Supplement File for "Trap Attention: Monocular Depth Estimation with Manual Traps"

Chao Ning
Northwestern Polytechnical University
Xi'an 710072, China
lagarto@mail.nwpu.edu.cn

Hongping Gan*
Northwestern Polytechnical University
Xi'an 710072, China
ganhongping@nwpu.edu.cn

1. Details of trap depth estimation network

Our depth estimation network consists of an encoder and a decoder with a block selection (BS) unit in between. In order to adapt to different application scenarios, we build our trap depth estimation networks with 3 sizes (Trap-S, Trap-M, Trap-L) by using different encoders (Trap-S uses an encoder of 12 blocks, and Trap-M/Trap-L uses an encoder of 24 blocks), and decoders of different sizes. The decoder size depends on the projected dimension of C, and a ratio of "$\frac{\text{hidden dimension}}{\text{output dimension}}$" in convolution based MLP. For the BS unit, the input is a group of feature maps from consecutive candidate encoder blocks, and the corresponding output is the maximum pixels across each position among this group. If the size of BS output is inconsistent with that of decoder, a convolution layer or a transposed convolution layer, which has the kernel size of 2×2 and the stride of 2, is used to downsample or upsample the BS output, respectively.

The detailed settings of Trap-S, Trap-M, and Trap-L are summarized in Table 1.

2. Additional visualization results

2.1. Additional visualization results on depth map

To make more detailed comparisons, we display the depth estimation results on NYU [10], and SUN RGB-D [11].

As depicted in Figure 2, our Trap-S and Trap-L models excel in predicting fine-grained depth information compared to other existing methods. Note that the sharp details can be only obtained by the network design. For example, although the depth estimation scores of our Trap-S is lower than those of NeW CRFs [13], it can be obviously observed that the predictions of Trap-S still have sharper details.

We also visualize the cross-dataset test results on SUN RGB-D dataset. According to Figure 3, the visualization results show that our method achieve the best robustness.

*Corresponding author

3. Results on the online KITTI evaluation benchmark

We train our proposed depth estimation model on the official split proposed by Geiger et al. [4] (42949 training samples, 1000 validation samples, and 500 testing images) to evaluate our method on the online KITTI evaluation benchmark.

Table 2 shows the performances of our proposed model and other depth estimation networks on the online KITTI evaluation server. According to Table 2, we can see that our Trap-L outperforms other depth estimation methods across all 4 metrics. For example, compared to previous state-of-the-art method, NeW CRFs [13], Trap-L can reduce "SILog", "sqErrorRel", "absErrorRel" and "iRMSE" by about 2.3%, 9.2%, 5.4% and 5.3%, respectively.

The visualization results of the predicted depth maps are generated by the online server are shown in Figure 1. Compared to the NeW CRFs method, our method can predict sharper and smoother depth for various objects, e.g., the trees, roofs and cars

4. Trap attention visualization

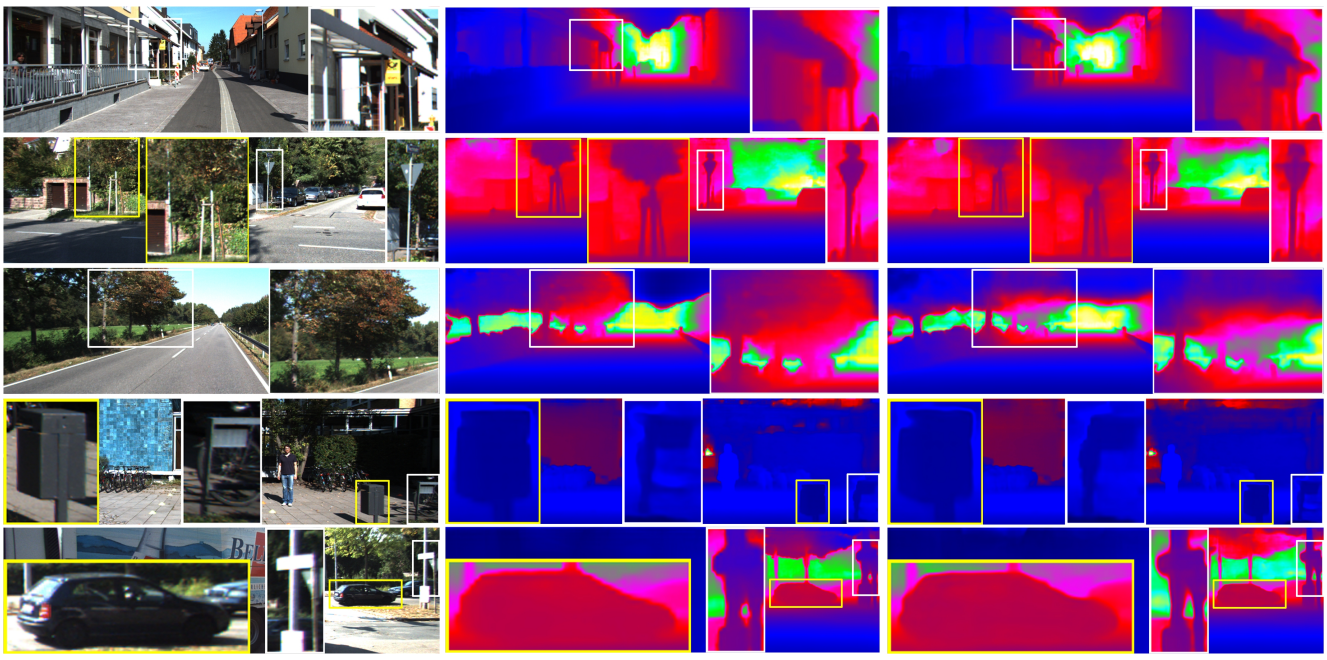
To verify and understand the effectiveness of the proposed trap attention, we visualize the attention maps and feature maps in Figure 4. The trap attention units focus on various regions for low-resolution feature map, while the trap attention units retain the plane information, such as texture, edges etc., for high-resolution feature map. For the last mixed feature map of our model, the trap attention units can capture distance information. As shown in the third last row of Figure 4, the right half feature maps for each image retain the features of distant objects.

5. Trap block vs. Transformer block

To further compare trap attention and MHA, we use a single trap block and a single Transformer block to decode the encoded features from XCiT-S [2], respectively. The

Table 1. The detailed setting of our Trap-S, Trap-M, and Trap-L models. The column of "α" is the ratio of "$\frac{\text{hidden dimension}}{\text{output dimension}}$" in our convolution based MLP. "Block section" indicates the candidate encoder blocks in 5 BS units, and "01-04" of first row denotes that the first 1 to first 4 blocks are the candidate blocks in the BS unit of $\frac{1}{2^2}$ input resolution stage. "R." denotes the resolution of input image. C is the projected channel dimension of decoder.

Model	Encoder	# Params	C	α	Block section ([$\frac{1}{2^2}$ R., $\frac{1}{4^2}$ R., $\frac{1}{8^2}$ R., $\frac{1}{16^2}$ R., $\frac{1}{32^2}$ R.])
Trap-S	XCiT-S12 [2]	28.3M	64	4	[01-04, 03-06, 05-08, 07-10, 09-12]
Trap-M	XCiT-M24 [2]	94.2M	128	4	[01-08, 05-12, 09-16, 13-20, 17-24]
Trap-L	Swin-L [8]	222.7M	192	4	[01-02, 03-04, 05-16, 11-22, 22-24]



(a) Input (b) NeW CRFs (c) Trap-L (ours)

Figure 1. Qualitative comparison with the state-of-the-art method on the online KITTI test dataset. Compared to NeW CRFs, our Trap-L can predict the depths with sharper details for various targets.

dataset used is the NYU dataset. For a fair comparison, we configure the two decoders with similar computational complexities. The trap decoder has 48.8 GFLOPs, while the Transformer decoder has 54.6 GFLOPs. Due to the quadratic complexity, MHA can not perform on large resolutions as it requires excessive GPU memory for computations, even though the computational complexity is similar to the trap block. Therefore, we use a lower resolution for the Transformer decoder. The results are shown in Table 3 and Figure 5. The trap block outperforms the Transformer block across seven metrics. Based on the visualization results, the trap block shows better performance in predicting the depth of small objects.

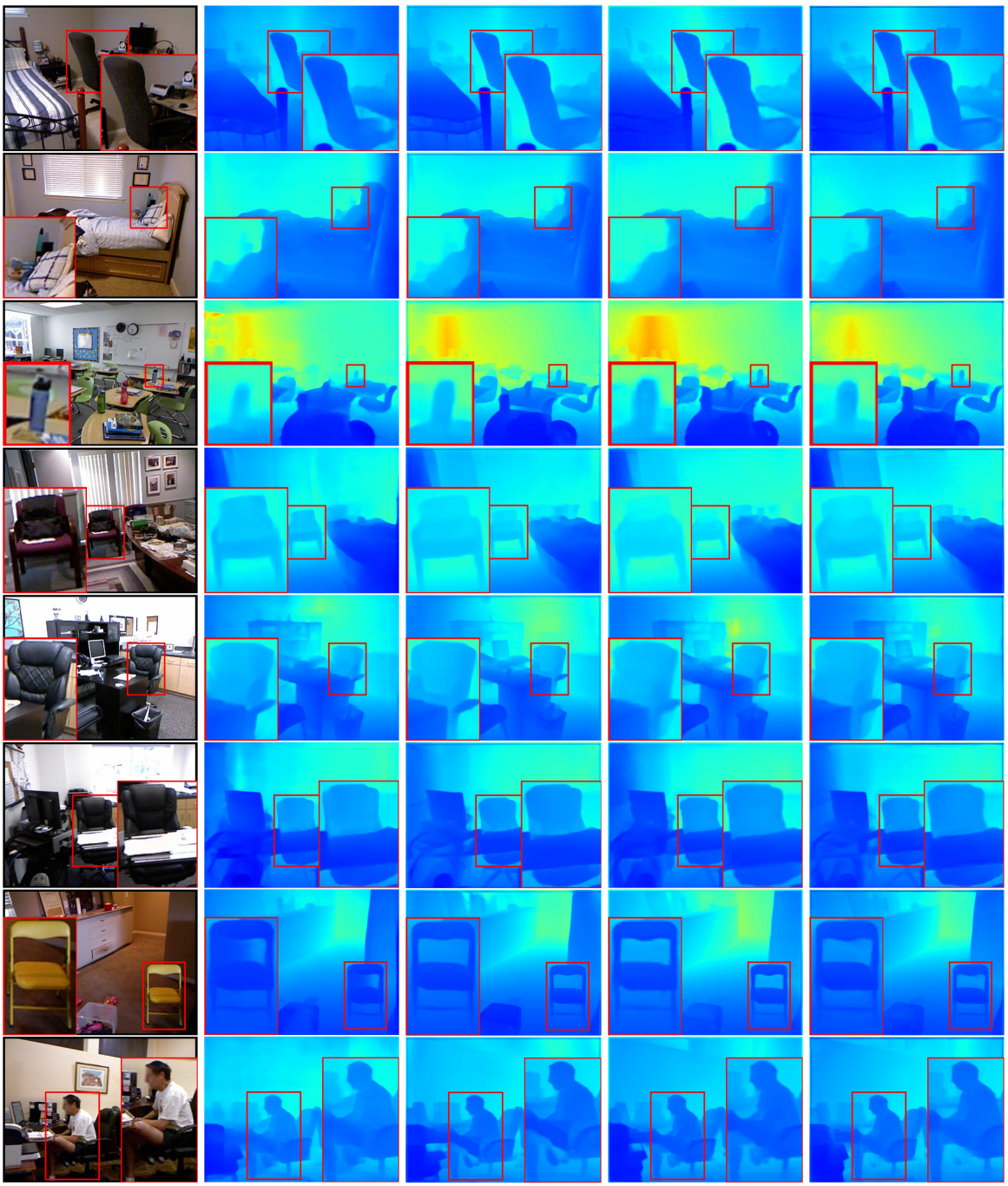
6. Model definition in Pytorch

The pseudo-pytorch-code associated with our trap interpolation and trap block are provided in Algorithm 1 and Algorithm 2, respectively.

References

[1] Shubhra Aich, Jean Marie Uwabeza Vianney, Md Amirul Islam, and Mannat Kaur Bingbing Liu. Bidirectional attention network for monocular depth estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11746–11752. IEEE, 2021.

[2] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan

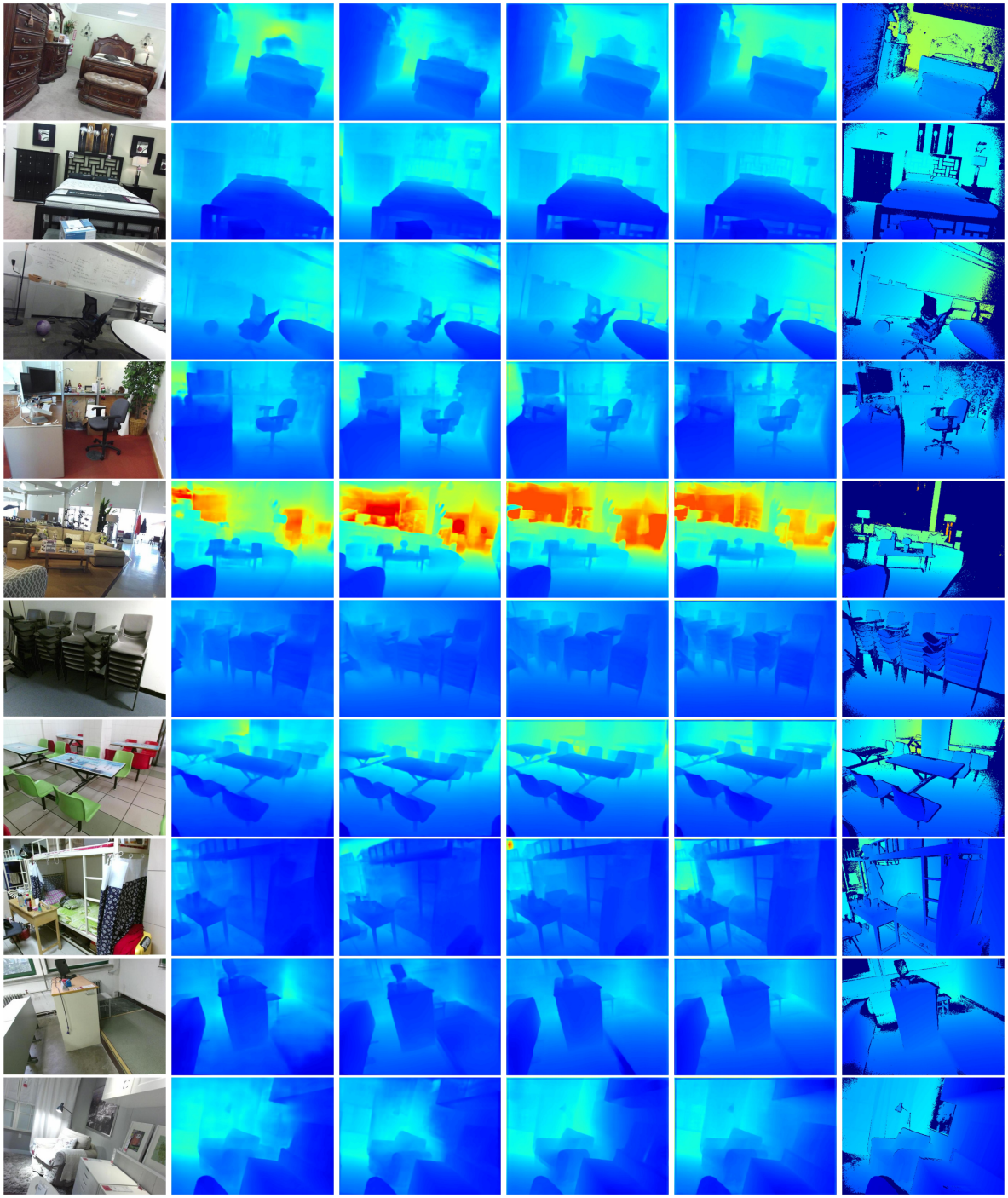


(a) Input (b) BTS (c) Trap-S (ours) (d) NeW CRFs (e) Trap-L (ours)

Figure 2. Detailed comparison with the state-of-the-art methods on the NYU dataset. It can be clearly observed that the predictions of our Trap-S and Trap-L have sharper details. Please zoom in for more details.

Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in Neural Information Processing Systems (NIPS)*, 34:20014–20027, 2021.

[3] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2002–2011, 2018.



(a) Input (b) BTS (c) AdaBin (d) Trap-S (ours) (e) Trap-L (ours) (f) GT

Figure 3. Comparison with the state-of-the-art methods on the SUN RGB-D dataset. Compared with BTS and AdaBin, our proposed Trap-S and Trap-L have more accurate prediction.

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[5] Vitor Guizilini, Rares Ambrus, Wolfram Burgard, and Adrien Gaidon. Sparse auxiliary networks for unified monocular depth prediction and completion. In *Proceedings of the IEEE Conference on Computer Vi-*

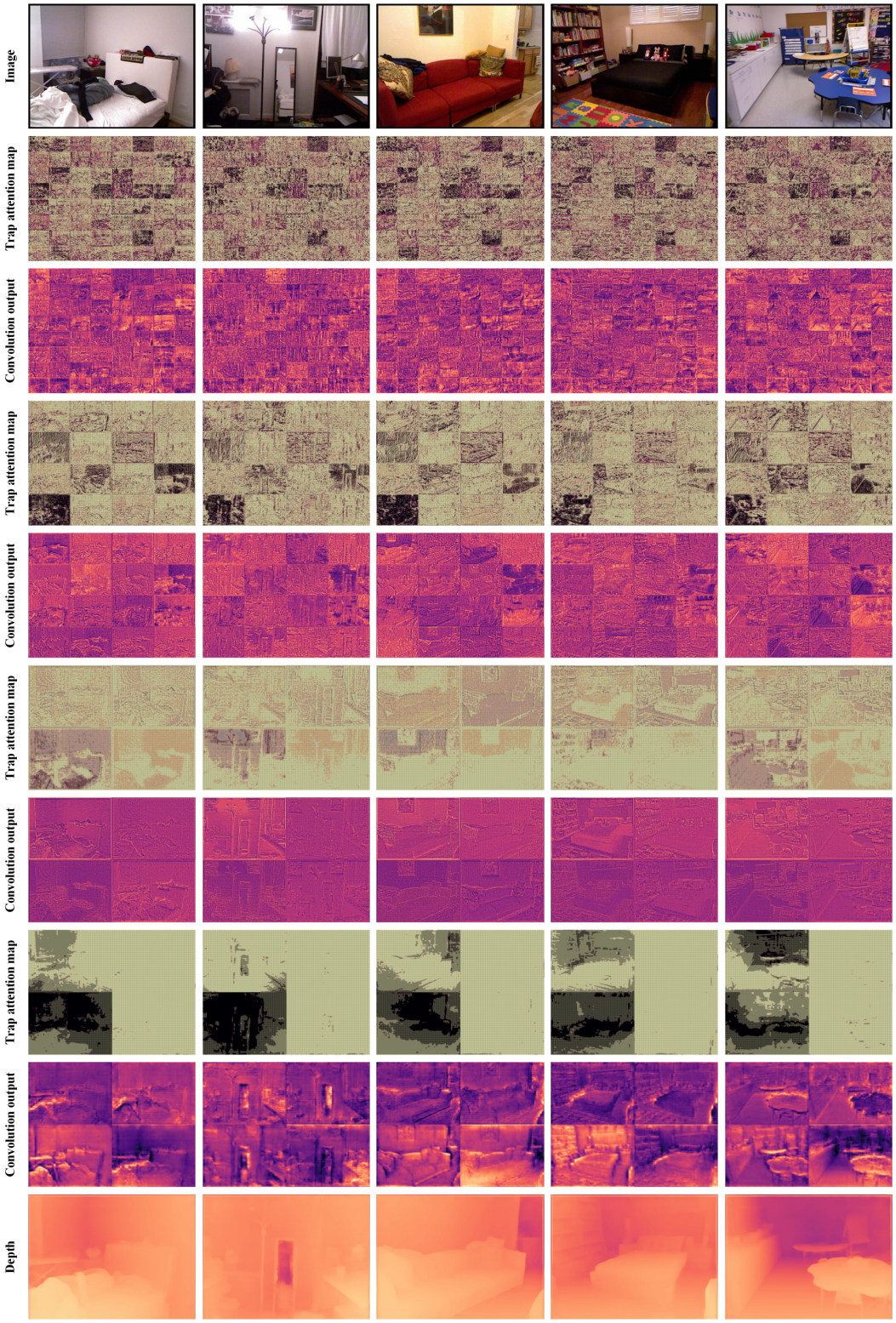


Figure 4. The attention maps and the counterpart convolution outputs of trap attention units. From top to bottom are the coarse depths to fine depths.

(a) Input (b) Transformer (c) Trap (ours) (d) GT

Figure 5. Qualitative comparison between Transformer block and Trap block on NYU dataset.

Algorithm 1: Pseudocode of Trap Interpolation in PyTorch-like style

```
def trapped_inter(x):
        B, C, H, W = x.shape
        mask1 = torch.round(torch.abs(torch.sin(x)))
        mask2 = torch.round(torch.abs(torch.cos(x)))
        mask3 = torch.round(torch.abs(2*torch.sin(x)*torch.cos(x)))
        mask4 = torch.round(torch.sin(x) ** 2)

        x1 = mask1 * x
        x2 = mask2 * x
        x3 = mask3 * x
        x4 = mask4 * x
        x = torch.cat([x1, x3, x2, x4], dim=1)

        x = x.view(B, 2, 2*C, H, W)
        x = x.permute(0, 2, 3, 1, 4).flatten(2).contiguous()
        x = x.view(B, 2*C, H * 2, W)
        x = x.view(B, 2, C, H * 2, W)
        x = x.permute(0, 2, 3, 4, 1).flatten(-1).contiguous()
        x = x.view(B, C, H * 2, W * 2)

        return x
```

Table 2. Comparison of performances on the online KITTI evaluation server.

Method	lower is better			
	SILog	sqErrorRel	absErrorRel	iRMSE
Yin et al. [12]	12.65	2.46	10.15	13.02
P3Depth [9]	12.82	2.53	9.92	13.71
DRON [3]	11.77	2.23	8.78	12.98
BTS [6]	11.67	2.21	9.04	12.23
BA-Full [1]	11.61	2.29	9.38	12.23
PackNet-SAN [5]	11.54	2.35	9.12	12.38
PWA [7]	11.45	2.30	9.05	12.32
NeWCRFs [13]	10.39	1.83	8.37	11.03
Trap-L (ours)	**10.15**	**1.66**	**7.92**	**10.45**

sion and Pattern Recognition (CVPR), pages 11078–11088, 2021.

[6] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[7] Sihaeng Lee, Janghyeon Lee, Byungju Kim, Eojindl Yi, and Junmo Kim. Patch-wise attention network for monocular depth estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1873–1881, 2021.

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[9] Vaishakh Patil, Christos Sakaridis, Alex Liniger, and Luc Van Gool. P3depth: Monocular depth estimation with a piecewise planarity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, 2012.

[11] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[12] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5684–5693, 2019.

[13] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Newcrfs: Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Algorithm 2: Pseudocode of Trap Block in PyTorch-like style

```
class TrapBlock(nn.Module):
    def __init__(
            self, in_features, hidden_features=None, out_features=None, act_layer=nn.GELU,
        norm_layer=LayerNorm2d, drop=0.1, ls_init_value=1., trap=True, drop_path=0.
    ):
        super().__init__()
        out_features = out_features or in_features
        hidden_features = hidden_features or in_features

        # 7x7 Conv
        self.dw_conv = nn.Conv2d(in_features, in_features,
                                 kernel_size=7, padding=3, groups=in_features)

        self.trap = trap
        if self.trap:
            self.downsample = nn.PixelUnshuffle(2)
            self.attn_conv = nn.Conv2d(in_features*4, in_features, kernel_size=3,
                         padding=1, groups=in_features)
            self.norm1 = norm_layer(in_features*4) if norm_layer else nn.Identity()
            self.gamma1 = nn.Parameter(ls_init_value * torch.ones(in_features))
                        if ls_init_value > 0 else None


        # ConvMLP
        self.fc1 = nn.Conv2d(in_features, hidden_features, kernel_size=1, bias=True)
        self.act = act_layer()
        self.fc2 = nn.Conv2d(hidden_features, out_features, kernel_size=1, bias=True)

        # LayerNorm
        self.norm1 = norm_layer(in_features) if norm_layer else nn.Identity()
        self.norm2 = norm_layer(in_features) if norm_layer else nn.Identity()

                self.gamma2 = nn.Parameter(ls_init_value * torch.ones(out_features))
                if ls_init_value > 0 else None

        self.shortcut = out_features == in_features
        self.drop = nn.Identity()
        self.drop_path = DropPath(drop_path)

    def forward(self, x):
        x = self.drop_path(self.dw_conv(x)) + x
        if self.trap:
            shortcut1 = x
            x = trapped_inter(self.downsample(x))
            x = self.norm1(x)
            x = self.attn_conv(x)
            x = x.mul(self.gamma1.reshape(1, -1, 1, 1))
            x = self.drop_path(x) + shortcut1

        if self.shortcut:
            shortcut2 = x
        x = self.norm2(x)
        x = self.fc1(x)
        x = self.act(x)
        x = self.fc2(x)
        if self.gamma2 is not None:
            x = x.mul(self.gamma2.reshape(1, -1, 1, 1))

        if self.shortcut:
            x = self.drop_path(x) + shortcut2

        return x
```

Table 3. Comparison of performances on the NYU dataset.

Method	higher is better			lower is better				
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	RMSE log	log10
Transformer	0.886	0.980	**0.995**	0.111	0.065	0.382	0.140	0.047
Trap	**0.894**	**0.982**	**0.995**	**0.106**	**0.059**	**0.379**	**0.136**	**0.046**