ECCV2020: Supplemental Material on

Multi-Loss Rebalancing Algorithm for Monocular Depth Estimation

Jae-Han Lee[0000−0002−3674−4023], and Chang-Su Kim[0000−0002−4276−1831]

School of Electrical Engineering, Korea University, Korea
jaehanlee@mcl.korea.ac.kr, changsukim@korea.ac.kr

S-1 Loss Function Space

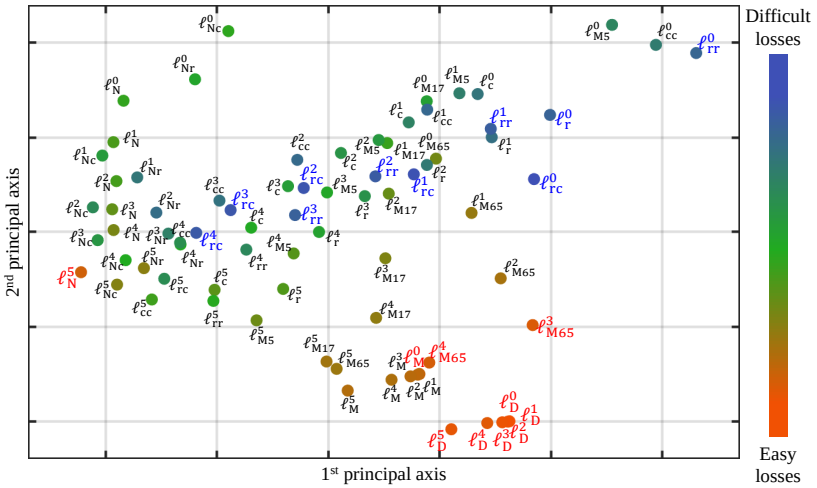


Fig. S-1. Visualization of the difficulty of each loss in the loss function space. The ten 'easiest' losses are in red fonts, while the ten 'most difficult' losses are in blue fonts.

Fig. S-1 visualizes the training difficulty of each loss in the loss function space. To quantify the difficulty, we do the weight initialization in Section 3.2 in the main paper, but do not perform the weight rebalancing (*i.e.* $\lambda = 0$). Then, after training, we compute the reduction in each loss, defined as

$$\bar{\Delta} L_i = w_i^{(0)} (L_i^{(T)} - L_i^{(0)}) \tag{S-1}$$

where T is the number of training periods. We find that the depth loss ℓ_{D}^0 at the finest scale is the easiest one with the most reduction. Table S-1 lists the scaled reduction of each loss, where the scale factor is determined to normalize the reduction of the easiest ℓ_{D}^0 to 1. Hence, a higher value means that the corresponding loss is an easy one. In Table S-1 and Fig. S-1, the ten 'easiest' losses are in red fonts, while the ten 'most difficult' ones are in blue. We see that all depth losses are on the easier side, while gradient losses at fine scales are on the difficult side. Also, fine scale losses tend to be more difficult than coarse ones.

Table S-1. Loss reduction rate of functions compared to the easiest loss function ℓ_{D}^{0}.

	Spatial scale											
	0		1		2		3		4		5	
Depth losses	ℓ_{D}^{0}	**1.000**	ℓ_{D}^{1}	0.999	ℓ_{D}^{2}	0.999	ℓ_{D}^{3}	0.997	ℓ_{D}^{4}	0.990	ℓ_{D}^{5}	0.996
Mean-removed losses	ℓ_{M}^{0}	0.864	ℓ_{M}^{1}	0.863	ℓ_{M}^{2}	0.862	ℓ_{M}^{3}	0.859	ℓ_{M}^{4}	0.850	ℓ_{M}^{5}	0.852
	ℓ_{M5}^{0}	0.499	ℓ_{M5}^{1}	0.488	ℓ_{M5}^{2}	0.517	ℓ_{M5}^{3}	0.568	ℓ_{M5}^{4}	0.625	ℓ_{M5}^{5}	0.685
	ℓ_{M17}^{0}	0.537	ℓ_{M17}^{1}	0.592	ℓ_{M17}^{2}	0.676	ℓ_{M17}^{3}	0.736	ℓ_{M17}^{4}	0.767	ℓ_{M17}^{5}	0.795
	ℓ_{M65}^{0}	0.707	ℓ_{M65}^{1}	0.795	ℓ_{M65}^{2}	0.849	ℓ_{M65}^{3}	0.907	ℓ_{M65}^{4}	0.906	ℓ_{M65}^{5}	0.808
Gradient losses	ℓ_{r}^{0}	0.448	ℓ_{r}^{1}	0.461	ℓ_{r}^{2}	0.475	ℓ_{r}^{3}	0.505	ℓ_{r}^{4}	0.551	ℓ_{r}^{5}	0.624
	ℓ_{c}^{0}	0.468	ℓ_{c}^{1}	0.491	ℓ_{c}^{2}	0.507	ℓ_{c}^{3}	0.532	ℓ_{c}^{4}	0.569	ℓ_{c}^{5}	0.636
	ℓ_{rr}^{0}	0.453	ℓ_{rr}^{1}	0.424	ℓ_{rr}^{2}	0.422	ℓ_{rr}^{3}	0.445	ℓ_{rr}^{4}	0.499	ℓ_{rr}^{5}	0.591
	ℓ_{rc}^{0}	0.340	ℓ_{rc}^{1}	0.355	ℓ_{rc}^{2}	0.364	ℓ_{rc}^{3}	0.380	ℓ_{rc}^{4}	0.421	ℓ_{rc}^{5}	0.504
	ℓ_{cc}^{0}	0.477	ℓ_{cc}^{1}	0.457	ℓ_{cc}^{2}	0.458	ℓ_{cc}^{3}	0.467	ℓ_{cc}^{4}	0.504	ℓ_{cc}^{5}	0.593
Normal losses	ℓ_{N}^{0}	0.591	ℓ_{N}^{1}	0.609	ℓ_{N}^{2}	0.607	ℓ_{N}^{3}	0.645	ℓ_{N}^{4}	0.725	ℓ_{N}^{5}	0.890
	ℓ_{Nr}^{0}	0.553	ℓ_{Nr}^{1}	0.474	ℓ_{Nr}^{2}	0.459	ℓ_{Nr}^{3}	0.491	ℓ_{Nr}^{4}	0.580	ℓ_{Nr}^{5}	0.758
	ℓ_{Nc}^{0}	0.591	ℓ_{Nc}^{1}	0.520	ℓ_{Nc}^{2}	0.500	ℓ_{Nc}^{3}	0.508	ℓ_{Nc}^{4}	0.579	ℓ_{Nc}^{5}	0.755

Fig. S-2 visualizes the correlation between losses. We see that depth losses are less correlated with the other losses. This is consistent with the visualization of the loss function space in Fig. S-1, in which depth losses are far from the other losses. Also, each blue box includes correlation coefficients between the same type of losses. Depth losses are highly correlated to one another. Similarly, global-mean-removed losses are highly correlated. On the other hand, normal losses are less correlated to one another.

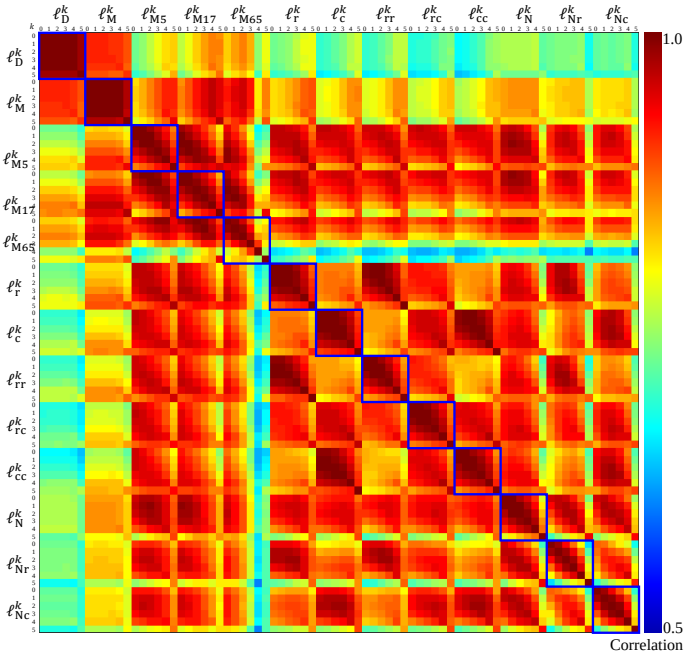


Fig. S-2. Visualization of the correlation matrix between the 78 losses. Correlation coefficients between the same type of losses are within blue boxes.

S-2 More Experimental Results

Experiments on KITTI [11]: We test the proposed algorithm on the KITTI dataset. For evalution, we follow the Eigen *et al.*'s protocol [9], including the split of training and test data. Table S-2 compares the proposed algorithm with conventional algorithms. KITTI provides the depths of sparse pixels only. Thus, many losses (local-mean-removed losses, gradient losses, and normal losses) and relative depth maps, which consider the depth information of adjacent areas, are not effective during training. Also, since the outdoor scenes include sky areas, mean-removed losses may not be computed. In spite of these disadvantages, the proposed algorithm provides comparable performance to the conventional algorithms.

Table S-2. Performance comparison on the KITTI dataset [11]. * Kuznietsov *et al.*, Semi-supervised deep learning for monocular depth map prediction, CVPR 2017.

	Experimental results of KITTI [11]					
	δ_1	δ_2	δ_3	RMSE_{lin}	RMSE_{log}	ARD
Eigen *et al.* [9]	69.2%	89.9%	96.7%	7.156	0.270	0.190
Liu *et al.* [34]	65.6%	88.1%	95.8%	7.046	-	0.217
Godard *et al.* [12]	86.1%	94.9%	97.6%	4.935	0.206	0.114
Kuzniestsov *et al.**	86.2%	96.0%	98.6%	4.621	0.189	0.113
Fu *et al.* [10]	**93.2%**	**98.4%**	**99.4%**	**2.727**	**0.120**	**0.072**
Proposed	86.4%	96.2%	98.6%	4.512	0.176	0.115

Different backbone networks: In this test, we verify that the proposed multi-loss rebalancing algorithm is effective regardless of the backbone network. To this end, we replace the encoder backbone with four widely-used networks: VGG16, ResNet50, ResNet152, and SENet154. Table S-3 lists the performance of each backbone. All models are trained for 9 epochs, except for the default model (PNAS) in the paper. For comparison, Table S-3 also includes the results of Eigen and Fergus [8], Laina *et al.* [25], Lee *et al.* [27], Hu *et al.* [16], which use VGG16, ResNet50, ResNet152, and SENet154 as the backbones, respectively. We see that, for each backbone, the proposed algorithm outperforms the conventional algorithm using the same backbone.

Qualitative comparison on NYUv2 [41]: Fig. S-3 and Fig. S-4 qualitatively compare depth estimation results of the proposed algorithm and the conventional algorithms [3,8-10,16,25,28]. For easier comparison, depth maps (odd rows) and error maps (even rows) are presented. Error maps visualize errors according to the δ_n metrics. Specifically, yellow, red and black areas indicate areas where the ratio between estimated depth and ground-truth depth is greater than 1.25, 1.25^2 and 1.25^3, respectively. Also, areas satisfying the δ_1 criterion are shown in green, and areas with no error are shown in blue. Fig. S-3 compares normal cases, while Fig. S-4 is for difficult images with relatively big errors. The proposed algorithm

Table S-3. Performance comparison of the proposed algorithm using different backbone networks. The conventional algorithms using the same backbones are also compared. [8], [25], [27], and [16] adopt VGG16, ResNet50, ResNet152, and SENet154 as the backbone networks, respectively.

	The higher, the better			The lower, the better					
	δ_1	δ_2	δ_3	RMSE_{lin}	ARD	log10	RMSE_{log}	RMSE_{si}	SRD
Proposed (VGG16)	77.2%	95.0%	99.0%	0.544	0.160	0.067	0.196	0.160	0.117
Proposed (ResNet50)	82.4%	96.3%	99.1%	0.482	0.138	0.058	0.174	0.142	0.094
Proposed (ResNet152)	86.0%	97.5%	99.4%	0.445	0.120	0.052	0.156	0.132	0.076
Proposed (SENet154)	**87.1%**	**97.5%**	**99.4%**	**0.426**	0.116	**0.049**	**0.149**	**0.123**	**0.074**
Proposed (PNAS)	87.0%	97.4%	99.3%	0.430	0.119	0.050	0.151	**0.123**	0.078
[8] (VGG16)	76.9%	95.0%	98.8%	0.641	0.158	-	0.214	0.171	0.121
[25] (ResNet50)	81.1%	95.3%	98.8%	0.573	0.127	0.055	0.195	-	-
[27] (ResNet152)	81.5%	96.3%	99.2%	0.572	0.139	-	0.193	-	0.096
[16] (SENet154)	86.6%	**97.5%**	99.3%	0.530	**0.115**	0.050	-	-	-

provides more reliable results than the conventional algorithms. The differences are more clearly observed in the challenging cases in Fig. S-4.

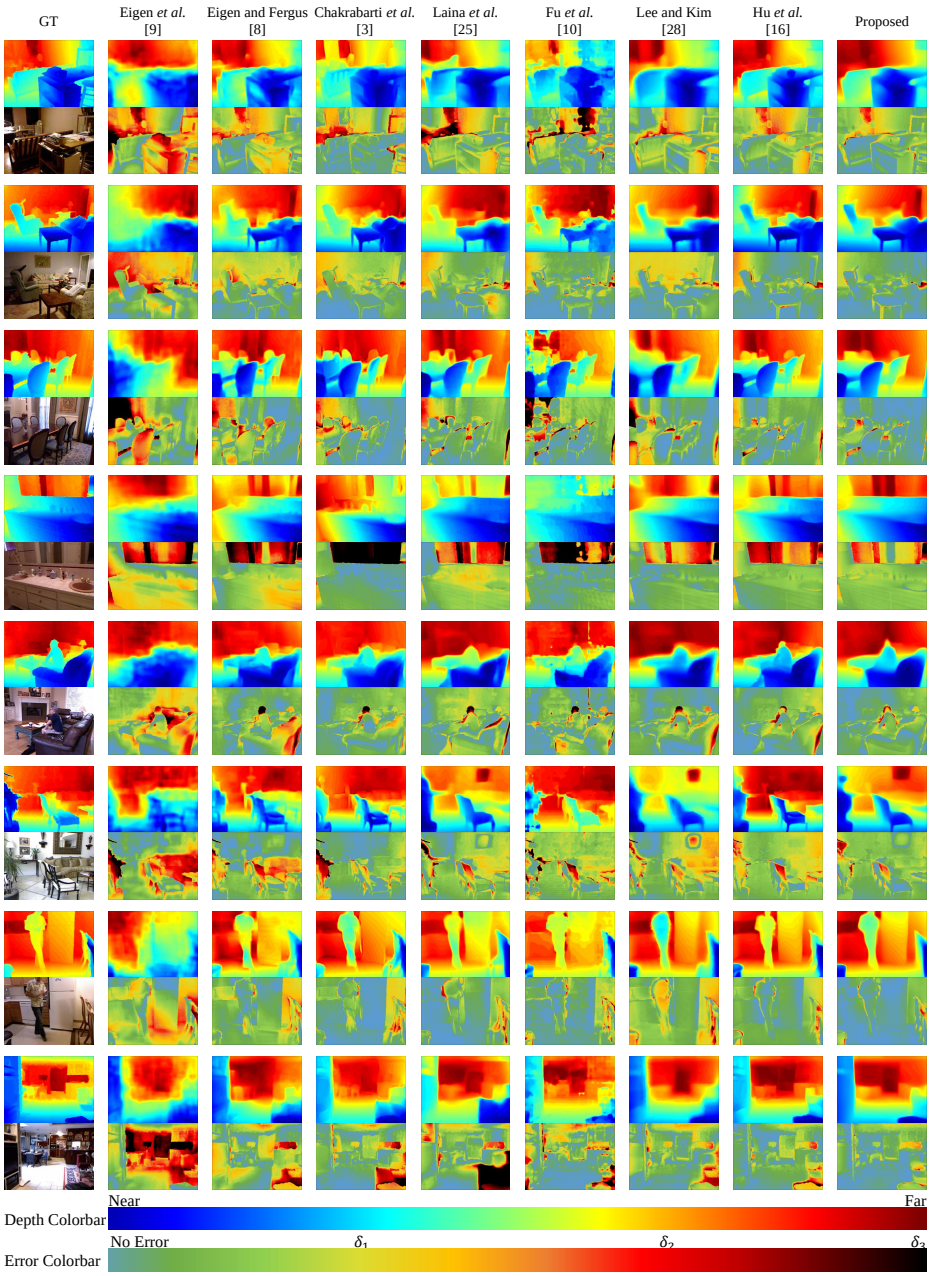


Fig. S-3. Qualitative comparison of depth estimation results on NYUv2. For easier comparison, depth maps (odd rows) and error maps (even rows) are provided. Relatively easy cases are shown.

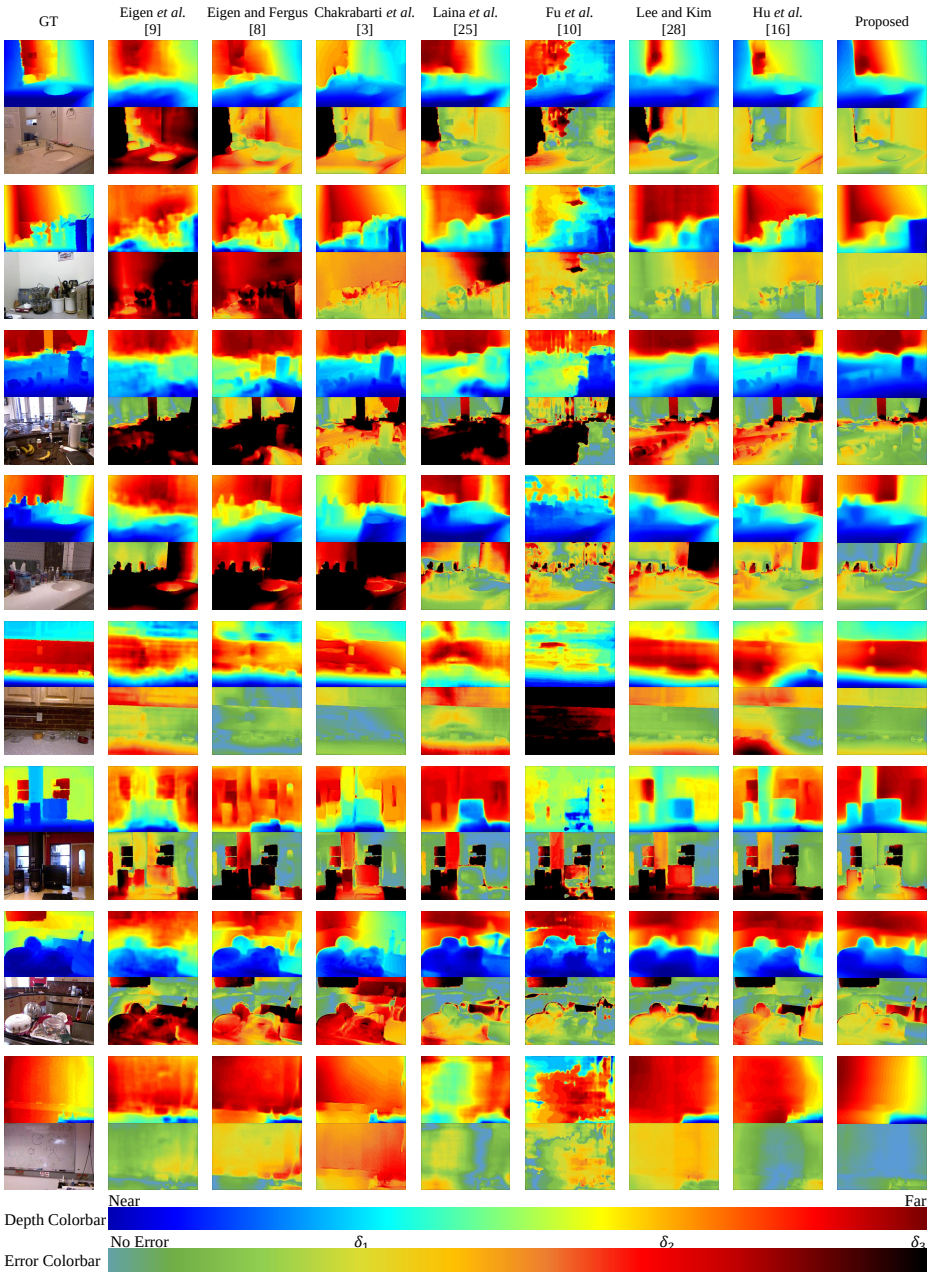


Fig. S-4. Qualitative comparison of depth estimation results on NYUv2. For easier comparison, depth maps (odd rows) and error maps (even rows) are provided. Relatively hard cases are shown.

S-3 Relative Depth Estimation

In addition to ordinary depth estimation (ODE), we apply the proposed algorithm to relative depth estimation (RDE). More specifically, we define four kinds of relative depth maps at four scales ($2 \sim 5$). Thus, there are 16 kinds of relative depth maps in total, which are shown in Fig. S-5. Note that we do not use the two finest scales 0 and 1, which incur high decoding complexity. Each relative depth map is named using the same conventions as the loss functions in Table 1. For example, $\mathbf{R}^2_{\mathrm{M17}}$ is defined as the relative depth map using the 17×17 average filter at scale 2,

$$\mathbf{R}^2_{\mathrm{M17}} = \log \mathbf{D}^2 - \log \mathbf{D}^2 \circledast \frac{\mathbf{J}_{17}}{17^2}. \tag{S-2}$$

Unlike the difference in (5), we use the log-difference for the relative depth representation in (S-2), which makes the relative depth between 1m and 10m and that between 10m and 100m treated equally. Also, we have $\mathbf{R}^5_{\mathrm{r}} = \nabla_{\mathrm{r}} \log \mathbf{D}^5$. The other relative depth maps are defined similarly. Note that $\mathbf{R}_{\mathrm{r}}$ and $\mathbf{R}_{\mathrm{c}}$ represent details in the horizontal and vertical directions, respectively, and $\mathbf{R}_{\mathrm{M}}$ records the ratio of each depth to the local average depth.

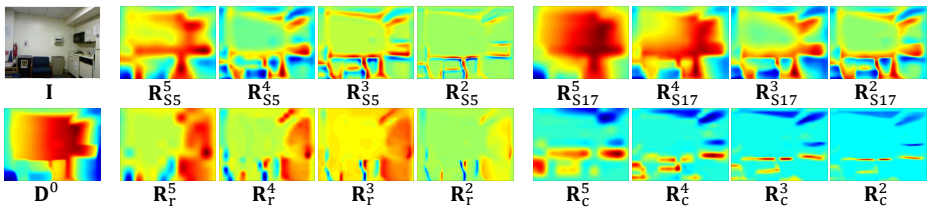


Fig. S-5. Examples of 16 relative depth maps. The corresponding image $\mathbf{I}$ and ordinary depth map $\mathbf{D}^0$ are also shown. Farther depths are in red, while closer ones in blue.

We design 16 decoders to estimate the 16 kinds of relative depth maps, but all the decoders share the same encoder that is designed for the ordinary depth estimator. The relative depth estimator is trained in the same way as the ordinary depth estimator. Section S-4 describes the decoder structure of the relative depth estimator.

We examine data distributions of various RGBD datasets. Fig. S-6 shows the distributions of ordinary depths and relative depths in five public datasets [11,39,41,43,45]. It also shows the symmetric KL-divergence between the datasets. For ordinary depths in $\mathbf{D}^0$, there are big differences between datasets. Especially, depths in the outdoor datasets KITTY and Make3D tend to be much bigger than those in the indoor ones NYUv2 and SUNRGB-D. In contrast, relative depth distributions in all datasets are quite similar to one another.

The relative depth estimator is trained by employing three indoor depths [4,41,43], two outdoor depths [11,39] and two relative depths [5,45] datasets jointly, without additional processing, such as the domain adaptation [24]. As

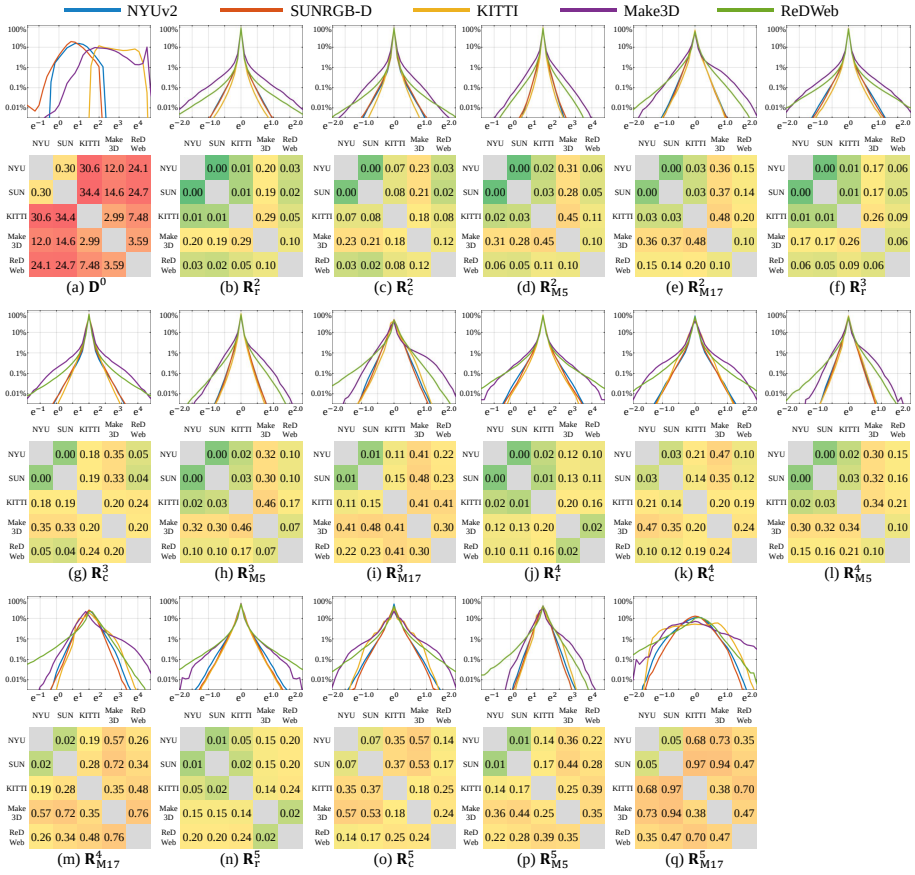
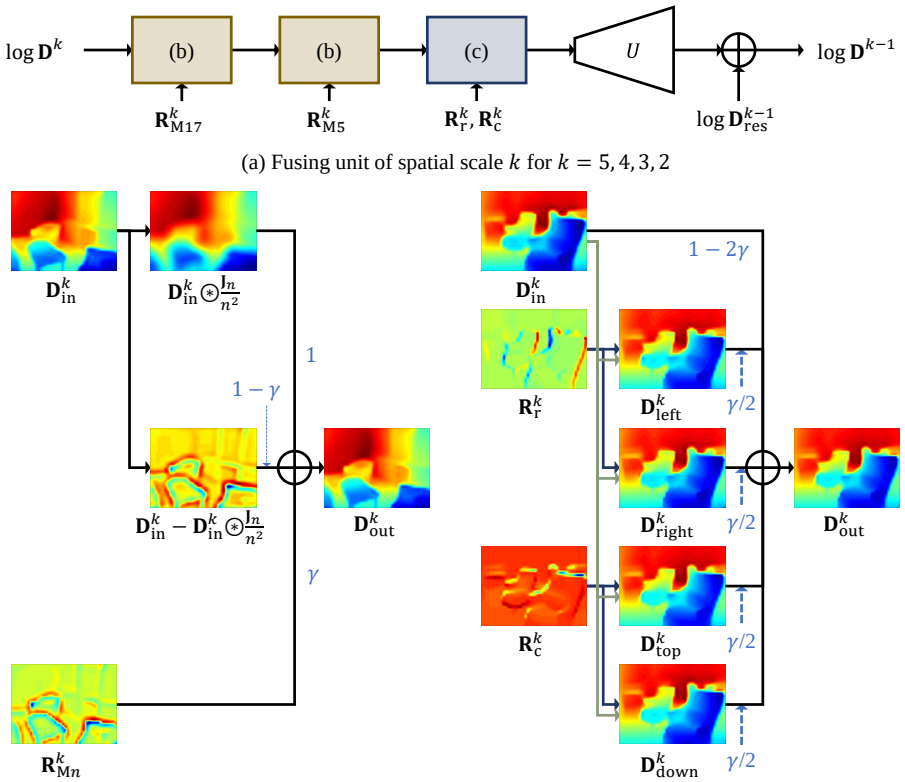


Fig. S-6. The odd rows compare the distributions of ordinary depths in (a) and relative depths in (b)∼(q) for five public datasets NYUv2 [41], SUNRGB-D [43], KITTI [11], Make3D [39], and ReDWeb [45]. The even rows show the symmetric KL-divergence between the datasets.

mentioned previously, relative depths have similar characteristics regardless of datasets. Therefore, we develop only a single relative depth estimator using the PNASNet backbone [33] and train it using all possible training data in all datasets, except for the DIW dataset [5]. DIW provides depths labels only for a pair of pixels per image. Thus we use DIW only for evaluation.

Fusing ordinary and relative depth maps: The ordinary depth estimator and the relative depth estimator estimate a single ordinary depth map $\mathbf{D}^0$ and 16 relative depth maps, respectively. The 16 relative depth maps convey different depth components. The union of those components is equivalent to the components of $\mathbf{R}^2$, which is $\mathbf{D}^2$ with the global scale removed. Thus, we fuse the 16 maps to $\mathbf{R}^2$ and interpolate it to obtain the finest relative depth map $\mathbf{R}^0$.



(a) Fusing unit of spatial scale k for $k = 5, 4, 3, 2$

(b) Module for combining $\mathbf{R}^k_{\mathrm{Mn}}$, n =17, 5

(c) Module for combining $\mathbf{R}^k_{\mathrm{r}}$ and $\mathbf{R}^k_{\mathrm{c}}$

Fig. S-7. At spatial scale k, each refining unit in (a) uses two modules in (b) and (c) to combine relative depth maps and yield a refined depth map at scale $k-1$.

Finally, we fuse the information in $\mathbf{R}^0$ with the ordinary depth map $\mathbf{D}^0$ to refine it.

The ordinary map $\mathbf{D}^0$ has a higher resolution than the relative ones. Thus, we decompose $\mathbf{D}^0$ into the low-resolution map $\mathbf{D}^5$ at scale 5 and the four residual maps $\mathbf{D}^k_{\mathrm{res}}$, $0 \le k \le 4$, by adopting the decomposition scheme in [28];

$$\log \mathbf{D}^k_{\mathrm{res}} = \log \mathbf{D}^k - U\left(\log \mathbf{D}^{k+1}\right) \tag{S-3}$$

where U is the bilinear interpolation to double the size of a depth map horizontally and vertically. Then, $\mathbf{D}^5$ is combined with the four relative depth maps $\mathbf{R}^5_{\mathrm{M17}}$, $\mathbf{R}^5_{\mathrm{M5}}$, $\mathbf{R}^5_{\mathrm{r}}$, $\mathbf{R}^5_{\mathrm{c}}$ at the same scale, as well as the residual map $\mathbf{D}^4_{\mathrm{res}}$, to yield a refined $\mathbf{D}^4$. This is done by the fusing unit in Fig. S-7 (a).

Given $\mathbf{D}^k$ at scale k, the fusing unit uses two types of modules in Fig. S-7 (b) and (c) to combine the relative maps, performs the bilinear interpolation, and then adds the residual map $\mathbf{D}^{k-1}_{\mathrm{res}}$ to yield a refined $\mathbf{D}^{k-1}$ at scale $k-1$. This is repeated from $k = 5$ to 2 to yield a refined $\mathbf{D}^1$. Since there is no relative depth

map at scale 1, the last fusing unit simply performs the bilinear interpolation and adds the residual map $\mathbf{D}^0_{\text{res}}$ to obtain a refined $\mathbf{D}^0$.

For each module in Fig. S-7 (b) or (c), let $\mathbf{D}^k_{\text{in}}$ and $\mathbf{D}^k_{\text{out}}$ denote the input and output logarithmic depth maps, which are approximations of $\log \mathbf{D}^k$, respectively. In Fig. S-7 (b), $\mathbf{D}^k_{\text{in}}$ is decomposed to the low-frequency component $\mathbf{D}^k_{\text{in}} \circledast \frac{\mathbf{J}_n}{n^2}$ and the high-frequency component $\mathbf{D}^k_{\text{in}} - \mathbf{D}^k_{\text{in}} \circledast \frac{\mathbf{J}_n}{n^2}$. Since the high-frequency component is computed in the same way as Eq. (13) in the paper, it should convey the same information as $\mathbf{R}^k_{\text{M}n}$ ideally. But, they are different in practice due to depth estimation errors. Thus, these two approximations for the high-frequency component are superposed with weights $1-\gamma$ and γ to generate a more reliable approximation, which is then added to the low-frequency component to yield $\mathbf{D}^k_{\text{out}}$. Also, in Fig. S-7 (c), four maps $\mathbf{D}^k_{\text{left}}$, $\mathbf{D}^k_{\text{right}}$, $\mathbf{D}^k_{\text{top}}$ and $\mathbf{D}^k_{\text{down}}$ are generated. For any pixel, if we know its depth and its gradients in the row and column directions, we can estimate the depths of the four adjacent pixels. Thus, from $\mathbf{D}^k_{\text{in}}$ and $\mathbf{R}^k_{\text{r}}$, we can reconstruct $\mathbf{D}^k_{\text{left}}$ and $\mathbf{D}^k_{\text{right}}$, which are both approximations of $\log \mathbf{D}^k$. Similarly, from $\mathbf{D}^k_{\text{in}}$ and $\mathbf{R}^k_{\text{c}}$, we obtain $\mathbf{D}^k_{\text{top}}$ and $\mathbf{D}^k_{\text{down}}$. Finally, by combining $\mathbf{D}^k_{\text{in}}$ with the four approximations, we obtain the output $\mathbf{D}^k_{\text{out}}$. In Fig. S-7 (b) and (c), a parameter γ is used to combine depth maps. It is simply set to minimize $\|\mathbf{D}^k_{\text{out}} - \log \mathbf{D}^k\|_1$ using training data.

S-4 Decoder Structure for RDE

In Fig. 1 in the paper, the decoders expand the encoder output to the resolution of $\mathbf{D}^0$ for ODE and to those of $(\mathbf{R}^k_{\text{M17}}, \mathbf{R}^k_{\text{M5}}, \mathbf{R}^k_{\text{r}}, \mathbf{R}^k_{\text{c}})$, $2 \le k \le 5$, for RDE. Fig. S-8 (a) shows the structure of each decoder, which includes 5 up-sampling blocks. For ODE, we set the hyper parameter $n_{\text{d}} = 1024$. On the other hand, we set $n_{\text{d}} = 256$ for RDE, which have 16 decoders, to keep the overall number of network parameters similar to those of ODE. Fig. S-8 (b) shows the detailed structure of each up-sample block. We adopt ReLU for the activation. In RDE, relative maps at scales $2 \sim 5$ are estimated. Thus, unlike ODE for $\mathbf{D}^0$, it is not necessary to perform up-sampling 5 times. For the decoders for maps at scale k, we eliminate the bilinear interpolation U from the last k up-sample blocks in Fig. S-8 (a).

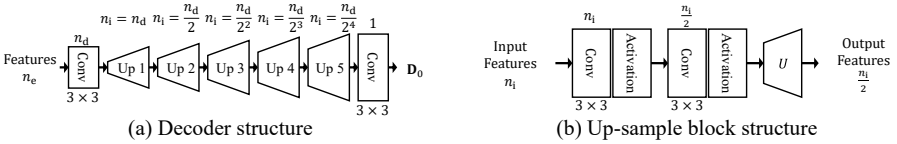


Fig. S-8. The structure of a decoder: For each convolution block, the number of output channels is specified at the top. Also, n_{e} is the number of encoder output channels, n_{i} is the number of input channels to an up-sample block, and n_{d} is a hyper-parameter to control the number of network parameters in the decoder.

Table S-4. Evaluation metrics for relative depth estimation: l_{ij} has a value of 1 or -1 depending on the relative relation between d_i and d_j.

Metrics for relative depth perception	
WHDR	Weighted human disagreement rate [52] for DIW [5], $\frac{\sum_{ij} w_{ij} \mathbf{1}(l_{ij} \neq \hat{l}_{ij})}{\sum_{ij} w_{ij}}$
WKDR	Weighted Kinect disagreement rate [52] for NYUv2 [41]
$\text{WKDR}^{=}$	WKDR for equivalent pairs of depths
$\text{WKDR}^{\neq}$	WKDR for inequivalent pairs of depths

S-5 Experiments for RDE

Fig. S-9 shows examples of estimated depth maps. In each example, a depth map from ODE and 16 relative depth maps from RDE are provided. All these 17 estimated depth maps are combined into the final result (ODE + RDE). The depth ratios between ODE + RDE and ODE are shown in the 'Refined Area' map. Note that ODE + RDE tends to refine depths near object boundaries.

Table S-4 lists the evaluation metrics for relative depth estimation. We follow the evaluation protocol of [45] for relative depth perception. Table S-5 compares relative depth estimation results. RDE means that only relative depth maps are fused to yield the results, while ODE + RDE fuses both ordinary and relative maps. Note that WKDR, $\text{WKDR}^{=}$ and $\text{WKDR}^{\neq}$ are measured on NYUv2, while WHDR is on DIW. DIW does not provide the ground-truth for dense depths. Thus, for DIW, ODE cannot be trained, and only the performance of RDE is reported. In terms of the three WKDR metrics, the proposed algorithm outperforms the conventional ones with large margins. Although RDE does not use the training data of DIW, it also yields competitive WHDR result.

Table S-6 and Table S-7 compares ODE and ODE + RDE performance on NYUv2 [41] and Make3D [39], respectively. We see that RDE improves the depth estimation results in terms of all metrics.

Table S-5. Relative depth perception results. WHDR is a metric for the DIW dataset [5], while the others are for NYUv2 [41].

	The lower, the better			
	WHDR	WKDR	$\text{WKDR}^{=}$	$\text{WKDR}^{\neq}$
Zoran *et al.* [52]	-	43.5%	44.2%	41.2%
Chen *et al.* [5]	14.39%	28.3%	30.6%	28.6%
Xian *et al.* [45]	**11.37%**	29.1%	29.5%	29.7%
RDE	14.67%	15.5%	16.1%	17.0%
ODE + RDE	-	**13.2%**	**14.0%**	**14.4%**

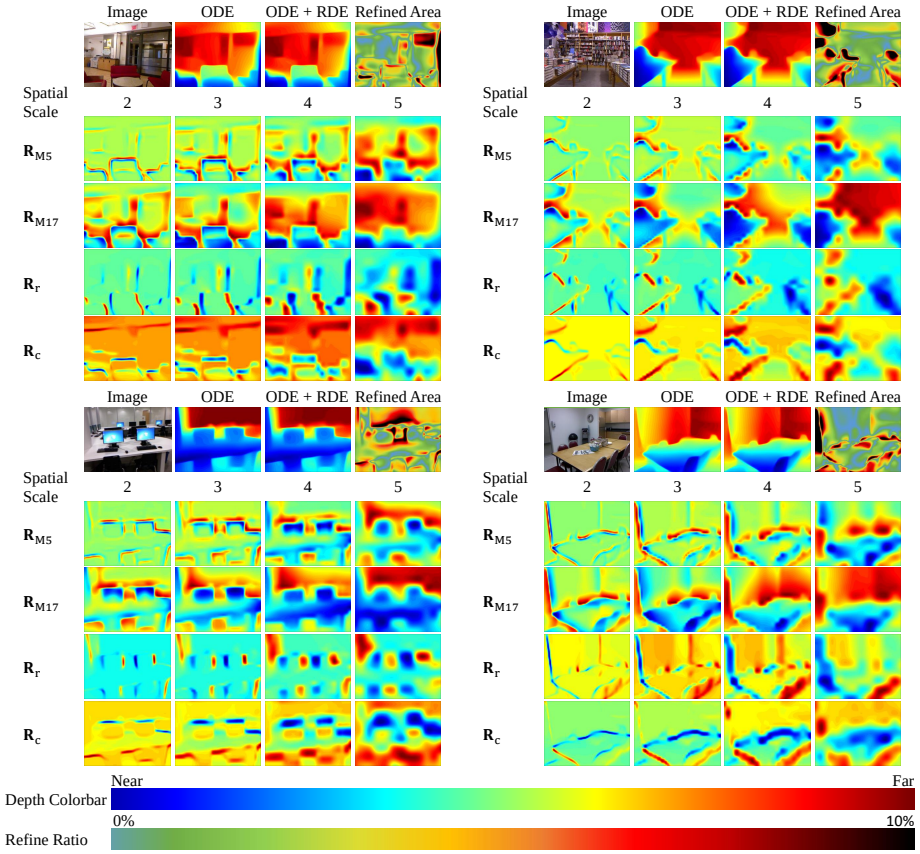


Fig. S-9. Examples of estimated ordinary and relative depth maps. ODE + RDE combines all these depth maps. The refined parts using the relative depth maps are depicted in the 'Refined Area' maps.

Table S-6. Effectiveness of RDE on NYUv2 [41].

	The higher, the better			The lower, the better					
	δ_1	δ_2	δ_3	RMSE_{lin}	ARD	log10	RMSE_{log}	RMSE_{si}	SRD
ODE	87.0%	97.4%	99.3%	0.430	0.119	0.050	0.151	0.123	0.078
ODE + RDE	87.4%	97.7%	99.5%	0.418	0.117	0.049	0.146	0.116	0.074

Table S-7. Effectiveness of RDE on Make3D [39].

	Evaluated in 0-70m depth range		
	RMSE_{lin}	ARD	log10
ODE	5.87	0.231	0.082
ODE + RDE	4.87	0.206	0.075