DELTAR: Depth Estimation from a Light-weight ToF Sensor and RGB Image - Supplementary Material -

Yijin Li[1], Xinyang Liu[1], Wenqi Dong[1], Han Zhou[1], Hujun Bao[1], Guofeng Zhang[1], Yinda Zhang[2⋆], and Zhaopeng Cui[1⋆]

[1] State Key Lab of CAD&CG, Zhejiang University
[2] Google

In this supplementary document, we provide additional experiments in Sec. A, describe more details on L5 principle in Sec. B and more implementation details in Sec. C , and discuss our limitation in Sec. D. Finally, we show more qualitative results in Sec. E. In this document, references that point to the main manuscript will be referenced as "P-".

A More Experiment Results

A.1 Different way to utilize L5 signal

We implement two variant ways to fuse the information from color and depth inspired by the depth-completion and depth-super-resolution methods, named "Point Fusion" and "Zone Fusion" respectively.

We first review what works in the fusion module. In the fusion module, in order to encode the distribution's location on the image, we establish the correspondence between the patch image and the distribution, and cross attention is only performed between the corresponding patch image and the distribution (see Fig. A-(a)). The difference between these two variants and our full model is the cross attention. As shown in Fig. A-(b), the cross attention of the full model is performed between all pixels' features within the patch and all sampled points' features of the distribution, while for zone fusion and point fusion, we remove the sampling operation, thus their distribution features are only the features extracted from the mean depth. Moreover, we only operate cross attention on the center pixel of the patch for point fusion.

The results are shown in Table A. The performance of these two variants drops significantly compared to our full model. It demonstrates that for our method, considering depth distribution brings an improvement of 1.8cm in RMSE while considering spatial correspondence between the image patch and L5's zones brings an improvement of 7.4cm in RMSE.

⋆ Corresponding authors

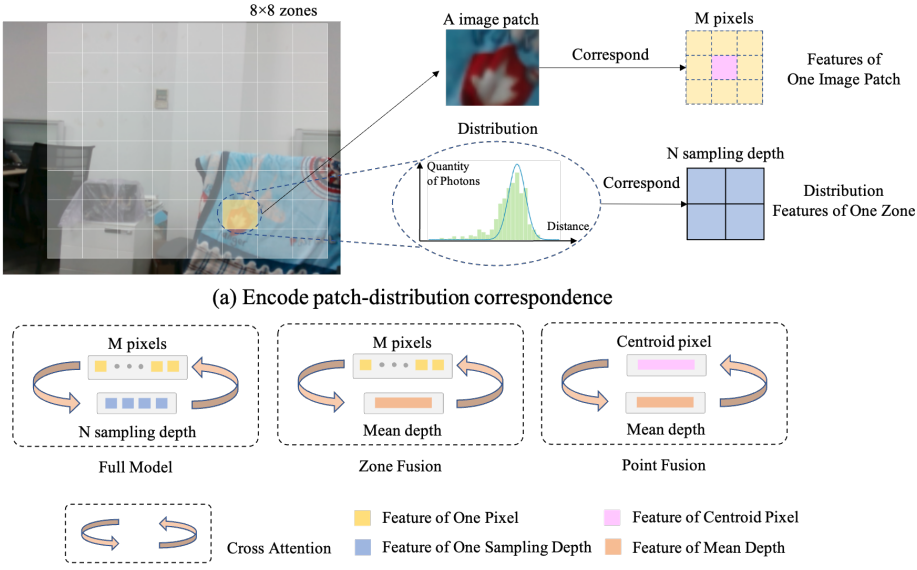


Fig. A. Implementation of zone Fusion and point Fusion. In (a) we review what works in the fusion module where the cross attention is only performed between the corresponding patch image and the distribution. In (b) we demonstrate how zone Fusion and point Fusion are derived. For both point Fusion and zone Fusion, we remove the sampling operation. For point Fusion, we only operate cross attention on the center pixel of the patch.

A.2 Impact of Training Data

It is worth noting that in the Table P-1, our method is only trained on synthetically simulated data on the NYU-Depth V2 and directly tested on the ZJU-L5 dataset without a finetune. The competitive performance indicates that the model generalizes reasonably well across datasets. To verify the impact of the collected real-world data, we fine-tune our network on the training set of the ZJU-L5 dataset, and show the evaluation results in Table B. As can be seen, the performance is further improved consistently as measured by all the metrics. This shows that real-world data is important for training to push for superior performance.

A.3 Effect of Sample Points' Number

To study the influence of the number of sample points on our distribution feature extractor, we train our network with various sampling numbers and measure the performance in terms of the Absolute Relative Error metric. The results are plotted in Fig. B. The error decreases as the number of sample points increases,

Models	$\delta_1 \uparrow$	REL$\downarrow$	RMSE$\downarrow$
Point Fusion	0.704	0.212	0.510
Zone Fusion	0.839	0.129	0.454
w/o Refine	0.850	0.126	0.462
Full	**0.853**	**0.123**	**0.436**

Table A. Comparison of different ways to utilize L5 signals.

Methods	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL$\downarrow$	RMSE$\downarrow$	$\log_{10} \downarrow$
Before Fine-tune	0.853	0.941	0.972	0.123	0.436	0.051
After Fine-tune	**0.880**	**0.959**	**0.982**	**0.112**	**0.415**	**0.049**

Table B. Results before and after fine-tuning with real-world data. Our model shows competitive performance if only trained on synthetic data, and can be further enhanced if fine-tuned with real-world data collected using our device setup.

and it basically converges when the number of sample points is more than 16. Therefore, in our experiments, we set the number of sample points to 16.

B More Details on L5 Principle

L5 utilizes a histogram-based algorithm to calculate the depth distribution within each zone. In detail, photons within each region are collected into histograms. Each bin in the histogram corresponds to a time window, representing the number of photons collected within a specific period. Multiplied by the speed of light, the depth range corresponding to a single bin is 4cm. A histogram contains 100 bins so L5 provides ranging up to 400 cm. Because the L5 is a device designed for low cost, in order to reduce the broadband's load, the measured histogram data is fitted with a normal distribution (see Fig.P-2), and finally, each zone only outputs the mean and variance of the distribution. L5 also returns a status code for each data. If it receives too few or the results are unstable, it will return an invalid code. See STMicroelectronics's webpage[3] for more details.

In our simulation of the L5 signal, we randomly crop a rectangle patch from the image and divide it equally into several zones. Then we simulate the L5 signals for each zone based on the ground truth depth map (see Sec. 5.1 in our paper). To mitigate the gap with the real L5 signals, we randomly dropout the zones to simulate the invalid signals from L5.

C More Implementation Details

C.1 Training Details

The proposed method is implemented in PyTorch [7] and is trained with AdamW [6] optimizer. We use the one-cycle policy [10] for the learning rate with max-lr =

[3] https://www.st.com/content/st_com/en/premium-content/premium-content-time-of-flight.html

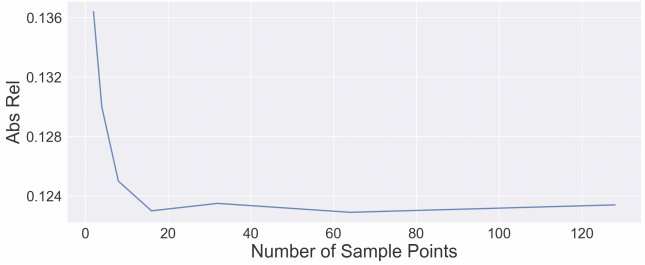


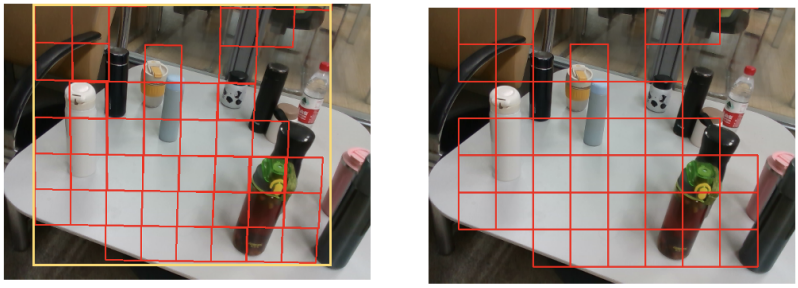
Fig. B. Effect of sample points' number in terms of Absolute Relative Error metric. The error decreases as the number of sample points increases and it starts to saturate when the number is bigger than 16.

3.0×10^{-4}. During training, it linearly increases from max-lr/25 to max-lr for the first 30% of iterations followed by cosine annealing to max-lr/75. We trained the network for 25 epochs with a batch size of 16, and it took 25 minutes per epoch on a single node with three NVIDIA RTX 3090 24GB GPUs.

C.2 Aligning L5's Zones to Image

For aligning the L5's zones to the image, only extrinsic parameters are not enough, because we do not know the precise depth and pixel-wise alignment which is required to conduct cross-image projection. We make an approximation that the depths of the zones are all equal to the mean of the depth distribution and compute projection of the zones' corner in the image by $p = K_c K_{L5}^{-1}(x, y, d)$, where K_c and K_{L5} are intrinsic parameters of camera and L5, respectively. (x, y) is the corner's pixel position in L5 and d is the depth. A sample after alignment is shown in Fig. C-(a).

Note that each region is no longer a precise axis-aligned rectangle after being warped to the color image and they do not share the same shape. However, we noticed that they are indeed very close to the rectangle due to the similar facing direction between the color camera and L5 sensor and they have almost the same shape. Let them become rectangles and share the same shape can simplify the following grouping operation (i.e., grouping them into a batch) without additional operations like padding and masking. To do this, we first fit them with a minimum axis-aligned bounding box (the yellow rectangle shown in Fig. C-(a)), and then divide the bounding box into 8×8 blocks equally to get the corrected zones. The rectification result is shown in the Fig. C-(b). We also show examples in Fig. D where the color images are blended with L5' depth. According to the status returned by L5, we hide the invalid zones which may receive too few photons or are unstable. It can be seen that L5 sometimes return noisy and even wrong measurement which may be caused by multi-path interference [3] (see the bottom right image in Fig. D, there is a white zone on the wall). As a result, it is non-trivial to improve the raw signal from L5 because of the measurement's extremely low resolution and high depth uncertainty. Thanks to our novel design



(a) Fitting a compact rectangle on warped zones (b) Rectification result

Fig. C. Rectification of the warped zones. Making the warped zones axis-aligned rectangles and share the same shape can simplify the grouping operation (i.e., grouping them into a batch) without additional operations like padding and masking. We do this by firstly (a) fitting with a minimum axis-aligned bounding box, then (b) dividing the bounding box into 8×8 blocks equally and getting the rectification result.

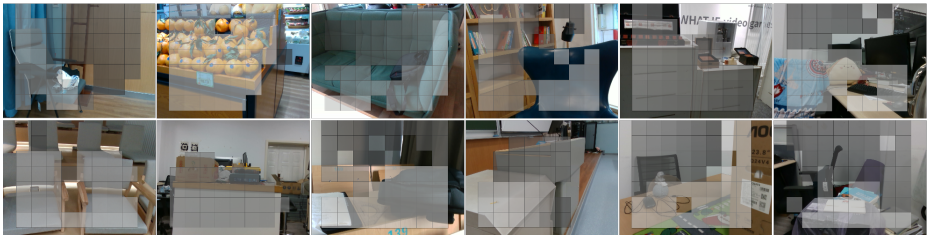


Fig. D. Blending color images with L5' depth. White color represents close range, black color represents long range.

of distribution feature extractor and the fusion network, we improve the depth quality of L5 and make it even on par with a commodity-level RGB-D sensor (i.e., Intel RealSense D435i).

C.3 Definition of Evaluation Metrics

We evaluate the performance using the following standard metrics where $\hat{d}_i$ represents predicted depth, d_i represents ground truth depth, and N is the number of valid ground truth values:

– Threshold Accuracy (δ_i with i=1,2,3):

$$\frac{\sum_{i=1}^{N}[max(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}) < 1.25^i]}{N} \tag{1}$$

where [] denotes Iverson brackets.

Name	Input	Layer Description	Output	Output Tensor Dim.
Input	/	Image	#1	H×W×3
	/	64 zones × N Sampling Points	#2	64×N×1
Distribution Depth Encoder				
PointNet_1	#2	MLP(1,40,40,40)	#3	64×N×40
PointNet_2	#3	MLP(40,64,64,64)	#4	64×N×64
PointNet_3	#4	MLP(64,176,176,176)	#5	64×N×176
Image Encoder				
Block_1	#1	Block1 from EfficientNet B5	#6	$1/2$H×$1/2$×24
Block_2	#6	Block2 from EfficientNet B5	#7	$1/4$H×$1/4$×40
Block_3	#7	Block3 from EfficientNet B5	#8	$1/8$H×$1/8$×64
Block_4	#8	Block5 from EfficientNet B5	#9	$1/16$H×$1/16$×176
Block_5	#9	Block8 from EfficientNet B5	#10	$1/32$H×$1/32$×2048
Fusion Module				
Transformer_1	#5,#9	TransEncoder(num_heads=4)	#11	$1/16$H×$1/16$×176
Decoder_1	Concat(#11,#10)	(Conv2d+BN+LeakyReLU)×2	#12	$1/16$H×$1/16$×1024
Transformer_2	#4,#8	TransEncoder(num_heads=4)	#13	$1/8$H×$1/8W$×64
Decoder_2	Concat(#13,#9)	(Conv2d+BN+LeakyReLU)×2	#14	$1/8$H×$1/8W$×512
Transformer_3	#3,#7	TransEncoder(num_heads=4)	#15	$1/4$H×$1/4$×40
Decoder_3	Concat(#15,#8)	(Conv2d+BN+LeakyReLU)×2	#16	$1/4$H×$1/4$×256
Decoder_4	Concat(#16,#7)	(Conv2d+BN+LeakyReLU)×2	#17	$1/2$H×$1/2$×128
Refinement Module				
RefineBlock_1	#17	MiniViT [1]	Depth	H×W×1

Table C. Details of our network architecture. Here we omit the details of refinement module as it can be found in [1].

– Mean Absolute Relative Error (REL):

$$\frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{d}_i - d|}{d} \tag{2}$$

– Root Mean Square Error (RMSE):

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{d}_i - d)^2} \tag{3}$$

– Average(log_{10}) Error:

$$\frac{1}{N}\sum_{i=1}^{N} log_{10}(|\hat{d}_i - d|) \tag{4}$$

C.4 Network Architecture

Our network details are listed in Table C. The transformer encoder in our fusion module (denoted as "TransEncoder") adopts the common designs of residual connection and layer normalization [5, 11]. For the sake of the computational cost, in our experiments, we only use the transformer in the first three fusion modules and not in the last one with $(H/2, W/2)$ resolution.

D Limitation

Our current system has some limitations, which could be fixed in future works. First, the network run-time is close but not fast enough for real-time performance. It takes about 74ms to process a 640×480 color image and L5 signals on a single GPU (NVIDIA RTX 2080 Ti), which might be insufficient for downstream applications such as SLAM. Note that, in this work, we did not optimize the network architecture for the run-time efficiency, and many standard approaches, such as [4, 8, 2], can be directly adopted. Another problem is that the system performance degenerates when the L5 signal is not available or noisy, for example, on the pixels that are not covered by the L5 signals or at the location beyond the L5's depth range. In future work, we hope to improve the depth quality out of L5's working range by combining surface normal as previous methods do [9, 12]. Lastly, removing the regular grid assumption in the alignment may further improve the quality and generalize our method for arbitrary camera/sensor placement.

E More Qualitative Results

We show more qualitative results on the ZJU-L5 dataset in Fig. E and Fig. F. Compared to other methods, our method makes the best use of high-resolution color images and low-quality L5 signals and produces the most accurate depths that are rich of details.

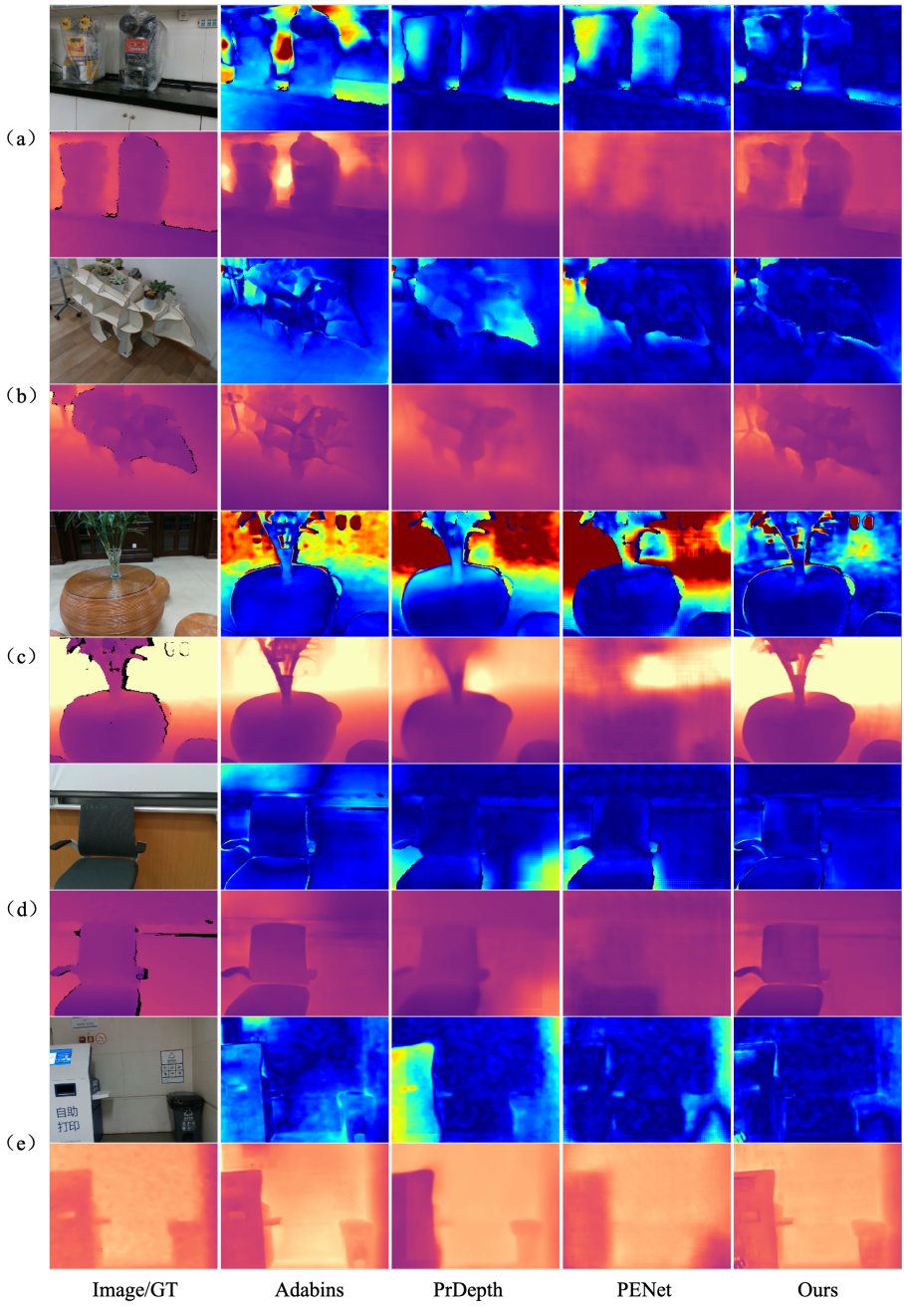


Fig. E. More qualitative comparison on ZJU-L5 dataset. Monocular estimation [1] tends to make mistakes even after aligned with real-world metric scale. Depth super-resolution and depth completion produce quite blurry results. On the contrary, our method makes the best use of high-resolution color image and low-quality L5 signals, and produces the most accurate depths that are rich of details.

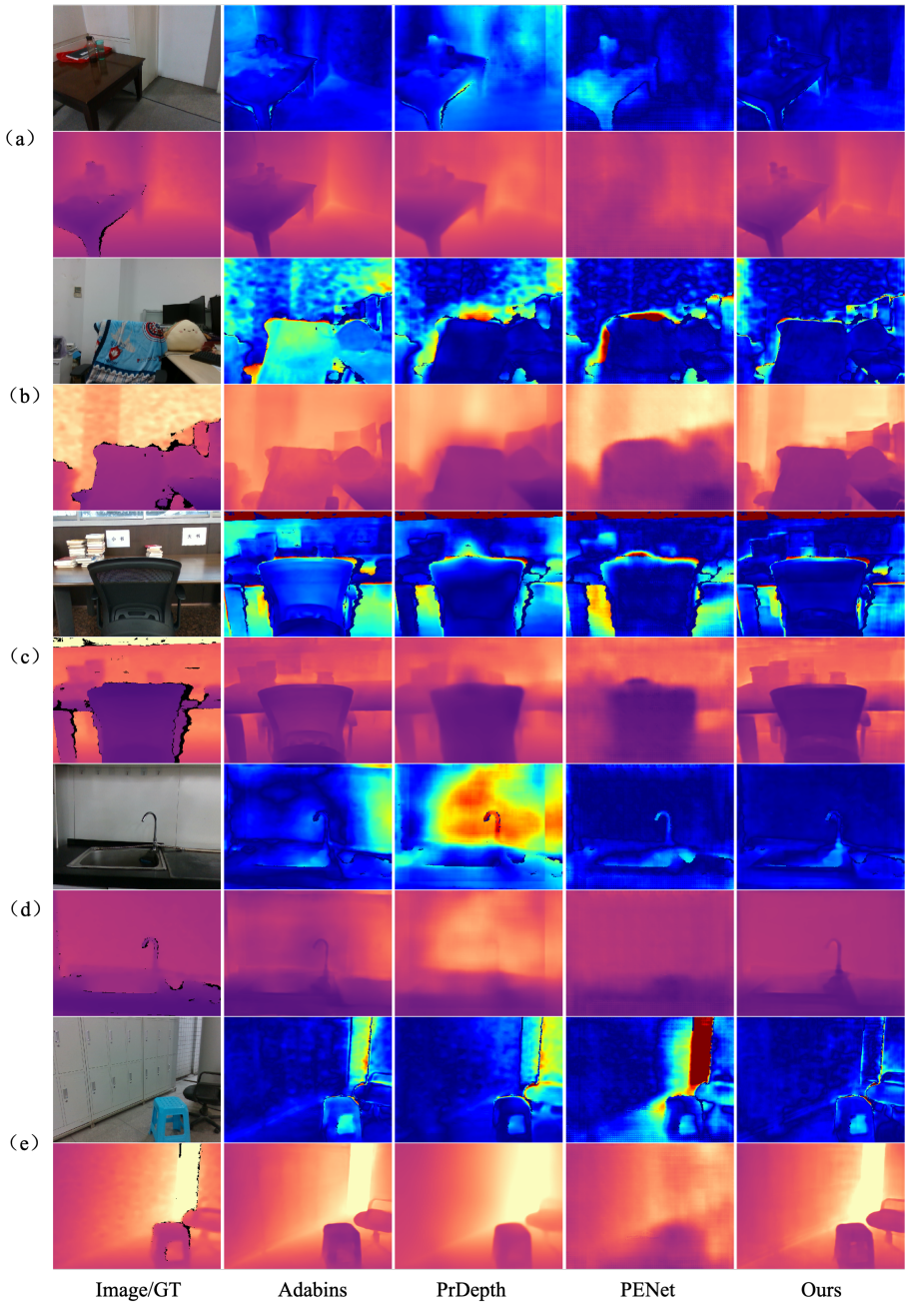


Fig. F. More qualitative comparison on ZJU-L5 dataset. Monocular estimation [1] tends to make mistakes even after aligned with real-world metric scale. Depth super-resolution and depth completion produce quite blurry results. On the contrary, our method makes the best use of high-resolution color image and low-quality L5 signals, and produces the most accurate depths that are rich of details.

References

1. Bhat, S.F., Alhashim, I., Wonka, P.: Adabins: Depth estimation using adaptive bins. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4009–4018 (2021)
2. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:2103.13630 (2021)
3. Gupta, M., Nayar, S.K., Hullin, M.B., Martin, J.: Phasor imaging: A generalization of correlation-based time-of-flight imaging. ACM Transactions on Graphics (ToG) **34**(5), 1–18 (2015)
4. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
5. Lin, T., Wang, Y., Liu, X., Qiu, X.: A survey of transformers. arXiv preprint arXiv:2106.04554 (2021)
6. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
7. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
8. Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: Binary neural networks: A survey. Pattern Recognition **105**, 107281 (2020)
9. Qiu, J., Cui, Z., Zhang, Y., Zhang, X., Liu, S., Zeng, B., Pollefeys, M.: Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3313–3322 (2019)
10. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-domain Operations Applications. vol. 11006, p. 1100612. International Society for Optics and Photonics (2019)
11. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8922–8931 (2021)
12. Zhang, Y., Funkhouser, T.: Deep depth completion of a single rgb-d image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 175–185 (2018)