Input-level Inductive Biases for 3D Reconstruction: Supplementary material

Wang Yifan[1]* Carl Doersch[2] Relja Arandjelović[2] João Carreira[2] Andrew Zisserman[2,3]

[1]ETH Zurich [2]DeepMind [3]VGG, Department of Engineering Science, University of Oxford

A. Generalization on Out-Of-Domain Test Data

Our method generalizes to unseen datasets of a comparable domain. However, when testing on a significantly different domain, *e.g.* trained on indoor scenes and testing on outdoor scenes, our framework will see a performance drop. Table 1 shows the performance of a model trained on ScanNet and evaluated on RGBD and SUN3D datasets. SUN3D is similar to ScanNet, our method (trained on ScanNet only) performs reasonably well and on par with methods trained on SUN3D. RGBD datasets contain many warehouse scenes where the depth range is significantly different from the one seen during training. Our model shows overfitting in this case. This is because unlike conventional plane-sweep methods, where the network essentially computes a cost-volume from RGB features that are explicitly aligned, our method has to learn the alignment itself. How to reduce such a gap in case of domain shift is a research direction for future work. Besides introducing more augmentation techniques, we think fine-tuning and scale-normalization are some promising directions to pursue.

method	train	test	abs.rel $\downarrow$	rmse $\downarrow$	$\delta < 1.25 \uparrow$
IIB (ours)	ScanNet	SUN3D	0.1291	0.3699	0.8298
IIB (ours)	SUN3D	SUN3D	0.0985	0.2934	0.9018
NAS [27]	SUN3D	SUN3D	0.1271	0.3775	0.8292
IIB (ours)	ScanNet	RGBD	0.2572	1.3102	0.5101
IIB (ours)	RGBD	RGBD	0.0951	0.5498	0.9065
NAS [27]	RGBD	RGBD	0.1314	0.6190	0.8565

Table 1. Generalisation performance.

B. Camera localization: implementation details

In this section, we give implementation details for Section 4.3. Given a pair of images and ground truth depth maps, the goal is to infer the relative position and orientation of the two cameras using the network. We use a perceiver with $\boldsymbol{c}_{j,i}, \boldsymbol{r}_{j,i}$ for the camera embedding and $\boldsymbol{n}_{j,i}$ for the epipolar constraint.

*Work done during internship at DeepMind.

Recall that cameras are parameterized with intrinsics $\boldsymbol{K}$, and extrinsics $\boldsymbol{R}$ and $\boldsymbol{t}$. We assume known $\boldsymbol{K}$'s, and without loss of generality that the first camera is fixed as $\boldsymbol{R}_1 = \boldsymbol{I}$ and $\boldsymbol{t}_1 = \boldsymbol{0}$. Thus, the extrinsics of the second camera $\boldsymbol{R}_2$ and $\boldsymbol{t}_2$ are the optimization variables, and these are parameterized as $\boldsymbol{t}_2 \in \mathbb{R}^3$ and $\hat{\boldsymbol{R}}_2 \in \mathbb{R}^{3\times3}$. To make sure that the extrinsic matrix is a rigid transformation, we compute the final rotation matrix as $\boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{V}$, where $\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V} = \mathrm{SVD}(\hat{\boldsymbol{R}}_2)$, SVD is singular value decomposition, $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal, and $\boldsymbol{S}$ is diagonal. Therefore we are overparameterizing $\boldsymbol{R}_2$ with 9 parameters but only 3 degrees of freedom, but empirically we find this gives good results.

Our optimization objective is L1LOG loss on both images, plus regularization terms to encourage both the translation and rotation to be small. Without regularization, we find that the optimization can get stuck in local minima with very large rotation and translation, which we don't expect will make sense to the network. Specifically, we let $L_{rot}(\boldsymbol{R}) = \arccos((\mathrm{trace}(\boldsymbol{R}) - 1)/2)$ in radians, and $L_{trans}(\boldsymbol{t}) = \|\boldsymbol{t}\|_2$. Then the final loss can be written as:

$$L(\boldsymbol{R}_2, \boldsymbol{t}_2) = \mathrm{L1L{\small OG}}(\boldsymbol{R}_2, \boldsymbol{t}_2) + \lambda_{rot} L_{rot}(\boldsymbol{R}_2) + \lambda_{trans} L_{trans}(\boldsymbol{t}_2)$$

We set $\lambda_{rot} = 1$ and $\lambda_{trans} = 20$. Note that during optimization, the translation coordinates use the default Sun3D scaling, which is $100\times$ smaller than what we report in our result table.

We use the ADAM optimizer with a cosine learning rate schedule for 200 steps and an initial learning rate of $5e{-}3$. We initialize $\boldsymbol{t}_2 = \epsilon_{trans}$ where $\epsilon_{trans} \in \mathbb{R}^3$ and $\hat{\boldsymbol{R}}_2 = \boldsymbol{I} + \epsilon_{rot}$ where $\epsilon_{trans} \in \mathbb{R}^3$. Each element of both ϵ_{rot} and ϵ_{trans} is distributed as $\mathcal{N}(0, 0.01)$. We find that the network occasionally gets stuck in local optima, so we run with 5 different random initializations and take the solution with the best total loss L.

C. Qualitative Results

Additional qualitative results are shown in Fig. 1.

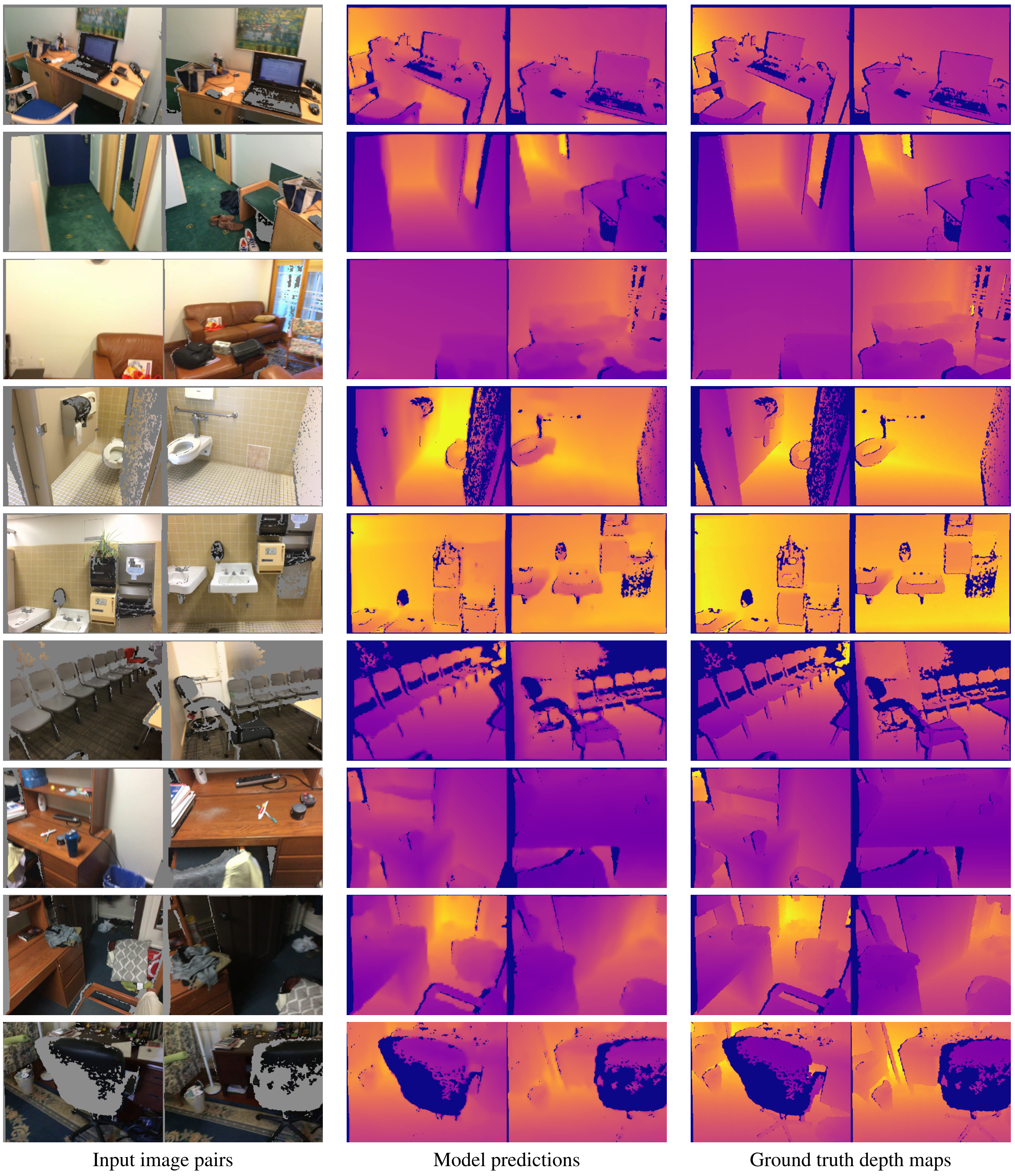

Figure 1. Additional examples of estimated depths using our best model on image pairs from ScanNet. Holes in the ground truth depth maps are masked out (shown in black).