u,v)\mathbf{D}_{r,\star}^o(u,v))|, \end{aligned}$$

$$\mathbf{D}_{r,\star}^o(u,v) = \frac{\mathbf{M}(u,v)(\mathbf{D}_r^o(u,v))^{-1}\sum_{u',v'}\mathbf{M}(u',v')}{\sum_{u',v'}\mathbf{M}(u',v')(\mathbf{D}_r^o(u',v'))^{-1}}$$

Object-motion-sparsity loss [13].

$$L_g(\Delta_r) = 2\sum_{c\in\{1,2,3\}}\langle|\Delta_r^c|\rangle\sum_{u,v}\sqrt{1+\frac{\Delta_r^c(u,v)}{\langle|\Delta_r^c|\rangle}}, \quad (2)$$

$$\langle|\Delta_r^c|\rangle = \frac{\sum_{u,v}|\Delta_r^c(u,v)|}{HW}, \quad (3)$$

with c is the channel index of the motion map Δ_r.

Photometric loss [4] . We adopt the per-pixel minimum reprojection loss and the method to mask out stationary pixels proposed by [4]. Each training sample contains a triplet of images at three different timesteps $(\mathbf{I}_{t-1}, \mathbf{I}_t, \mathbf{I}_{t+1})$. The image at the middle timestep $\mathbf{I}_t$ is used as reference image, and the other two are used as source images. As a result, we have two reconstructed images $\mathbf{I}_{t\leftarrow t-1}$ and $\mathbf{I}_{t\leftarrow t+1}$. The first step is to derive the masks $\mathbf{M}_p$ that exclude stationary pixels from the final photometric loss.

$$\mathbf{M}_p(u,v) = [\min_{t'\in P} pe(\mathbf{I}_t, \mathbf{I}_{t\leftarrow t'})(u,v) < \min_{t'\in P} pe(\mathbf{I}_t, \mathbf{I}_{t'})(u,v)],$$

$$\begin{aligned} pe(\mathbf{I}_s, \mathbf{I}_r)(u,v) &= \frac{\alpha}{2}(1 - \mathrm{SSIM}(\mathbf{I}_r, \mathbf{I}_s)(u,v)) \\ &+ (1-\alpha)(|\mathbf{I}_r(u,v) - \mathbf{I}_s(u,v)|), \end{aligned}$$

with $P = \{t-1, t+1\}$ and α is a hyper-parameter which is set to 0.85 in our experiments. $\mathrm{SSIM}(\cdot)$ is a function to calculate image structural similarity index measure [18]. This function returns pixel-wise SSIM measure of shape $(H \times W \times 1)$. In addition, $pe(I_r, I_s) \in \mathbb{R}^{H\times W\times 1}$ is calculated as pixel-wise photometric error. The final photometric loss used in our framework is,

$$L_p = \frac{1}{\sum_{u,v}\mathbf{M}_p(u,v)}\sum_{u,v}\mathbf{M}_p(u,v)\min_{t'\in P}(pe(\mathbf{I}_{t\leftarrow t'}, \mathbf{I}_t)(u,v)).$$

1.2. Training the Scene Depth Network

The first stage of our proposed framework is to train (1) a depth network Θ^{scene} to predict depth $\mathbf{D}_r^{scene}$, (2) a camera pose network Φ^{cam} to predict camera pose $(\mathbf{R}^{cam}, \mathbf{T}^{cam})$, and (3) an object pixel-wise motion network Ψ to predict object pixel-wise motion Δ_r. However, jointly training all three networks at the beginning could cause a trivial solution in which the predicted camera pose is wrong (for example, $\mathbf{R}^{cam} = \mathbf{I}$, $\mathbf{T}^{cam} = \mathbf{0}$), and the motions of all pixels (including pixels in both static and dynamic regions) are accounted by the pixel-wise motion. Although correct pixel correspondences between the two input images could still be obtained, this phenomenon could lead to a degradation in depth prediction .

To avoid this issue, our strategy is to train the three networks as follows,

- We firstly train the depth network $\mathbf{D}_r^{scene}$ and pose network Φ^{cam} for Q epochs. In our experiments, we set $Q = 5$ for the Cityscapes dataset and $Q = 10$ for the KITTI dataset.
- After that, we freeze Θ^{scene} and Φ^{cam} and train only Ψ for 1 epoch. The purpose of this step is for Ψ to learn predicting object pixel-wise motion at the same scale as the depth predicted by Θ^{scene}.
- We then jointly train all the three networks together.

This training stategy, together with the object-motion-sparsity loss (Sec. 1.1) help to avoid the predicted pixel-wise motion Δ_r interfering with the learning process of the camera pose network Φ^{cam}.

1.3. Self-supervised Ground Segmentation

We first derive the pseudo ground masks $\hat{\mathbf{M}}_r^{gnd} \in \{0,1\}^{H\times W\times 1}$. Specifically, given a camera intrinsic $\mathbf{K}$, we map the predicted depth $\mathbf{D}_r^{scene}$ into a point cloud and use it for ground points detection following [19]. Then, we have $\mathbf{M}_r^{gnd}(u,v) = 1$ if the pixel (u,v) corresponds to a 3D point that belongs to the ground. $\mathbf{M}_r^{gnd}(u,v) = 0$, otherwise. The loss function used to train Υ_{gnd} has two terms: (1) a binary-cross entropy (BCE) loss between the model's prediction and the pseudo ground-truth masks, (2) the edge-aware smoothness loss applied to the predicted mask.

$$\mathbf{M}_r^{gnd} = \Upsilon^{gnd}(\mathbf{I}_r, \mathbf{D}_r^{scene}), \tag{4}$$

$$L_{gnd} = BCE(\mathbf{M}_r^{gnd}, \hat{\mathbf{M}}_r^{gnd}) + L_s(\mathbf{M}_r^{gnd}, \mathbf{I}_r). \tag{5}$$

After training, Υ_{gnd} is utilized to predict ground masks used in later stages. We set a threshold of 0.5 to convert the model predictions from soft masks to binary masks.

1.4. Self-supervised Object Detection

Training the self-supervised object detection model can be separated into two stages. In the first stage, we adopt the SlotAttention [14] model Υ_{label}^{obj} to derive pseudo ground-truth object masks. Since Υ_{label}^{obj} is only able to produce masks of dynamic objects, we use its predictions as pseudo label to train a MaskRCNN [9] model Υ^{obj} that is able to predict masks for both static and dynamic objects.

1.4.1 Training the Slot Attention Model Υ_{label}^{obj}

The input to the the Slot Attention model consists of the motion map Δ_r and the depth for dynamic regions which is derived based on the $\mathbf{D}_r^{scene}$ and Δ_r (see details below). The output of the framework includes the latent representation, the segmentation mask and the reconstruction corresponding to each dynamic object.

Unlike [14] which draws slot representation $\mathbf{q}_i$ from the one Gaussian distribution, we have two different distributions, one for background and one for foreground including L moving objects. As a result, the slot representation is initialized as below:

$$\mathbf{q}_{init}^0 \sim \mathcal{N}(\mu_{bg}, \phi_{bg}), \tag{6}$$

$$\{\mathbf{q}_{init}^l\}_{l\in\{1,2,\dots,L\}} \sim \mathcal{N}(\mu_{fg}, \phi_{fg}), \tag{7}$$

where L is a hyper-parameter indicating the maximum number of moving objects in an image, $\mathcal{N}$ is a Gaussian distribution whose mean and standard deviation are μ and σ. We set $L = 4$ in our experiments. The slot representations are then decoded into $(L+1)$ updated slot representation $\mathbf{q}^l$, $(L+1)$ soft-masks $\mathbf{M}_r^{l,label}$ and $(L+1)$ reconstructions $\mathbf{W}_r^{l,rec}$ for the input pixel-wise motion Δ_r and depth of the dynamic region.

$$\{\mathbf{M}_r^{l,label}, \mathbf{W}_r^{l,rec}, \mathbf{q}^l\}_{l\in\{0,1,\dots,L\}} = \Upsilon_{obj}^{label}(\mathbf{W}_r), \tag{8}$$

$$\mathbf{W}_r = \text{Concatenate}(\mathbf{M}_\Delta \odot \Delta_r, \mathbf{M}_\Delta \odot \mathbf{D}_r^{scene}), \tag{9}$$

$$\mathbf{M}_\Delta(u,v) = \begin{cases} 1 & \text{if } \|\Delta_r(u,v)\|_2^2 \geq a \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

with a is a hyper-parameter. The final reconstruction $\mathbf{W}_r^{rec}$ could be derived from the model's outputs, which is then used to compute the reconstruction loss.

$$\mathbf{W}_r^{rec} = \sum_{l\in\{0,1,\dots,L\}} \mathbf{M}_r^{l,label} \odot \mathbf{W}_r^{l,rec}. \tag{11}$$

Simply training the model with the reconstruction loss between $\mathbf{W}_r^{rec}$ and $\mathbf{W}_r$ causes an object to be over-segmented [14]. To avoid this, for each foreground mask with index $l \in \{1,2,\dots,L\}$, we predict a scalar $z^l \in [0,1]$, representing a probability that the mask is empty.

$$z^l = MLP(\mathbf{q}^l) \tag{12}$$

with *MLP* denotes a multi-layer perceptron network. The updated foreground mask $\tilde{\mathbf{M}}_r^{l,label}$ (with $l > 0$), the updated background mask $\tilde{\mathbf{M}}_r^{0,label}$ and the updated reconstruction $\tilde{\mathbf{W}}_r^{rec}$ are then obtained,

$$\tilde{\mathbf{M}}_r^{l,label}(u,v) = \frac{(z^l \mathbf{M}_r^{l,label}(u,v))^2}{\sum_{p\in\{1,\dots,L\}} z^p \mathbf{M}_r^{p,label}(u,v)}, \tag{13}$$

$$\tilde{\mathbf{M}}_r^{0,label} = 1 - \sum_{l \in \{1,2,\ldots,L\}} \tilde{\mathbf{M}}_r^{l,label}, \tag{14}$$

$$\tilde{\mathbf{W}}_r^{rec} = \sum_{l \in \{0,1,\ldots,L\}} \tilde{\mathbf{M}}_r^{l,label} \odot \mathbf{W}_r^{l,rec}. \tag{15}$$

Subsequently, we compute the reconstruction loss between the model's input and the two reconstructions.

$$L_{recon} = \frac{\sum_{u,v} (\mathbf{W}_r^{rec}(u,v) - \mathbf{W}_r(u,v))^2}{HW} + \frac{\sum_{u,v} (\tilde{\mathbf{W}}_r^{rec}(u,v) - \mathbf{W}_r(u,v))^2}{HW}.$$

Furthermore, we also encourage the predicted background mask to be similar to the masks of static region $(1 - \mathbf{M}_\Delta)$ derived from the predicted pixel-wise motion.

$$L_{bg} = BCE(\mathbf{M}_r^{0,label}, (1 - \mathbf{M}_\Delta)) + BCE(\tilde{\mathbf{M}}_r^{0,label}, (1 - \mathbf{M}_\Delta)).$$

We additionally utilize a sparsity loss for the predicted z^l. This loss encourages the predicted foreground mask to be empty, avoiding the over-segmentation of object masks.

$$L_z = \frac{1}{L} \sum_{l \in \{1,2,\ldots,L\}} z^l. \tag{16}$$

The final loss used to train Υ_{obj}^{label} is,

$$L_{slot} = L_{recon} + L_{bg} + L_z. \tag{17}$$

Since the input into this model only contain information in the dynamic region, Υ_{obj}^{label} fail to segment masks of static objects. For this purpose, we firstly set a threshold for the soft masks produced by Υ_{obj}^{label} and used them as pseudo ground-truth object masks for training a *MaskRCNN* model that is able to detect both static and dynamic objects.

1.4.2 Training the MaskRCNN Model Υ^{obj}

At this stage, we aim to use the masks produced by Υ_{obj}^{label} as pseudo ground-truth objects to train the MaskRCNN model Υ^{obj}. We make an observation that in the first few epochs, Υ^{obj} is able to discover more objects, then it tends to overfit to the noisy pseudo ground-truth masks and eventually fail to detect static objects in an image. To avoid this, Υ^{obj} is trained using the pseudo ground-truth mask for the first E epochs. After that, we follow a self-training strategy in which we use the model's predictions as a new set of pseudo ground-truth masks for training itself in the next training epoch. By doing this, the model is able to detect both static and dynamic objects.

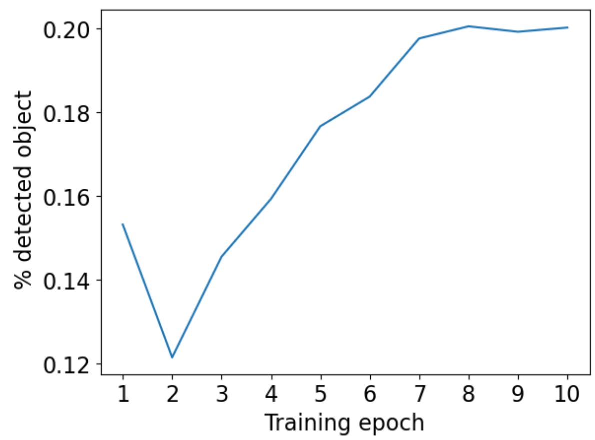

Figure 1. Percentage of objects being detected by ours self-supervised object detection model at each training epoch.

Fig. 1 shows the percentage of objects detected by our self-supervised object detection model in the Cityscapes dataset at each training epoch. For this dataset, we set $E = 2$ in our experiments, which means that the self-training strategy is applied starting from the third training epoch. From the figure, it can be seen that the detection rate of our model drop in the second iteration, indicating it starts over-fitting to the noisy pseudo ground-truth masks produced by Υ_{label}^{obj} and detecting less objects compared to the first epoch. By following the self-training method, our model is able to discover more objects from the third epoch to the seventh epoch. After that, the curve remains stable as the model converges to its own predictions.

Static/dynamic object classification. It is worth mentioning how we distinguish between static and dynamic object in our framework. Here, we define two function: (1) Intesect$(\mathbf{M}_1, \mathbf{M}_2)$ that takes two masks as input and return the number of pixels (u, v) such that $\mathbf{M}_1(u,v) = \mathbf{M}_2(u,v) = 1$, and (2) Size$(\mathbf{M}) = \sum_{u,v} \mathbf{M}(u,v)$ that computes a mask's size. Given these two functions, the object is classified as follows,

$$\text{Object mask } \mathbf{M}_r^o \text{ is: } \begin{cases} \text{dynamic} & \text{if } \frac{\text{Intesect}(\mathbf{M}_r^o, \mathbf{M}_\Delta)}{\text{Size}(\mathbf{M}_r^o)} \geq b \\ \text{static} & \text{otherwise} \end{cases}$$

with b is set to 0.5 in our experiments.

2. Additional Results on WaymoOpen and nuScene Dataset

We further train and evaluate our method on the WaymoOpen [15] and the nuScene [2] dataset. Similar to [16], we apply our method on top of two backbones of a depth estimation network: Monodepth2 [4], and LiteMono [1]. We use the same dataset split and object masks as done in

	Method	Dynamic obj		Static bg		All	
		Abs ↓	A1 ↑	Abs ↓	A1 ↑	Abs ↓	A1 ↑
Waymo	Monodepth2 (M) [4]	0.749	0.416	0.152	0.810	0.173	0.797
	Dynamo (M)[16]	0.234	0.674	0.122	0.862	0.130	0.851
	Ours (M)	**0.168**	**0.760**	**0.115**	**0.879**	**0.122**	**0.864**
	LiteMono (L) [1]	0.599	0.506	0.140	0.827	0.158	0.816
	Dynamo (L) [16]	0.194	0.750	0.110	0.891	0.116	0.878
	Ours (L)	**0.150**	**0.802**	**0.108**	**0.896**	**0.112**	**0.887**
NuScenes	Monodepth2 (M) [8]	0.418	0.570	0.447	0.735	0.425	0.723
	Dynamo (M)[16]	0.228	0.684	0.196	0.761	0.193	0.765
	Ours (M)	**0.183**	**0.764**	**0.178**	**0.828**	**0.172**	**0.828**
	LiteMono (L) [1]	0.502	0.517	0.431	0.734	0.419	0.720
	Dynamo (L) [16]	0.198	0.753	0.184	0.781	0.179	0.787
	Ours (L)	**0.179**	**0.758**	**0.181**	**0.830**	**0.175**	**0.830**

Table 1. Results on Waymo Open and nuScenes datasets. **M**: Monodepth2 [4] depth network backbone. **L**: LiteMono [1] depth network backbone. **Abs**: Absolute relative error. **A1**: Accuracy metric $\delta < 1.25$.

[16]. Note that our method is trained in a fully unsupervised manner and the object masks are only used for evaluation. Results in Tab. 1 shows that our method outperforms [1, 4, 16] by a large margin in dynamic regions and we are competitive in static regions. Fig. 2 shows examples of predicted object motions (visualized as optical flow maps) on two different scenes.

Figure 2. Predicted object motions (visualized as optical flow map)

3. More Quantitative Results

Here, we show the comparison between previous works and the models trained following our proposed framework with more evaluation metrics. Following prior works [4, 12, 13], for the Cityscape dataset, we use the ground-truth depth calculated from the provided disparity map, whereas KITTI's ground-truth depth is derived from sparse LiDAR points. The results for the Cityscapes dataset are shown from Tab. 2 to Tab. 4, and that for the KITTI dataset are shown from Tab. 5 to Tab. 7. For each table, the best results are highlighted in **bold**. Additionally, our models are underlined if it is better than all previous works. For dynamic regions in both datasets, our models outperform all prior works by large margins across different metrics. Moreover, we achieve the new state-of-the-art performance on the Cityscapes dataset.

4. More Quanlitative Results

More qualitative results on both the Cityscapes dataset (Fig. 3 and Fig. 4) and the KITTI dataset (Fig. 5 and Fig. 6) are shown in this section. Areas with more intense red color in the error maps represent higher error. Since the ground-truth depth used to evaluate the models on KITTI dataset are created from sparse reprojected LiDAR points, we use the improved ground-truth depth from [17] to visualize the error map. This set of ground-truth depth is more dense as it is accumulated from 5 consecutive LiDAR frames. However, [17] excludes moving objects from their computed ground-truth depth because the process of accumulating depth from consecutive frames causes errors for the dynamic regions. Therefore, for dynamic region, we fill the missing values in the improved ground-truth depth with those derived from applying SGM [10] to a pair of stereo images. The filled ground-truth depth is then used to visualize the error map for the KITTI dataset.

Dynamic Region (Cityscapes)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2* [4]		0.286	6.036	8.760	0.298	0.716	0.877	0.936
Li *et al.** [13]		0.188	1.654	5.341	0.220	0.735	0.93	0.976
InstaDM [12]	✓	0.189	2.538	5.723	0.216	0.795	0.932	0.972
RMDepth [11]		–	–	–	–	–	–	–
DaCNN [8]		–	–	–	–	–	–	–
ResNet18 [5] + Ours		0.107	0.784	3.790	0.145	0.907	0.970	0.987
ResNet18 [5] + Ours	✓	**0.100**	**0.647**	3.653	0.139	**0.911**	0.973	0.989
PackNet [6] + Ours		0.112	0.703	3.694	0.146	0.890	0.969	0.989
PackNet [6] + Ours	✓	0.104	0.666	3.650	0.141	0.901	0.972	0.989
DiffNet [20] + Ours		0.113	0.657	3.592	0.144	0.889	0.970	0.988
DiffNet [20] + Ours	✓	0.105	0.692	3.706	0.142	0.893	0.974	0.989
BrNet [7] + Ours		0.119	0.670	3.596	0.148	0.872	0.964	0.988
BrNet [7] + Ours	✓	0.106	0.577	**3.519**	**0.138**	0.892	**0.975**	**0.991**

Table 2. Comparison between our models and previous works on the Cityscapes dataset (dynamic region).

Static Region (Cityscapes)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2* [4]		0.119	1.461	6.601	0.173	0.877	0.968	0.989
Li *et al.** [13]		0.118	1.198	7.179	0.187	0.835	0.954	0.985
InstaDM [12]	✓	0.102	1.058	6.026	0.156	0.895	0.974	0.991
RMDepth [11]		–	–	–	–	–	–	–
DaCNN [8]		–	–	–	–	–	–	–
ResNet18 [5] + Ours		0.093	0.961	5.850	0.150	0.907	0.976	0.991
ResNet18 [5] + Ours	✓	0.091	0.931	5.962	0.151	0.904	0.975	0.991
PackNet [6] + Ours		0.090	0.858	5.750	0.146	0.909	0.977	0.992
PackNet [6] + Ours	✓	0.089	0.859	5.784	0.145	0.908	0.977	0.993
DiffNet [20] + Ours		0.083	0.775	5.537	0.139	0.919	0.980	0.993
DiffNet [20] + Ours	✓	0.082	0.781	5.467	0.136	0.921	0.981	**0.994**
BrNet [7] + Ours		**0.080**	0.736	5.400	0.136	0.922	0.980	0.993
BrNet [7] + Ours	✓	**0.080**	**0.717**	**5.392**	**0.135**	**0.923**	**0.982**	**0.994**

Table 3. Comparison between our models and previous works on the Cityscapes dataset (static region).

All Region (Cityscapes)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2* [4]		0.127	1.678	6.730	0.182	0.872	0.964	0.986
Li *et al.** [13]		0.119	1.186	7.052	0.188	0.833	0.953	0.985
InstaDM [12]	✓	0.106	1.104	5.994	0.161	0.890	0.972	0.990
RMDepth [11]		0.100	0.839	5.774	0.154	0.895	0.976	0.993
DaCNN [8]		0.113	1.380	6.305	–	0.888	–	–
ResNet18 [5] + Ours		0.094	0.931	5.751	0.150	0.906	0.975	0.991
ResNet18 [5] + Ours	✓	0.092	0.895	5.847	0.151	0.903	0.975	0.991
PackNet [6] + Ours		0.091	0.831	5.647	0.147	0.908	0.976	0.992
PackNet [6] + Ours	✓	0.090	0.830	5.679	0.146	0.908	0.977	0.993
DiffNet [20] + Ours		0.085	0.753	5.435	0.140	0.916	0.979	0.993
DiffNet [20] + Ours	✓	0.083	0.757	5.375	0.137	0.919	0.980	0.993
BrNet [7] + Ours		0.084	0.722	5.305	0.138	0.918	0.979	0.993
BrNet [7] + Ours	✓	**0.081**	**0.696**	**5.293**	**0.135**	**0.921**	**0.981**	**0.993**

Table 4. Comparison between our models and previous works on the Cityscapes dataset (all region).

Dynamic Region (KITTI)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
PackNet [6]		0.213	2.820	6.361	0.312	0.762	0.866	0.923
DiffNet [20]		0.177	2.072	5.942	0.298	0.792	0.889	0.930
BrNet* [7]		0.183	1.715	5.673	0.289	0.760	0.891	0.937
RMDepth [11]		–	–	–	–	–	–	–
DaCNN [8]		–	–	–	–	–	–	–
DiffNet [20] + Ours		0.158	1.468	5.288	0.275	0.838	0.910	0.938
DiffNet [20] + Ours	✓	**0.143**	**1.191**	**5.095**	0.268	**0.845**	**0.911**	0.941
BrNet [7] + Ours		0.160	1.273	5.157	0.269	0.812	0.906	0.943
BrNet [7] + Ours	✓	0.162	1.207	5.099	**0.267**	0.812	0.907	**0.945**

Table 5. Comparison between our models and previous works on the KITTI dataset (dynamic region).

Static Region (KITTI)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
PackNet [6]		0.108	0.814	4.647	0.184	0.883	0.962	0.983
DiffNet [20]		0.101	0.743	4.444	**0.176**	**0.899**	**0.967**	0.984
BrNet* [7]		0.104	0.702	4.530	0.179	0.885	0.964	**0.985**
RMDepth [11]		–	–	–	–	–	–	–
DaCNN [8]		–	–	–	–	–	–	–
DiffNet [20] + Ours		0.101	0.687	**4.416**	0.177	0.894	0.966	0.984
DiffNet [20] + Ours	✓	**0.099**	**0.658**	4.471	**0.176**	0.895	0.966	0.984
BrNet [7] + Ours		0.103	0.690	4.464	0.177	**0.889**	0.964	**0.985**
BrNet [7] + Ours	✓	0.102	0.673	4.496	**0.176**	**0.889**	0.965	**0.985**

Table 6. Comparison between our models and previous works on the KITTI dataset (static region).

All Region (KITTI)								
Method	Sematic prior	Error metrics				Accuracy metrics		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
PackNet [6]		0.110	0.834	4.661	0.188	0.881	0.960	0.982
DiffNet [20]		0.102	0.753	4.459	0.179	0.897	0.965	0.983
BrNet* [7]		0.106	0.711	4.536	0.182	0.884	0.963	0.984
RMDepth [11]		0.108	0.710	4.513	0.183	0.884	0.964	0.983
DaCNN [8]		**0.099**	**0.661**	**4.316**	**0.173**	**0.897**	**0.967**	**0.985**
DiffNet [20] + Ours		0.102	0.693	4.422	0.180	0.892	0.965	0.983
DiffNet [20] + Ours	✓	0.100	0.662	4.473	0.179	0.894	0.965	0.983
BrNet [7] + Ours		0.103	0.692	4.464	0.179	0.888	0.963	0.984
BrNet [7] + Ours	✓	0.103	0.675	4.492	0.179	0.888	0.964	0.984

Table 7. Comparison between our models and previous works on the KITTI dataset (all region).

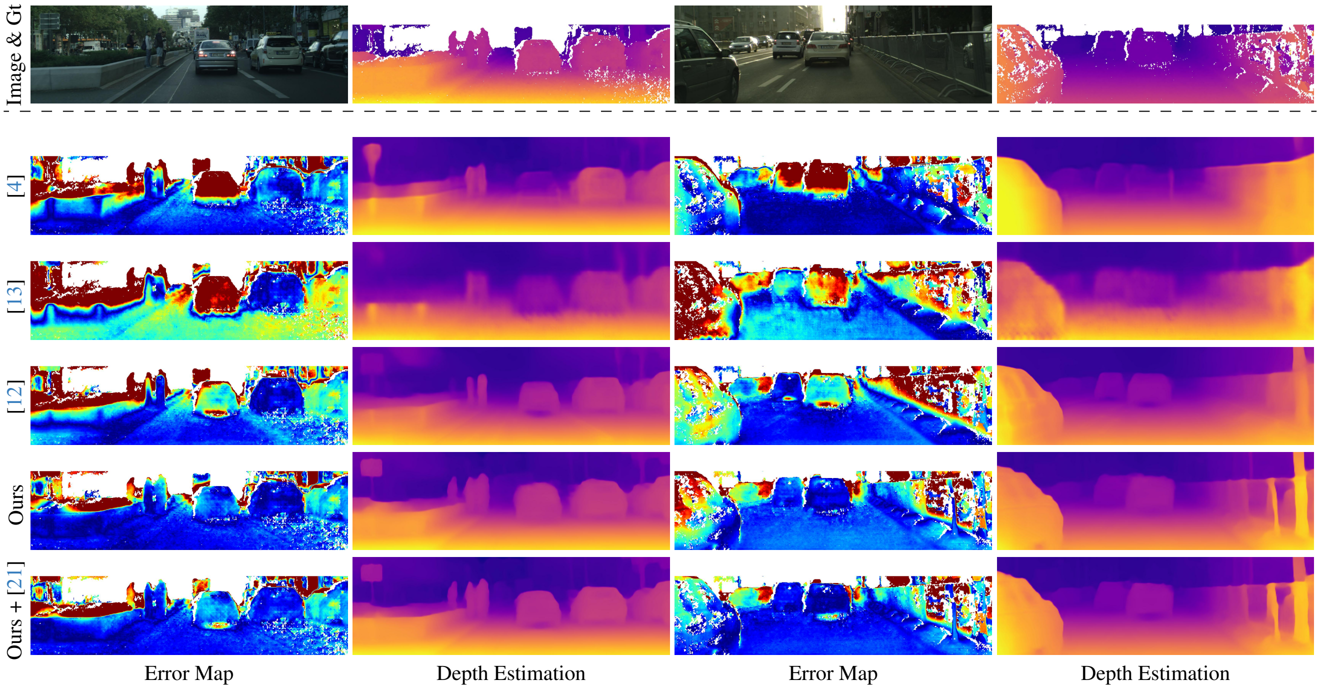

Figure 3. Qualitative results 1 (Cityscapes).

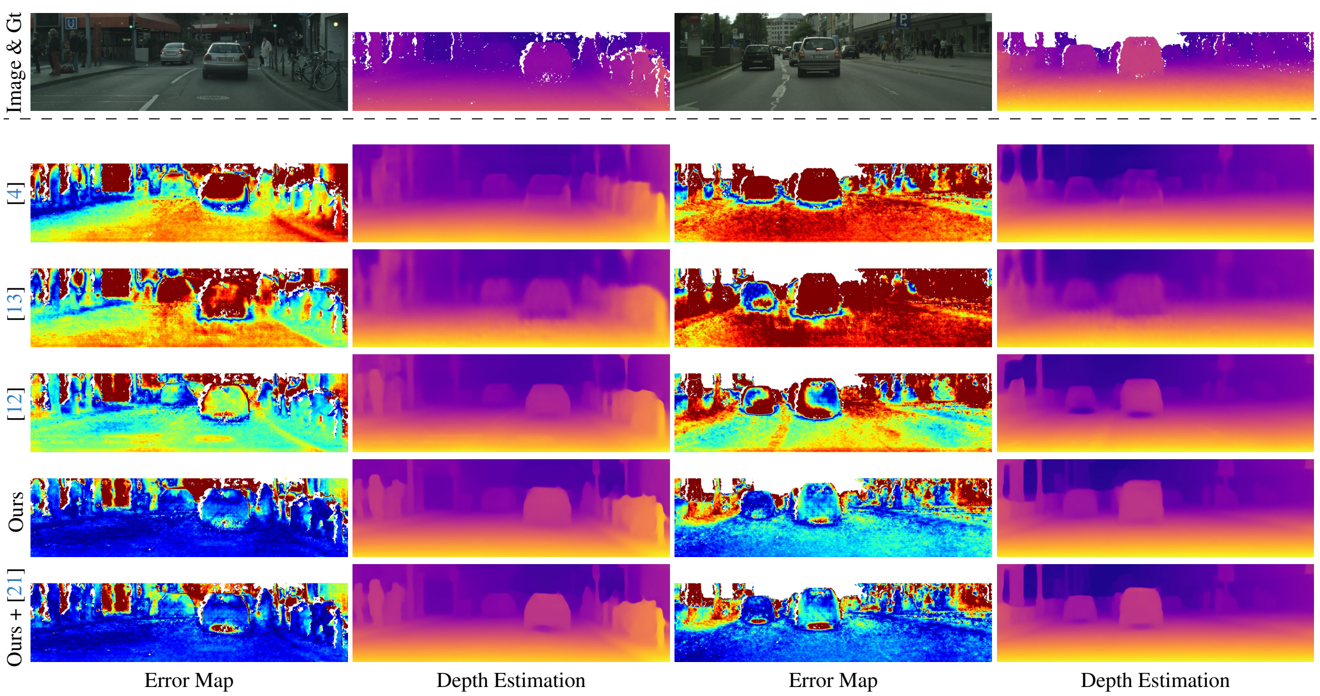

Figure 4. Qualitative results 2 (Cityscapes).

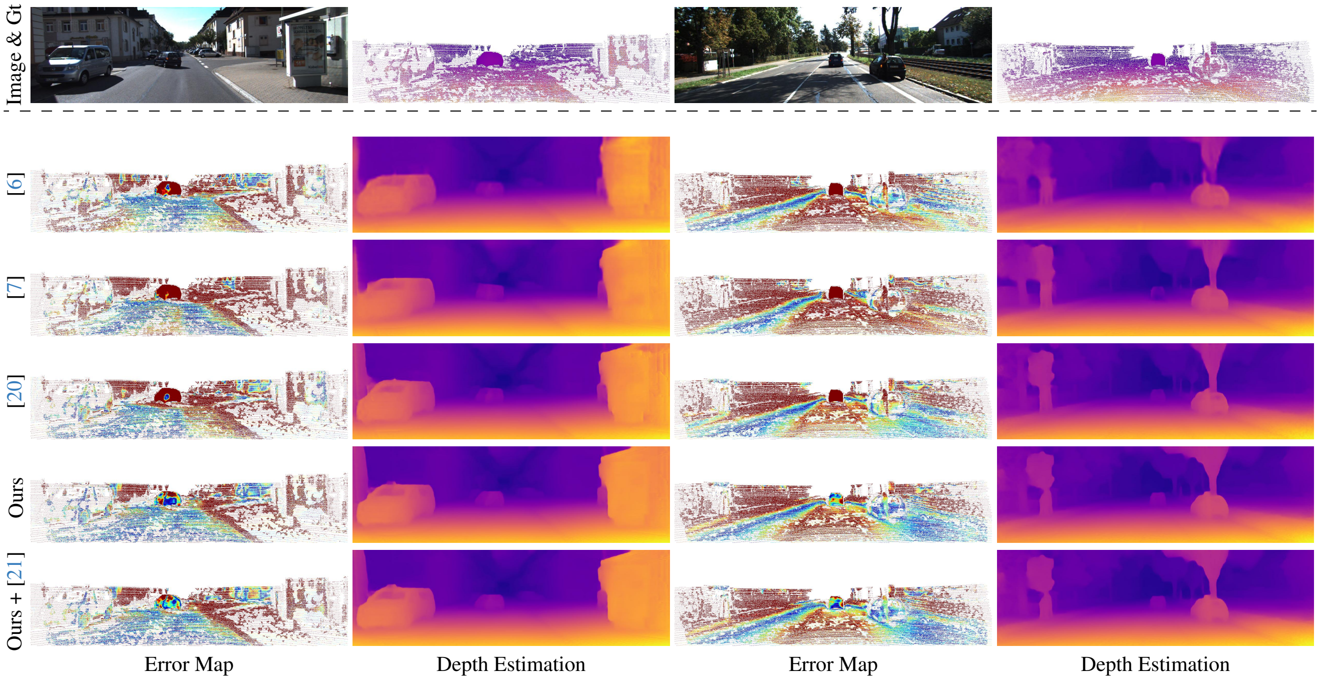

Figure 5. Qualitative results 1 (KITTI).

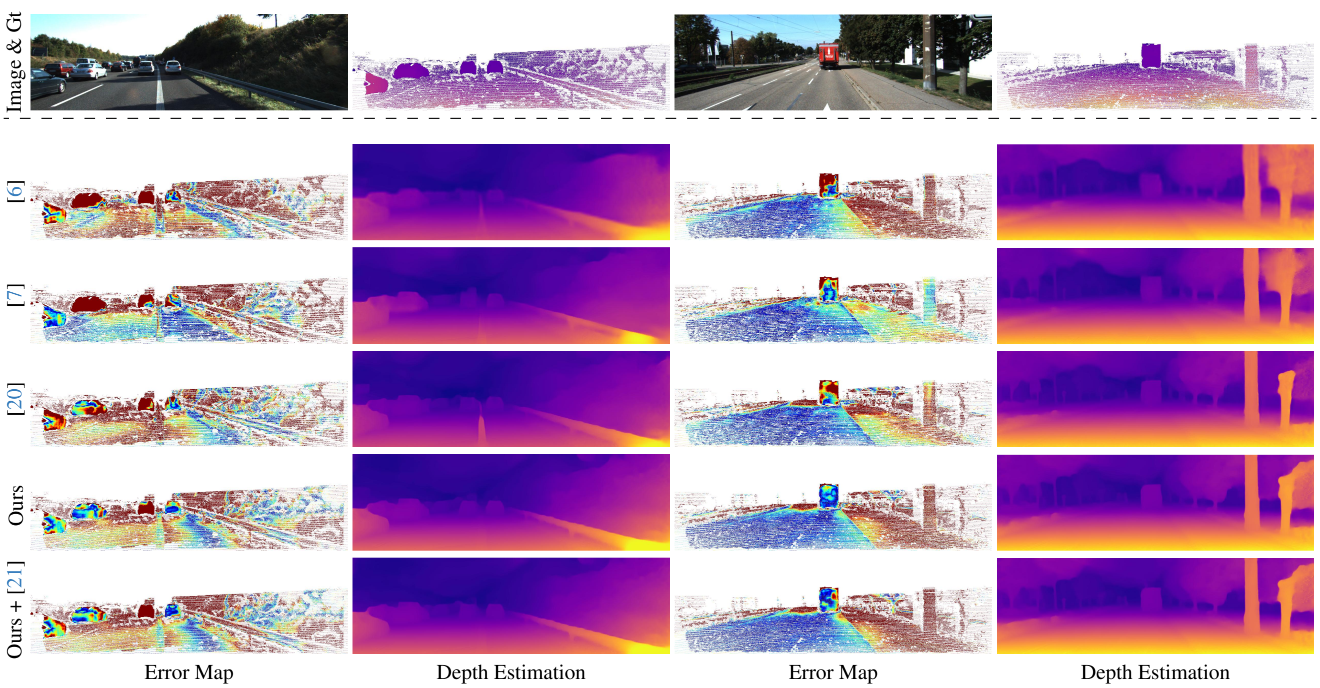

Figure 6. Qualitative results 1 (KITTI).

References

[1] Lite-mono: A lightweight cnn and transformer architecture for self-supervised monocular depth estimation. In *CVPR*, 2023. 3, 4

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar 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. 3

[3] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 1

[4] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019. 1, 3, 4, 5, 7

[5] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 5

[6] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2485–2494, 2020. 5, 6, 8

[7] Wencheng Han, Junbo Yin, Xiaogang Jin, Xiangdong Dai, and Jianbing Shen. Brnet: Exploring comprehensive features for monocular depth estimation. In *European Conference on Computer Vision*, pages 586–602. Springer, 2022. 5, 6, 8

[8] Wencheng Han, Junbo Yin, and Jianbing Shen. Self-supervised monocular depth estimation by direction-aware cumulative convolution network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8613–8623, 2023. 5, 6

[9] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[10] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 4

[11] Tak-Wai Hui. Rm-depth: Unsupervised learning of recurrent monocular depth in dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1675–1684, 2022. 5, 6

[12] Seokju Lee, Sunghoon Im, Stephen Lin, and In So Kweon. Learning monocular depth in dynamic scenes via instance-aware projection consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1863–1872, 2021. 4, 5, 7

[13] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Unsupervised monocular depth learning in dynamic scenes. In *Conference on Robot Learning*, pages 1908–1917. PMLR, 2021. 1, 4, 5, 7

[14] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020. 2

[15] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 3

[16] Yihong Sun and Bharath Hariharan. Dynamo-depth: Fixing unsupervised depth estimation for dynamical scenes. *Advances in Neural Information Processing Systems*, 36, 2024. 3, 4

[17] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017. 4

[18] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[19] Feng Xue, Guirong Zhuo, Ziyuan Huang, Wufei Fu, Zhuoyue Wu, and Marcelo H Ang. Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2330–2337. IEEE, 2020. 2

[20] Hang Zhou, David Greenwood, and Sarah Taylor. Self-supervised monocular depth estimation with internal feature fusion. *arXiv preprint arXiv:2110.09482*, 2021. 5, 6, 8

[21] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *arXiv preprint arXiv:2304.06718*, 2023. 7, 8