Supplementary Materials of PPEA-Depth

Anonymous submission

1 Ablations for Adapter Design

We conduct experiments to assess various adapter designs using RepLKNet-B. All the reported results are from the student network.

1.1 Encoder Adapter

We compare different encoder adapter designs for the domain adaptation stage on the KITTI dataset.

Type of Blocks to Attach Adapters Our method involves attaching encoder adapters to two types of blocks within the RepLKNet (Ding et al. 2022) backbone: RepLKBlock and ConvFFN. We conducted an ablation study by adding adapters to only one type of these blocks. The results are presented in Table 1. Evidently, the frozen encoder benefits more from adding adapters to RepLKBlock rather than ConvFFN. RepLKBlock, which leverages a large kernel size of up to 31x31, stands as the core innovation of RepLKNet. This outcome aligns with the intuition that adapters on more intricate blocks carry greater significance.

Receptive Field of Projectors Adapters project the input feature down to a lower dimension and then project it up. As depth estimation is a regression task, will encoder adapters benefit from a larger receptive field? We conducted experiments using different receptive fields for the adapters of RepLKBlock, which, as indicated by the results in Table 1, holds greater importance than ConvFFN. We compared two types of projectors: linear projection and convolution with a kernel size of 3, while maintaining a bottleneck ratio of 0.25. The results are presented in Table 2. From the results, it can be inferred that all metrics benefit from a larger receptive field in the down projector, while the enhancement resulting from a wider receptive field in the up projector is not as apparent.

BatchNorm and Skip Connections We experiment with different encoder adapter designs by varying the position of the BatchNorm module and skip connections (Figure 1), while keeping the inner adapter design the same (bottleneck ratio is 0.25, down projector is a 3x3 convolution and up projector is linear). The evaluation results of different designs are in Table 3.

The BatchNorm module before the pre-trained block is part of the original RepLKBlock and ConvFFN design (more details are shown in Figure 3). We design to let the adapter share the BatchNorm module with the original pre-trained block, and unfreeze the parameters of the BatchNorm during training. From the results in Table 3, it can be concluded that:

- The final depth estimation accuracy benefit from the BatchNorm block before the encoder adapter. Removing the prefix BatchNorm before encoder adapters results in an obvious performance loss (Design (b)).
- The BatchNorm module before the pre-trained block is indispensable (Design (c)). As the parameters in the pre-trained blocks are frozen, the frozen parameters are matched with the output of the prefix BatchNorm, removing the prefix BatchNorm for the pre-trained block leads to degraded results.
- Sharing the same prefix BatchNorm module between the pre-trained block and the encoder adapter (Design (d)) and training an extra BatchNorm for the encoder adapter (Design (a)) show no obvious difference in the final depth estimation errors and accuracy. The sharing strategy leads to a lower trainable parameter cost, so design (a) in Figure 1 is our choice.

1.2 Decoder Adapter

We compare different decoder adapter designs for the scene adaptation stage on the CityScapes dataset. We keep the encoder adapter design consistent (bottleneck ratio is 0.25, down projector is a 3x3 convolution, and up projector is linear) and use the same weights trained from the domain adaptation stage as initial model weights in the following experiments.

Input Scales The depth encoder generates feature maps at four levels (F^0, F^1, F^2, F^3), corresponding to $1/4$, $1/8$, $1/16$, and $1/32$ of the original image spatial shape. Upsampling feature maps at different levels to the $1/4$ scale, we concatenate and feed them to the decoder adapters. We compare different input feature scales for the decoder adapter, and the results are in Table 4, suggesting that using the concatenation of the shallowest and deepest features as input is the optimal choice, considering both parameter count and performance.

Adapter Position	Params (M)	*Errors*↓				*Accuracy*↑		
		AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
RepLKBlock	17.7	0.093	0.718	4.285	0.171	0.909	0.968	0.984
ConvFFN	3.78	0.100	0.769	4.442	0.175	0.900	0.966	0.984
All	21.9	0.090	0.666	4.175	0.168	0.912	0.969	0.984

Table 1: **Comparison of Attaching Adapters to Different Types of Blocks.** Attaching adapters to more sophisticated blocks in backbone leads to better results.

Down Projector	Up Projector	Params (M)	*Errors*↓				*Accuracy*↑		
			AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Linear	Linear	7.5	0.095	0.776	4.372	0.174	0.909	0.967	0.983
Conv	Linear	21.9	**0.090**	**0.666**	**4.175**	**0.168**	**0.912**	**0.969**	**0.984**
Conv	Conv	35.4	**0.090**	0.671	4.203	0.169	**0.912**	0.968	**0.984**

Table 2: **Influence of Encoder Adapter Receptive Field.** Domain adaptation stage benefits from employing encoder adapters with down projector of larger receptive fields, while the improvement brought by up projector of larger receptive fields is not obvious.

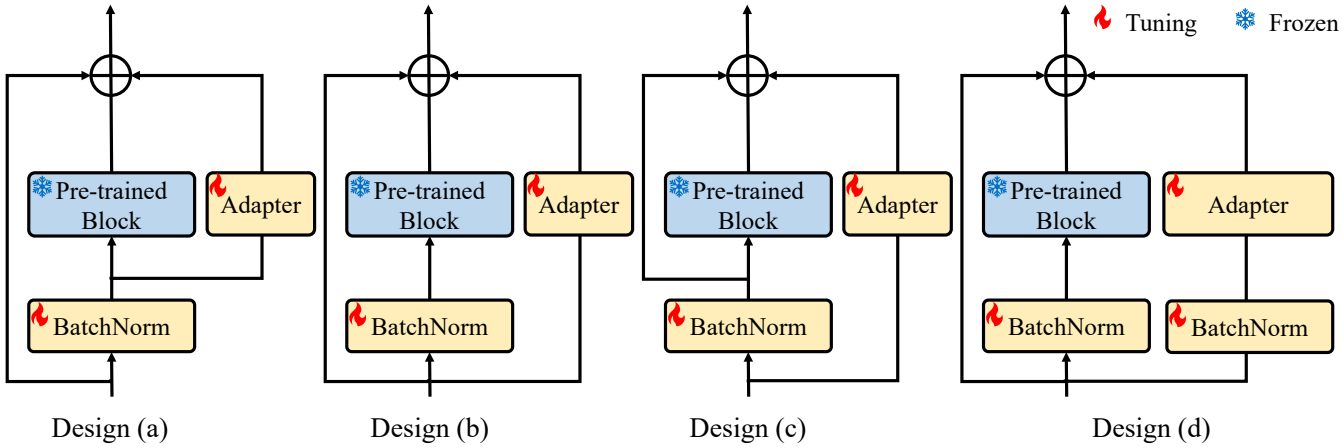


Figure 1: **Different Encoder Adapter Designs**. Design (a) is our final choice.

Design	*Errors*↓				*Accuracy*↑		
	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
a	**0.090**	**0.666**	**4.175**	**0.168**	**0.912**	**0.969**	**0.984**
b	0.095	0.769	4.407	0.175	0.905	0.965	0.983
c	0.103	0.761	4.499	0.181	0.890	0.962	0.983
d	0.091	0.680	4.219	0.170	0.911	0.968	0.984

Table 3: **Comparison of Different BatchNorm and Skip Connection Designs** in Encoder Adapters.

Input Scales	Params (M)	*Errors*↓				*Accuracy*↑		
		AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
3	0.149	0.103	0.977	5.693	0.157	0.895	0.974	0.991
0, 3	0.185	**0.100**	**0.976**	5.673	**0.152**	**0.904**	**0.977**	**0.992**
0, 1, 2, 3	0.486	0.101	0.982	**5.666**	**0.152**	**0.904**	**0.977**	**0.992**

Table 4: **Comparison of Different Input Feature Levels** to Decoder Adapter.

2 Inference Latency Introduced by Adapter

As adapters are additional modules added to the network, we investigate the inference latencies introduced by the encoder and decoder adapters. We measure the total time required for the model to predict depth maps for 1,525 images in the CityScapes test dataset and calculate the average inference time per image. Our encoder backbone is RepLKNet-B, and we use a batch size of 12. The results are presented in Table 5. Thanks to the parallel data stream design, the inference time remains largely consistent whether adapters are used or not.

Encoder Adapters	Decoder Adapters	Time (s)
		0.037
●		0.036
●	●	0.038

Table 5: **Average Inference Time Per Image** on CityScapes test set with and without adapters.

3 Supplementary Details

3.1 Overview of the Whole Framework

We follow the self-supervised depth estimation framework proposed by Watson et al. (2021), whose depth network consists of a teacher network and a student network to improve estimated depths and better handle dynamic objects, and this design is also adopted in Guizilini et al. (2022); Feng et al. (2022). The teacher depth network is trained jointly with the student, sharing the same pose predictions, and is discarded during evaluation. Both the teacher and student networks are based on the U-Net architecture. They share identical designs for the depth network decoder but differ in their encoders. The teacher's input is a single image frame (the frame at timestamp t, denoted as I_t). This input, I_t, undergoes processing through the depth encoder and subsequently the depth decoder, leading to pixel-wise depth estimation.

As shown in Figure 2, the input of the student network includes two frames, I_t and its previous frame I_{t-1}. The two frames are first separately extracted by the first stage of the encoder to generate first-level features F_t^0, F_{t-1}^0, which are both in the shape of $(B, C, H/4, W/4)$, where B is the batch size, C is the number of channels, H and W represent the spatial shape of I_t. Then a cost volume is built based on F_t^0, F_{t-1}^0, relative camera pose T (which is currently predicted by the pose network), and a set of depth values D_c.

Iterating over D_c, we project F_{t-1}^0 using T and the iterated depth value d_i to generate $F_{t-1\to t}^0$ according to the pixel correspondences mentioned in Equation (1) in the main paper. The L1 difference between $F_{t-1\to t}^0$ and F_t^0 is adopted to build the cost volume. The minimum and maximum depths (d_{min}, d_{max}) are initialized as 0.1m and 100m respectively, and are dynamically tuned in the learning process according to the strategy proposed in Watson et al. (2021). The depth set D_c is generated by uniformly sampling depth values in $[d_{min}, d_{max}]$ in logarithm space.

The generated cost volume C_t is in the shape of $(B, |D_c|, H/4, W/4)$, where $|D_c|$ represents the number of depths in set D_c. Then C_t is concatenated with F_t^0 at the second dimension, and the concatenated feature is compressed to the shape of $(B, C, H/4, W/4)$ by a 3x3 convolution called *reduce conv*. Then the output of the reduce conv is fed to the rest stages of the depth encoder and then to the depth decoder, and finally predicts a depth map D_t.

The cost volume design in the student network introduces I_{t-1} and an iteration over all possible depths in the encoder, and exploits the relationship between two consequent frames to improve depth estimation. However, such a design exaggerates the depth estimation error in the dynamic object areas and tends to predict depths of such areas as infinity (Watson et al. 2021).

3.2 Teacher-Student Distillation Scheme

To overcome the infinity-depth issue as mentioned above, a teacher-student training scheme is employed in our network as in previous works (Watson et al. 2021; Feng et al. 2022; Guizilini et al. 2022).

For the dynamic object areas, the predictions from the student network are unreliable. Such unreliable area M is computed by comparing the predicted depths from the teacher and student in a pixel-wise manner:

$$M = \max(\frac{D_s - D_t}{D_t}, \frac{D_t - D_s}{D_s}) > 1$$

During the training process, the teacher network is supervised by the image reprojection loss (as mentioned in Section 3.1 in the main paper). For the student network, the reliable area $(\neg M)$ is also supervised by the image reprojection loss, while the unreliable area (M) is instead supervised by a depth consistency loss to enforce a knowledge distillation from the teacher. The depth consistency loss is the L1 difference of the predicted depths between the teacher and the student. In the depth consistency loss, gradients to depths predicted by the teacher are blocked, ensuring the distillation is unidirectional, i.e. only from teacher to student.

Different from the previous works (Watson et al. 2021; Feng et al. 2022; Guizilini et al. 2022), we do not freeze the teacher depth network and pose network to fine-tune in the last five epochs, since we do not observe a significant performance improvement with such technique on PPEA-Depth.

3.3 Amount of Encoder Adapter Parameters

All parameter counts mentioned in the main paper are based on the teacher network. Here, we provide supplementary details regarding the tunable parameter count for encoder and decoder adapters in the student network. Throughout all experiments, the settings for student encoder adapters (down projector, up projector, and bottleneck ratio) remain the same as those of the teacher network. The student network has more tunable parameters than the teacher because apart from the encoder adapters and the BatchNorm module, the reduce conv module (as detailed in Section 3.1) also receives updates. Refer to Table 6 for specific parameter details.

Pre-trained Backbone	Adapter Type	Ratio	Tuning Params(M)	
			Teacher	Student
RepLKNet-B	Encoder	0.0625	8.15	8.41
	Encoder	0.25	21.9	22.2
	Decoder	0.25	0.185	0.185
RepLKNet-L	Encoder	0.0625	18.1	18.6
	Encoder	0.25	47.6	48.1
	Decoder	0.25	4.15	4.15

Table 6: **Details of Tunable Parameters** in Teacher and Student Depth Network Adapters.

3.4 RepLKNet

In Section 3.3 of the main paper, we briefly introduce the main pipeline of RepLKNet-31B, which we adopt as the encoder backbone of both the teacher and student depth network. Here we introduce the detailed structure of the two key modules to which we attach encoder adapters, RepLKBlock and ConvFFN. For more details please refer to Ding et al. (2022).

The detailed structures of RepLKBlock and ConvFFN are shown in Figure 3. Our proposed encoder adapters share the same prefix BatchNorm module with RepLKBlock and ConvFFN (as in Figure 1). Since we adopt a different batch size during training compared with the RepLKNet pre-training process, all parameters of BatchNorm modules are not frozen.

3.5 Training Details

PPEA-Depth is implemented with PyTorch (Paszke et al. (2017)) and is trained on GeForce RTX 3090 GPU. We use the Adam optimizer (Kingma and Ba (2014)), with $\beta_1 = 0.9, \beta_2 = 0.999$. At the domain adaptation stage (Stage 1), the batch size is set to 12. This stage trains for 25 epochs, and the initial learning rate is 1e-4. The learning rate decays to 1e-5 after 15 epochs and decays to 1e-6 after 20 epochs. During the scene adaptation stage (Stage 2), we utilize a batch size of 24 for training. This stage is conducted over 10 epochs, employing a learning rate of 1e-5 for CityScapes and 5e-5 for DDAD.

3.6 Evaluation Metrics Details

We evaluate our depth estimation results using the standard depth assessment metrics (Eigen and Fergus 2015), including Absolute Relative Error (*AbsRel*), Squared Relative Error (*SqRel*), Root Mean Squared Error (RMSE), Root Mean Squared Log Error ($\mathrm{RMSE}_{\log}$), δ_1, δ_2, and δ_3. The specific formulas to calculate these metrics are as follows:

$$
\begin{aligned}
AbsRel &= \frac{1}{n}\sum_i \frac{p_i - g_i}{g_i} \\
SqRel &= \frac{1}{n}\sum_i \frac{(p_i - g_i)^2}{g_i} \\
\mathrm{RMSE} &= \sqrt{\frac{1}{n}\sum_i (p_i - g_i)^2} \\
\mathrm{RMSE}_{\log} &= \sqrt{\frac{1}{n}\sum_i (\log p_i - \log g_i)^2}
\end{aligned}
\tag{1}
$$

and the values δ_1, δ_2, and δ_3 represent the percentages of pixels where the condition $\max\,(p/q,\ q/p) < 1.25, 1.25^2, 1.25^3$ is satisfied. Here, g denotes the ground truth depth, p stands for the predicted depth, and n represents the number of pixels.

4 Supplementary Qualitative Results

We provide more qualitative comparisons on CityScapes in the last two pages. The images are organized from left to right, showcasing the original image, the estimated depth obtained by full fine-tuning a U-Net from scratch, and the estimated depth produced by our PPEA-Depth approach. We also provide a qualitative video demo (*demo.mp4* in the .zip file). Please check it out for a more dynamic representation of our approach's performance.

References

Ding, X.; Zhang, X.; Zhou, Y.; Han, J.; Ding, G.; and Sun, J. 2022. Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs. *arXiv preprint arXiv:2203.06717*.

Eigen, D.; and Fergus, R. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, 2650–2658.

Feng, Z.; Yang, L.; Jing, L.; Wang, H.; Tian, Y.; and Li, B. 2022. Disentangling Object Motion and Occlusion for Unsupervised Multi-frame Monocular Depth. *arXiv preprint arXiv:2203.15174*.

Guizilini, V.; Ambruș, R.; Chen, D.; Zakharov, S.; and Gaidon, A. 2022. Multi-Frame Self-Supervised Depth with Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 160–170.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Watson, J.; Aodha, O. M.; Prisacariu, V.; Brostow, G.; and Firman, M. 2021. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *Computer Vision and Pattern Recognition (CVPR)*.

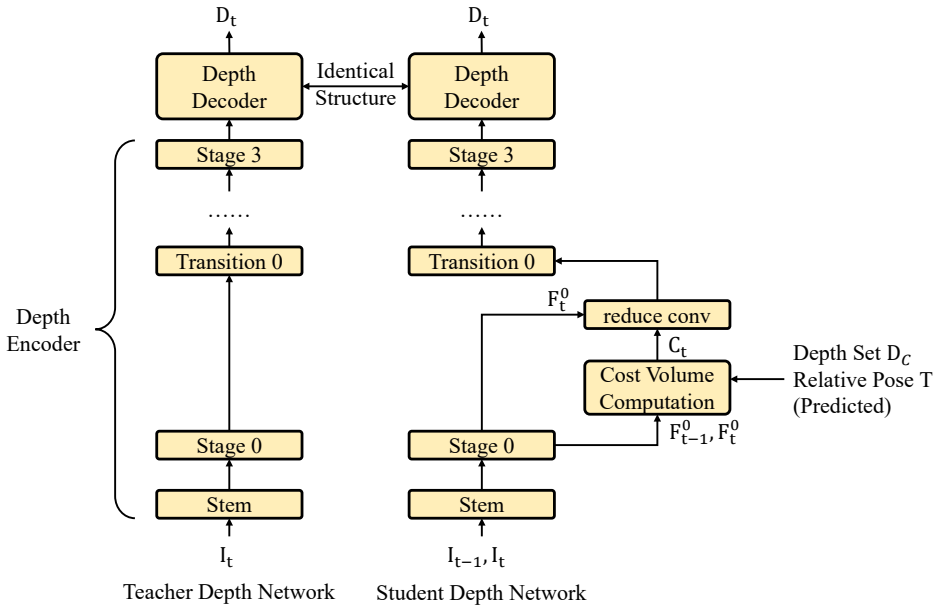


Figure 2: **Detailed Structure of Teacher and Student Depth Network.**

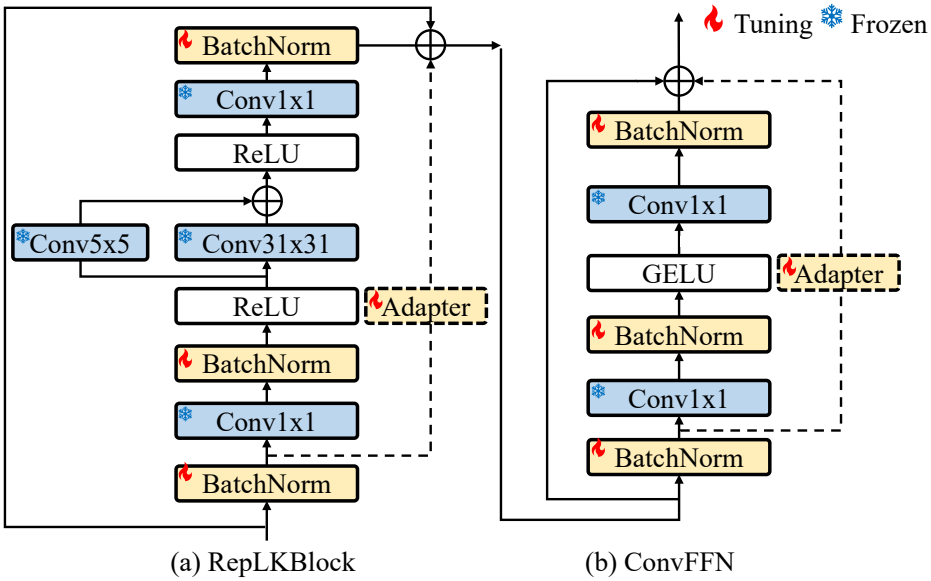
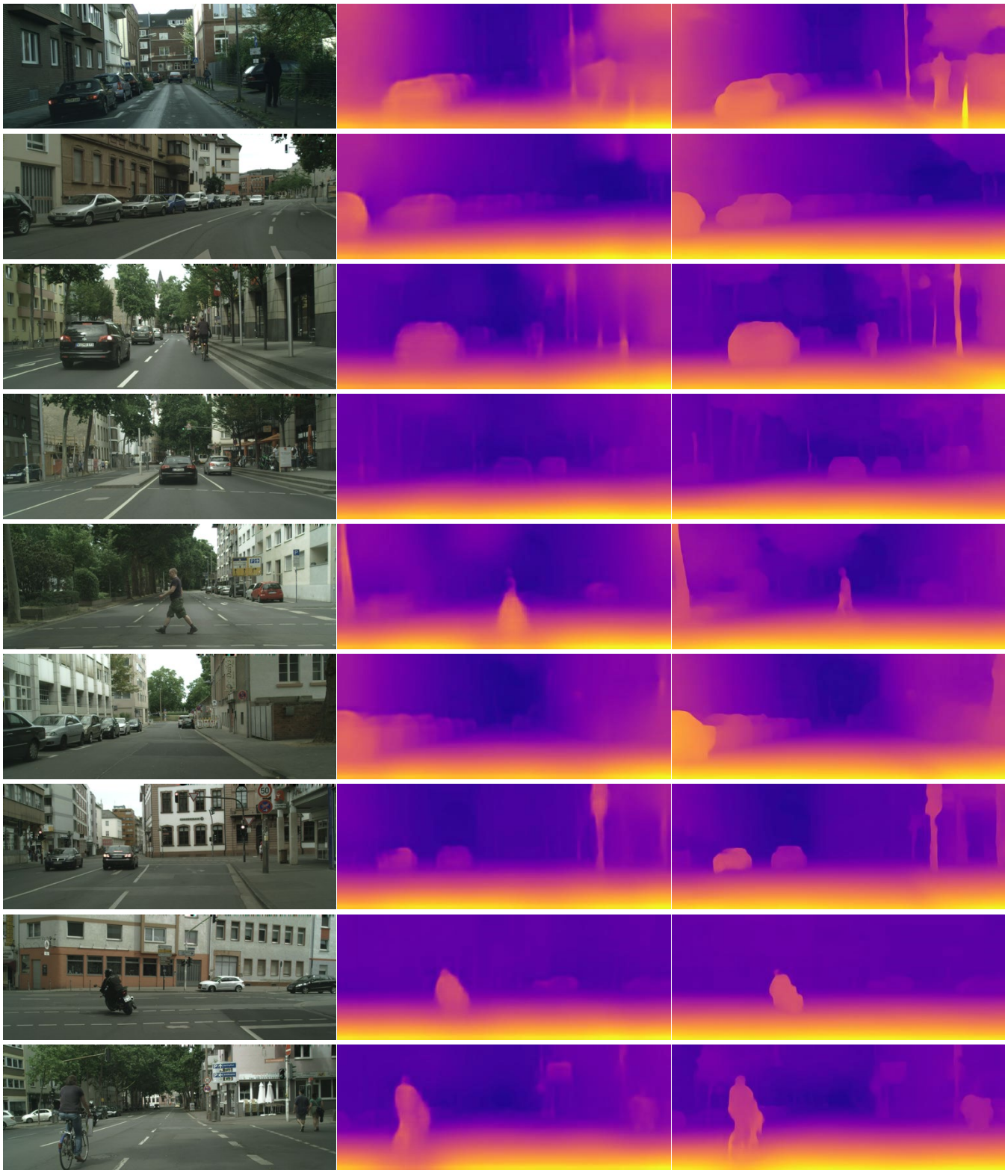


Figure 3: **Detailed Structure of RepLKBlock and ConvFFN**. The dashed line depicts the position of encoder adapters in PPEA-Depth.



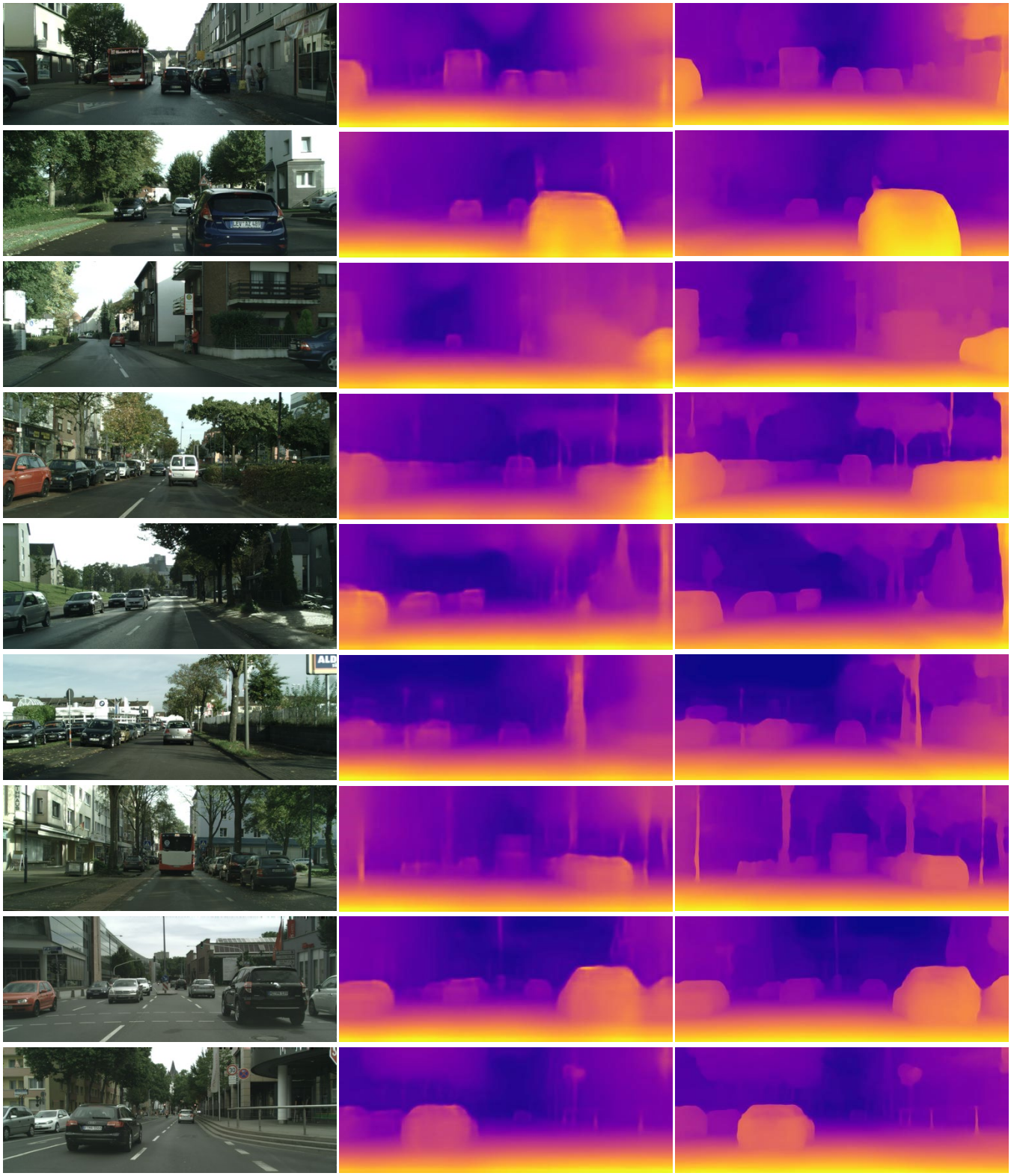


Figure 4: **Qualitative Comparisons on CityScapes test dataset**. *Left*: original input; *Middle*: estimated depth by full fine-tuning a U-Net from scratch; *Right*: estimated depth by our PPEA-Depth.