Acknowledgements. We thank anonymous reviewers for their constructive comments. This work was funded in part by a research contract from Galen Robotics and in part by Johns Hopkins University internal funds.

Disclosure. Under a license agreement between Galen Robotics, Inc. and the Johns Hopkins University, Dr. Russell H. Taylor and the University are entitled to royalty distributions on technology related to technology described in the study discussed in this publication. Dr. Russell H. Taylor also is a paid consultant to and owns equity in Galen Robotics, Inc. This arrangement has been reviewed and approved by the Johns Hopkins University in accordance with its conflict-of-interest policies.

Appendix

A. Efficient Implementation of Attention with Relative Positional Encoding

The attention with relative positional encoding is computed as

$$\alpha_{i,j} = \underbrace{e_{I,i}^T W_{\mathcal{Q}}^T W_{\mathcal{K}} e_{I,j}}_{\text{(1) data-data}} + \underbrace{e_{I,i}^T W_{\mathcal{Q}}^T W_K e_{p,i-j}}_{\text{(2) data-position}} + \underbrace{e_{p,i-j}^T W_{\mathcal{Q}}^T W_{\mathcal{K}} e_{I,j}}_{\text{(3) position-data}} .$$

In the following context, we use term (2) data-position (D2P) as an example. For simplicity, we denote $e_{I,i}^T W_{\mathcal{Q}}^T$ as $\mathcal{Q}_i$ and $W_{\mathcal{K}} e_{p,i-j}$ as $K_{r,i-j}$. In term (2) for any i, j, the relative distance follows a fixed pattern [6] shown as

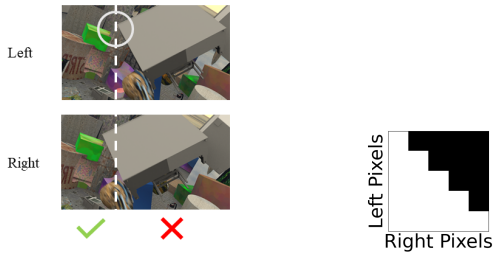
$$\text{D2P} = \begin{bmatrix} \mathcal{Q}_0 K_{r,0} & \mathcal{Q}_0 K_{r,-1} & \cdots & \mathcal{Q}_0 K_{r,-I_w+1} \\ \mathcal{Q}_1 K_{r,1} & \mathcal{Q}_1 K_{r,0} & \cdots & \mathcal{Q}_1 K_{r,-I_w+2} \\ & & \vdots & \\ \mathcal{Q}_{I_w-1} K_{r,I_w-1} & \mathcal{Q}_{I_w-1} K_{r,I_w-2} & \cdots & \mathcal{Q}_{I_w-1} K_{r,0} \end{bmatrix}$$

where the first row starts with a relative distance of 0 and ends with a relative distance of $-I_w + 1$. Subsequent rows simply offset the first row by increasing amounts. Furthermore, since the relative distance is bounded by $I_w - 1$ and $-I_w + 1$, therefore we can pre-compute all possible values of $K_{r,i-j}$ as

$$\tilde{K}_r = \begin{bmatrix} K_{r,I_w-1} \\ K_{r,I_w-2} \\ \vdots \\ K_{r,-I_w+1} \end{bmatrix}$$

We then slice $\tilde{K}_r$ with an increasing offset, which can be done computation-lightly to obtain

$$K_r = \begin{bmatrix} K_{r,0} & K_{r,-1} & \cdots & K_{r,-I_w+1} \\ K_{r,1} & K_{r,0} & \cdots & K_{r,-I_w+2} \\ & & \vdots & \\ K_{r,I_w-1} & K_{r,I_w-2} & \cdots & K_{r,0} \end{bmatrix}$$



(a) Left Right Geometry Visualization (b) Attention Mask

Figure 1. Attention mask visualization, where white indicates allowable attention region while black indicates forbidden attention region.

The term (2) D2P can then be computed as a matrix product between $\mathcal{Q}$ and K_r. The term (3) position-data can be computed similarly, thus, reducing the computation cost of relative position.

B. Attention Mask

As described in Sect. 3.2.4, if x_L, x_R represent the x-coordinates in pixel-space for the projection of a physical point into the left and right images, then $x_R \leq x_L$ for all physical points (with positive x-axis pointing to the right). Therefore when searching for correspondences, the network only needs to search for pixels in the right image that are to the left of the current pixel position in the left image. An example is given in Fig. 1(a): If the corner of the table (highlighted with the circle) is to be matched, the network only needs to search to the left of the dashed line in the right image. This can be achieved using a binary attention mask shown in Fig. 1(b), where for each pixel in the left image, the allowable attended pixel locations in the right image marked as white are to the left of source pixel location. Mathematically, this binary attention mask can be achieved by setting the values of forbidden attended pixels (marked as black in Fig. 1(b)) in the attention matrix to negative infinity. Thus, after softmax, the attention values α_h in Equation 2 of those pixels will be zero and they'll be excluded from the disparity computation.

C. Relative Positional Encoding

In Fig. 2, we visualize the evolution of feature descriptors as the Transformer updates them **without the positional encoding**. It is worth noting the textureless region, such as the table, does not have distinct patterns to resolve matching ambiguities.

In contrast, the evolution of feature descriptors **with positional encoding** in Fig. 3 propagates the edge information into the center of the textureless region progressively, which helps to resolve the ambiguity.

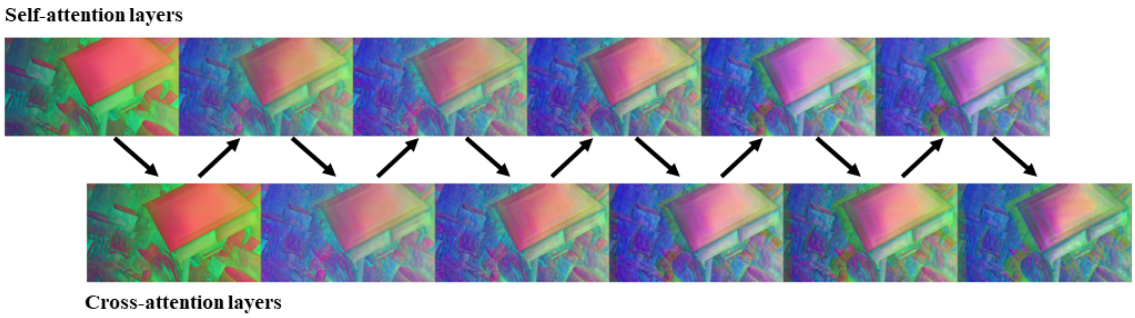


Figure 2. Evolution of feature map **without positional encoding**. First row - input to Transformer self-attention layers 1-6. Second row - input to Transformer cross-attention layers 1-6.

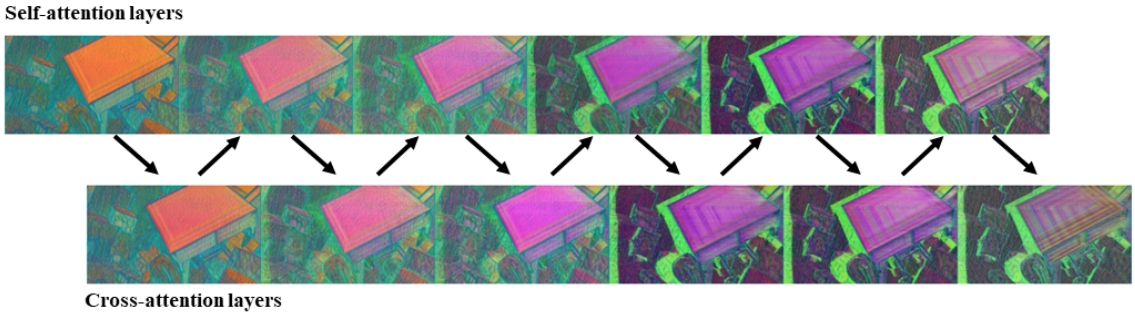


Figure 3. Evolution of feature map **with positional encoding**. First row - input to Transformer self-attention layers 1-6. Second row - input to Transformer cross-attention layers 1-6.

D. Attention Span

We compute attention span of both the self- and cross-attention layers, which is the spatial span of pixels that are above the uniform attention value (i. e., $1/I_w$ with I_w being image width). The layer-wise attention span in Fig. 4 illustrates how self- and cross-attention shift from a global context to local one as processing moves to higher layers in the network. This is particularly true for cross-attention, where the final attention span is only around 15 pixels (0.01 of image width).

The evolution of self-attention and cross-attention are also visualized in Fig. 5 and Fig. 6, where brighter regions are the more attended regions. It can be observed that the attention span from the source pixel (labeled as a red cross-hair) slowly converges to local context as the Transformer progresses, confirming the quantitative result in Fig. 4. Both attention mechanisms stay focused on edges even if they are far away from the source pixel (i.e., not within the local context). In cross-attention, it can be observed that the final attention shrinks towards the target pixel (labeled as a blue cross-hair).

E. Generalization Mechanism

We visualize the input feature maps to the Transformer using UMAP [13] in Fig. 7, where the dimensionality reduced embedding is trained only on Scene Flow data. Each data point represents a pixel that the Transformer operates on. We observe that the representations learned by STTR cluster into two regions (Fig. 7(a)). To further understand this clustering phenomenon, we visualize the corresponding pixels using a color mask belonging to one of the clusters. The intensity of a pixel in the color mask is higher if it is closer to the centroids of the cluster. Interestingly, the feature extractor groups pixels into textured (blue) and textureless (red) regions. Pixels with a higher intensity in the color mask are mostly correlated to texture edges. To verify that the blue region indeed contains more texture than the red region, we compute the mean Sobel edge-gradient on the nor-

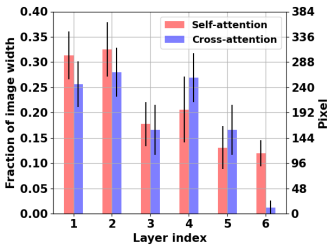


Figure 4. Attention span of both self- and cross-attention evaluated on Scene Flow dataset. Image resolution 960×540. Left: attention span in fraction of image width. Right: attention span in pixels.

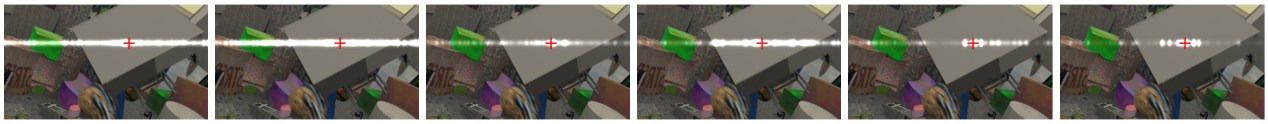


Figure 5. Evolution of self-attention of left image pixel on left image, with source pixel labeled as red cross-hair. First row - attention map of self-attention layers 1-6. Second row - attended pixels of self-attention layers 1-6 are highlighted.

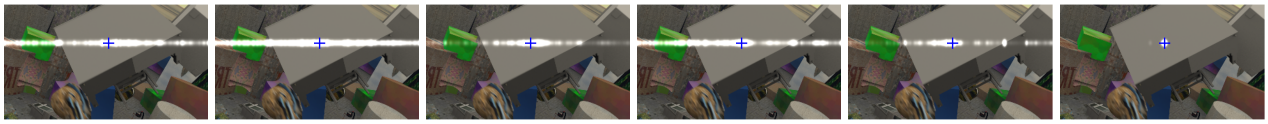


Figure 6. Evolution of cross-attention of source left image pixel on right image, with target ground truth pixel labeled as blue cross-hair. First row - attention map of cross-attention layers 1-6. Second row - attended pixels of cross-attention 1-6 are highlighted.

Table 1. Quantitative comparison of mean edge-gradient based on intensity computed on each dataset in the blue and red clusters.

Dataset	Red Cluster	Blue Cluster
Scene Flow	0.46	**12.02**
MPI Sintel	16.70	**19.15**
KITTI 2015	4.06	**23.67**
Middleburry 2014	7.73	**27.48**
SCARED	**17.39**	15.42

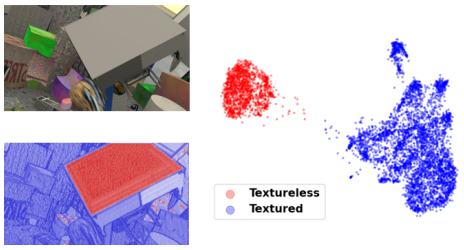
malized image intensities of each dataset. The result is summarized in Table 1 where, with the exception of SCARED, all datasets have larger magnitude edge-gradients in the blue cluster than red cluster, confirming that blue cluster contains more "texture" than red cluster. In Fig. 7(b), we show that regardless of the domain, the embeddings are always contained within the same space. We additionally show the individual UMAP reduction of features extracted from MPI Sintel, KITTI 2015, Middlebury 2014, and SCARED dataset on top of Scene Flow in Fig. 8(a-d). We hypothesize that this implicitly learnt feature clustering improves the generalization of STTR and makes the Transformer matching process easier.

F. Qualitative result of context adjustment layer (CAL)

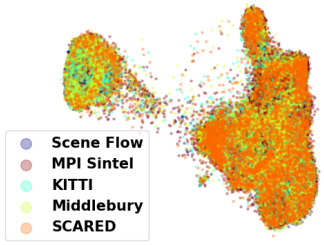
We provide visualizations of context adjustment layer's effect in Fig. 9. Comparing the CAL output in Fig. 9(a), the prediction without CAL in Fig. 9(b) the result lacks smoothness due to lack of cross eipolar-line context.

G. Attention Stride

Since the attention module is flexible with respect to the stride of features over which to attend, STTR can run at a faster speed and lower memory footprint at the cost of performance. We do *not* have to re-train STTR as attention



(a) Scene flow



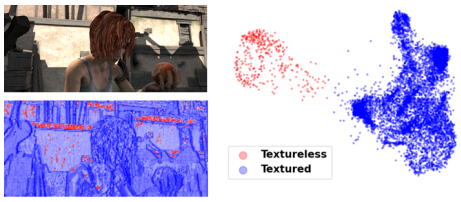
(b) All datasets

Figure 7. (a) UMAP visualization of feature map (right) and the corresponding color mask (bottom left) of input image (top left). (b) Umap visualization of all dataset.

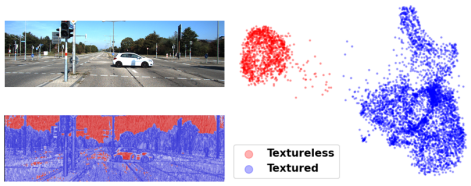
Table 2. Ablation result on attention stride. Input image resolution is 960×540. Inference performance reported are median memory in GB ↓ and speed in Hz ↑ over 100 runs.

Attention Stride	3 px Error ↓	EPE ↓	Occ IOU ↑	Inference Performance
3	**1.26**	**0.45**	0.92	7.4 / 1.35
4	1.43	0.71	**0.97**	3.8 / 2.76
5	1.70	1.04	0.96	**2.2** / **4.36**

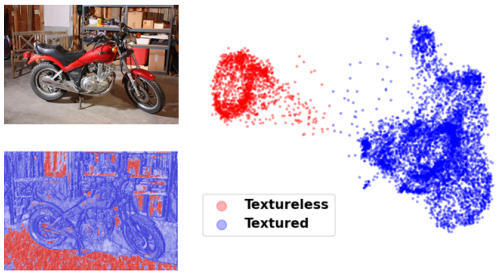
stride is only an inference hyperparameter. The result is summarized in Table 2.



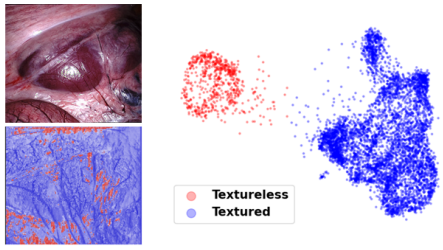
(a) MPI Sintel



(b) KITTI 2015

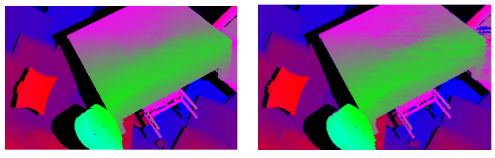


(c) Middlebury 2014



(d) SCARED

Figure 8. UMAP visualization of feature map from each domain.



(a) Disparity **with** CAL (b) Disparity **without** CAL

Figure 9. Qualitative result of disparity with/without CAL.

H. Lightweight Implementation

An additional lightweight model is implemented for STTR for faster inference speed and lighter memory consumption while maintaining the similar performance as STTR. The major changes include:

- We remove the flexibility in STTR where attention stride s can change during inference but instead fix it to $s = 4$. Thus, the features extracted do not need to be maintained at the full image resolution any more. This reduces memory consumption.
- As discussed in Equation 13, the memory consumption is proportional to the number of heads N_h. On the other hand, the number of parameters is proportional to $N_h C_h$. Therefore, we can maintain the same number of parameters while halving memory consumption by increasing C_h by two and decreasing N_h by two.

We use the same training protocol for the lightweight model as discussed in Sect. 4. We compare the performance of the lightweight STTR and STTR on the Scene Flow benchmark in Table 3 and cross-domain generalization performance on different datasets in Table 4. As shown in Table 3, the performance of lightweight STTR drops compared to STTR in Scene Flow evaluation, especially in terms of 3 px Error and EPE, while inference performance improves with less memory and faster speed. The generalization performance improves in EPE for MPI Sintel, 3 px Error and EPE for Middlebury 2014 and SCARED, while worsens in 3 px Error for MPI Sintel and 3 px error and EPE for KITTI 2015. The occlusion IOU consistently worsens compared to STTR.

Table 3. Evaluation on Scene Flow. Inference performance reported are median memory in GB ↓ and speed in Hz ↑ over 100 runs.

	3 px Error ↓	EPE ↓	Occ IOU ↑	Inference Performance
STTR ($s = 3$)	**1.26**	**0.45**	0.92	7.4 / 1.35
STTR ($s = 4$)	1.43	0.71	**0.97**	3.8 / 2.76
Lightweight STTR	1.54	0.50	**0.97**	**2.6** / **5.50**

I. Inference Memory Consumption and Speed

As discussed in Sect. 3.5, STTR can run at a constant memory and speed without a manually limited disparity range. We compare the inference speed and memory consumption with the prior work where a larger disparity range will consume more memory and slow down inference speed. Since PSMNet [4] uses a feature downsampling rates of 4 and AANet [16] uses a feature pyramid, we set attention stride $s = 4$ for STTR to match the setting in PSMNet. As shown in Table 5, in order for prior work to predict a larger disparity range, the memory consumption increases while inference speed drops. In comparison, STTR runs with a constant memory consumption and speed.

Table 4. Generalization without fine-tuning on MPI Sintel, KITTI 2015, Middlebury 2014, and SCARED dataset. Trained only on Scene Flow dataset.

	MPI Sintel			KITTI 2015			Middlebury 2014			SCARED		
	3 px Error ↓	EPE ↓	Occ IOU ↑	3 px Error ↓	EPE ↓	Occ IOU ↑	3 px Error ↓	EPE ↓	Occ IOU ↑	3 px Error ↓	EPE ↓	Occ IOU ↑
STTR	**5.75**	3.01	**0.86**	**6.74**	**1.50**	**0.98**	6.19	2.33	**0.95**	3.69	1.57	**0.96**
Lightweight STTR	5.82	**2.95**	0.69	7.20	1.56	0.95	**5.36**	**2.05**	0.76	**3.30**	**1.19**	0.89

Table 5. Evaluation of inference memory in GB ↓ and speed in Hz ↑ across different disparity ranges. Image resolution (W×H) and maximum disparity values are in pixels. Reported are median values across 100 runs. N/A: disparity range exceeds image width.

Network	Image Resolution	Maximum Disparity				
		192	384	576	768	960
PSMNet [4]	960 × 576	3.9 / 2.21	7.6 / 1.20	11.4 / 0.85	15.1 / 0.62	18.8 / 0.50
AANet [16]		0.6 / 14.01	0.7 / 9.65	0.9 / 6.91	1.1 / 5.59	1.3 / 4.28
STTR		3.8 / 2.76				
Lightweight STTR		2.0 / 4.42				
PSMNet [4]	672 × 480	2.3 / 3.89	4.5 / 2.00	6.6 / 1.39	N/A	
AANet [16]		0.3 / 20.14	0.4 / 14.70	0.5 / 11.06	N/A	
STTR		1.8 / 5.77			N/A	
Lightweight STTR		0.87 / 9.11			N/A	
PSMNet [4]	384 × 192	Resolution Too Small		N/A		
AANet [16]		0.1 / 22.1	0.1 / 21.4	N/A		
STTR		0.3 / 18.9		N/A		
Lightweight STTR		0.2 / 25.6		N/A		

J. Dataset Information and Pre-processing

Scene Flow [12] FlyingThings3D Full dataset (final pass) provides realistic artifacts but does not provide occlusion information. Therefore, we subsample the Full dataset using the corresponding occlusion information from DispNet/FlowNet2.0 dataset. After pre-processing, Scene Flow contains 21818 training images of resolution 960×540, with maximum disparity 602 px (0.67 of image width). The test dataset contains 4248 images with maximum disparity 468 (0.49 of image width). MPI Sintel [3] contains 1063 images of resolution 1024 × 436, with maximum disparity 487 px (0.46 of image width). KITTI 2015 [14] contains 200 images of resolution 1242×375, with maximum disparity of 192 px (0.15 of image width). We note that pixels with disparities larger than 192 are intentionally masked out in the dataset. Middlebury 2014 [15] quarter resolution subset contains 15 images of various image resolution, with maximum disparity of 161 px (0.22 of image width). SCARED [1] requires additional pre-processing since it only provides the depth data and corresponding camera intrinsics parameters. There are 7 subsets in total, containing 27 videos. Since subsets {4,5} contain very large camera intrinsic errors [1], we choose to leave them out of our evaluation since this introduces unnecessary uncertainty. Furthermore, other than the first frames of each video, subsequent frames are interpolated using the kinematics information of the robot with synchronization and kinematics error. Therefore, the depth values for subsequent frames are not accurate. We also exclude those images for reliable evaluation. The left and right 100 pixels were cropped due to invalidity after rectification. After pre-processing, SCARED contains 19 images of resolution 1080 × 1024 with maximum disparity of 263 px (0.24 of image width).

K. Loose Analogy to Biological Stereo Vision

It has been shown that the biological stereo vision system (e. g. that of a human) perceives depth from stereo images by relying on low-level cortex cues. One example that demonstrates this effect is the random-dot stereograms experiment [9], where there is no meaningful texture in the images but only random dots; yet, humans can still perceive the 3D objects. At the same time, the biological vision system imposes geometric assumptions on objects as a piecewise smoothness prior [7], which involves mid-level cortex processing. In a way, STTR emulates the biological stereo system in that the Transformer processes the images at a low-level to finds matches between features. STTR then locally refines the raw disparity using the context adjustment layer, which is in loose analogy to mid-level vision.

L. Training Time and Number of parameters

The total training time on Scene Flow dataset [12] for STTR is approximately 120 hours on one Titan RTX GPU with batch size 1 for 15 epochs. We note that the training time will vary with the batch size and number/type of GPUs used. The number of parameters of STTR compared with contemporary architectures is summarized in Table 6. Other than LEAStereo [5] which is optimized using Neural Architecture Search, STTR has the least number of parameters.

Table 6. Number of parameters in contemporary architectures for stereo depth estimation.

Approach	**Network**	**Params** [M]
3D Convolution	GANet-11 [18]	6.6
	PSMNet [4]	5.2
	GC-Net[10]	3.5
	LEAStereo [5]	**1.8**
Correlation	iResNet [11]	43
	DispNetCorr1D [12]	42
	HD^3 [17]	39
	AANet [16]	3.9
Hybrid	GwcNet-g [8]	6.5
Classification	Bi3D [2]	37
Transformer	**STTR (Ours)**	2.5

References

[1] Max Allan, Jonathan Mcleod, Cong Cong Wang, Jean Claude Rosenthal, Ke Xue Fu, Trevor Zeffiro, Wenyao Xia, Zhu Zhanshi, Huoling Luo, Xiran Zhang, et al. Stereo correspondence and reconstruction of endoscopic data challenge. *arXiv preprint arXiv:2101.01133*, 2021.

[2] Abhishek Badki, Alejandro Troccoli, Kihwan Kim, Jan Kautz, Pradeep Sen, and Orazio Gallo. Bi3d: Stereo depth

estimation via binary classifications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1600–1608, 2020.

[3] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[4] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[5] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *arXiv preprint arXiv:2010.13501*, 2020.

[6] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[7] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429. IEEE, 2009.

[8] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019.

[9] Bela Julesz. Foundations of cyclopean perception. 1971.

[10] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[11] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 2018.

[12] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[13] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[14] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[15] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014.

[16] Haofei Xu and Juyong Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1959–1968, 2020.

[17] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6044–6053, 2019.

[18] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.