Acknowledgements We thank Johannes Kopf and Michael Zollhoefer for helpful discussions, and Vladimir Tankovich for HITNet implementation. We thank anonymous reviewers and meta reviewers for their feedback on the paper. This work was done during internship at Reality Labs, Meta Inc.

A. Network Details

Our code base is built upon MMSegmentation [3].

A.1. Stereo Network

We use HITNet [11] as our per-frame stereo network to extract disparity and features on a per-frame basis. HITNet builds a initial cost volume with a pre-specified disparity range. In our implementation, we set the maximum disparity to be 320 following the original implementation. The channel size of extracted features is 24, which is used in fusion network to evaluate disparity confidence. The memory state generated by the stereo network includes the frame-wise estimated disparity, the feature of left and right images at the 1/4 resolution for disparity confidence computation, and left RGB image.

Our HITNet code is adapted from open-sourced implementation[1][2] in PyTorch [9] and confirmed by the authors of HITNet. We note that the HITNet adapted is based on Table 6 of [11] instead of the HITNet XL or HITNet L in Table 1 of [11]. As HITNet works across different datasets (Table 2-4 in [11]), we have chosen HITNet as our stereo network. As described in Sect.4.2, we remove pixels with extreme scene flow (>210px) and disparity (<1px or >210px). These removed pixels correspond to unrealistically fast, far or close objects during simulation, which are not considered in our intended applications such as augmented reality. In comparison, [11] removes pixels with disparity >192px. Thus, the reported per-pixel accuracy performance differs slightly on FlyingThings3D.

A.2. Motion Network

Our motion network builds on top of RAFT3D [13]. Specifically, we adapt the inter-frame transformation $\boldsymbol{T}$ estimation process. We first provide the necessary background of the estimation process and detail the difference between our motion network and RAFT3D next paragraph. Let $\mathbf{p}$ be a location in pixel coordinates and $\mathbf{P}$ be a location in Cartesian coordinates. RAFT3D uses a context extractor to provide semantic information for object grouping, and a feature extractor for cross-frame correspondence matching. A piece-wise rigid constraint is realized by grouping pixels based on their motion and appearance similarities and enforcing the motion within a group to be the same. The GRU module takes the image context feature, correlation information, current estimated transformation, and corresponding scene flow as input. The GRU then makes corrections $\Delta\mathbf{s}^k$ to the scene flow, extracts a set of rigid-embeddings $\mathcal{V}$ and associated confidence of estimates $\mathbf{w}$. A differentiable Gauss-Newton (GN) optimization step is performed to update the transformation based on the residual error (Eq. (1)). The update scheme is iterative for $\mathcal{K}$ steps, with the transformation $\boldsymbol{T}^0$ and flow $\mathbf{s}^0$ initialized to identity and zero respectively. Given the i-th pixel at $t-1$ and its local m neighborhood pixels $\mathcal{N}_m(i)$, the k-th iteration residual error to be minimized is:

$$E_i^k(\zeta_i) = \sum_{j \in \mathcal{N}_m(i)} \mathcal{V}_{ij} \| \mathbf{p}_j + \mathbf{s}_j^k - \pi(e^{\zeta_i^k} \boldsymbol{T}_i^{k-1} \mathbf{P}_j) \|_{2,\mathbf{w}_j} \tag{1}$$

$$\mathbf{s}_j^k = \pi(\boldsymbol{T}_i^{k-1} \mathbf{P}_j) - \mathbf{p}_j + \Delta\mathbf{s}_j^k \tag{2}$$

$$\mathcal{V}_{ij} = \| \mathcal{V}_i - \mathcal{V}_j \|_2 \tag{3}$$

where $e^{\zeta_i^k}$ is the incremental motion on the SE3 manifold to be made to the previous transformation $\boldsymbol{T}_i^{k-1}$, $\mathbf{s}_j^k$ is the raw predicted scene flow, $\| \cdot \|_{2,\mathbf{w}}$ is weighted ℓ_2 norm, $\mathbf{w}_j$ is the flow confidence, and $\mathcal{V}_{ij}$ is the ℓ_2 distance of rigid-embeddings. The updated transformation is $\boldsymbol{T}^k = e^{\zeta^k} \boldsymbol{T}^{k-1}$. The final scene flow induced transformation from transformation prediction $\boldsymbol{T}$ in image coordinate can be computed as:

$$\mathbf{s}_{\boldsymbol{T}} = \pi(\boldsymbol{T} \pi^{-1}(\mathbf{p})) - \mathbf{p} \tag{4}$$

The magnitude $\mathbf{s}_{\boldsymbol{T}}$ and confidence $\mathbf{w}_j$ of the flow is used later in fusion network.

The difference between the transformation $\boldsymbol{T}$ prediction process of our motion network and RAFT3D is summarized in this paragraph. As shown in Eq. (1), the similarities between rigid embeddings of pixels critically determines if two pixels are grouped together as the same object. Thus, generating high-quality rigid embeddings has a direct impact of motion accuracy. Originally, RAFT3D uses a pre-trained ResNet-50 [4] as the context extractor to provide semantics information. Instead, we replace the pre-trained ResNet-50 with a network similar to HRNet [15] that aggregates the context information in a hierarchical level. An overview is shown in Fig. 1. The context extractor extracts the semantic information from four resolutions – $1/4, 1/8, 1/16$ and $1/32$ resolutions. We use bilinear upsampling to resize all feature maps to the $1/8$ resolution and use one 1×1 convolution to aggregate the information. Compared to ResNet-50, which has 40M parameters, our context extractor only has 3M parameters. We found that our motion network outperforms RAFT3D on the FlyingThings3D [7] dataset while having 1/5-th of the parameters of RAFT3D. In our implementation, we use the RGB image as semantic input instead of features from stereo to reduce dependency on the stereo network.

[1] https://github.com/MJITG/PyTorch-HITNet-Hierarchical-Iterative-Tile-Refinement-Network-for-Real-time-Stereo-Matching

[2] https://github.com/meteorshowers/X-StereoLab

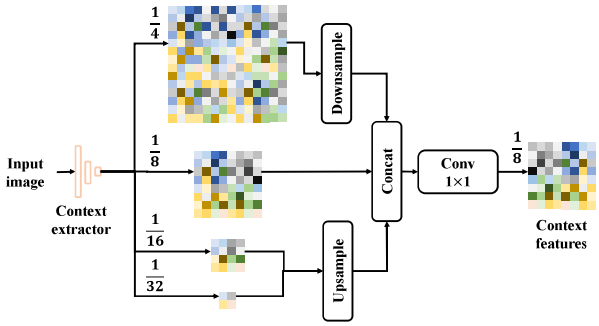

Figure 1: Overview of context extractor.

Once inter-frame motion $\boldsymbol{T}$ is recovered, we use differentiable rendering [10] to forward-project all feature maps [6] from previous frame to current frame, which is equivalent to Lagrangian flow where pixels of past frame are *pushed* to current frame. This is in contrast to Eularian flow, where pixels are *pulled* [1], which is proposed in Spatial Transformer [5] and implemented as the "grid_sample" function in PyTorch [9].

We evaluate the accuracy of the transformation map $\boldsymbol{T}$ of our motion network because accurate motion prediction is critical to our process. We compare the performance of our motion network and RAFT3D in Tab. 1. To quantify accuracy of $\boldsymbol{T}$, we report the flow EPE (FEPE) on optical flow FEPE^{of} and scene flow FEPE^{sf}, and threshold metrics of 1px following [13]. We evaluate the performance in two settings, 1) taking the ground truth disparity as input and 2) taking the noisy stereo disparity estimates as input. For fair comparison, we re-train RAFT3D using our training setup and report its result. CODD performs better than RAFT3D in both FEPE^{of} and FEPE^{sf} regardless of the input type, with only 1/5-th of the parameters of RAFT3D.

A.3. Fusion Network

The fusion network uses a set of input cues to determine the $\mathbf{w}_{\text{reset}}$ and $\mathbf{w}_{\text{fusion}}$. For appearance correlation, we project the input features from stereo network to a feature dimension of 32. The fusion weight is then regressed at $1/4$ resolution for better context awareness while the reset weight is regressed at the full resolution for better outlier rejection.

Table 1: Ablation study of the transformation prediction $\boldsymbol{T}$ of the motion network. GT: ground truth disparity as input. S: stereo network disparity estimates as input. ‡: official checkpoint.

	FEPE^{of}	$\delta^{\text{of}}_{1\text{px}}$	FEPE^{sf}	$\delta^{\text{sf}}_{1\text{px}}$	Parameters
RAFT3D (GT) [13] ‡	2.145	0.131	2.177	0.138	45.0M
RAFT3D (GT) [13]	1.808	0.133	1.847	0.141	45.0M
Motion (Ours, GT)	**1.754**	**0.127**	**1.793**	**0.135**	**8.5M**
RAFT3D (S) [13]	2.458	0.149	2.514	0.158	45.0M
Motion (Ours, S)	**1.902**	**0.134**	**1.949**	**0.142**	**8.5M**

Table 2: Results with different stereo networks on FlyingThings3D [7].

	TEPE	$\delta^{\text{t}}_{3\text{px}}$	TEPE_{r}	$\delta^{\text{t}}_{100\%}$	EPE	$\delta_{3\text{px}}$
STTR	0.482	**0.014**	11.434	0.374	**0.449**	**0.014**
STTR + CODD	**0.448**	0.017	**10.109**	**0.290**	0.454	**0.014**
PSMNet	1.371	0.056	35.136	0.466	1.079	0.045
PSMNet + CODD	**1.266**	**0.052**	**31.958**	**0.354**	**1.052**	**0.044**
GwcNet	0.959	0.041	22.598	**0.409**	0.752	**0.032**
GwcNet + CODD	**0.924**	**0.039**	**19.953**	0.402	**0.726**	**0.032**

B. Qualitative Visualization

B.1. Space-time volume

Similar to [18], we visualize the network's output by concatenating the predictions over the temporal axis to build a space-time volume on the MPI Sintel dataset. We then take a slice along the horizontal axis and visualize the depth prediction over time in Fig. 2. Even though this does not trace the same object point over time, it gives insights of how stable the model's prediction temporally. We find that CODD contains less high frequency variation compared to the per-frame model.

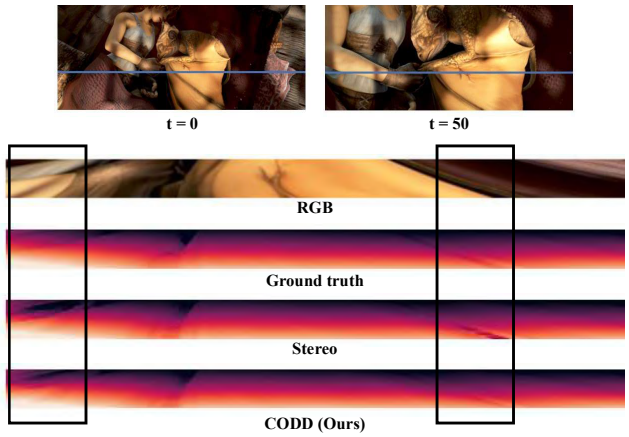

Figure 2: Space-time volumes created from MPI Sintel dataset [2]. (a)-(b) RGB images of the start and end of the video sequence. The blue line indicates the horizontal slice taken. (c) RGB of the space-time volume slice. (d)-(f) Disparity of the space-time volume slice. The high frequency noise of the stereo prediciton is successfully removed with our CODD framework, as highlighted in the black boxes.
© copyright Blender Foundation | durian.blender.org

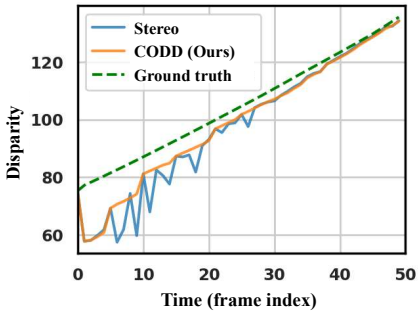

Figure 3: Disparity when tracing the same 3D point across time.

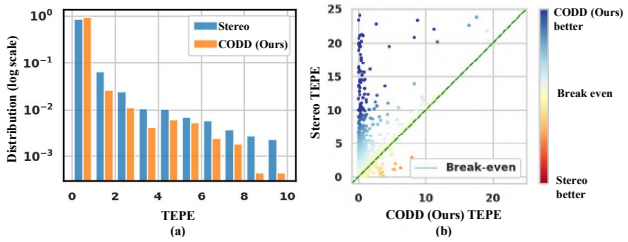

Figure 4: (a) Distribution of TEPE. (b) Pixel-wise improvement comparison. Diagonal line indicates break-even.

B.2. Improvements over Per-frame Stereo

We visualize the improvements over stereo qualitatively in Fig. 3. As ground truth disparity constantly increases, the stereo predictions vary frequently. By incorporating temporal information, CODD removes many of the temporal jitter of the stereo prediction and outputs more consistent and accurate estimates.

We plot the distribution of TEPE comparing our and stereo's results in Fig. 4. We find that CODD successfully reduce TEPE across different magnitudes as shown in Fig. 4a, suggesting more consistent predictions across time. To understand how the performance of each pixel has changed instead of the whole image, we additionally plot the TEPE of each pixel of stereo network and CODD in Fig. 4b with one-to-one correspondence, where the diagonal line indicates break-even (*i.e.*, TEPE of both networks are the same), top left region indicates that CODD successfully reduces the TEPE, and bottom right region indicates that CODD makes the TEPE larger. As most of the plotted points are clustered in the top left region, *i.e.*, our TEPE is smaller than stereo network TEPE, CODD reduces TEPE of the stereo network for the majority of the pixels. We note that even in the cases where the TEPE of stereo network is large (y-axis), CODD can reduce the error close to zero (x-axis), suggesting large improvements in terms of temporal consistency over the stereo network for these cases.

C. Additional Experiments

We include additional experiments that sheds light on CODD in the following sections.

C.1. Using Different Stereo Networks

We present additional experiments on FlyingThings3D [7] to demonstrate that our framework works with different stereo networks. We follow the same experiment setup as Sect.5.2 and use other stereo networks to produce the per-frame disparity estimates. Tab. 2 summarizes our findings, which are consistent with our findings for HITNet. We note that all metrics improves with the introduction of our motion and fusion networks, with the only exceptions of STTR $\delta_{\mathrm{t}}^{3\mathrm{px}}$, EPE and GwcNet $\delta_{100\%}^{\mathrm{t}}$. The results suggest that our proposed framework in theory should be applicable to other stereo networks.

C.2. End-to-End Fine-tuning

We note that our framework is designed to be end-to-end differentiable. Thus, during fine-tuning, we can in theory directly fine-tune all components together, instead of first fine-tuning the stereo network and then the other components. However, due to the gradients from different losses back-propagating to the stereo network, in comparison to only the stereo losses in the case of per-frame stereo network, end-to-end fine-tuning may not be a fair comparison. Thus, we report these additional results in appendix.

Denoting the loss of stereo network as ℓ_{S}, the loss of motion network as ℓ_{M}, and the loss of fusion network as ℓ_{F}, during end-to-end fine-tuning, the total loss is the weighted sum of all sub-network losses:

$$\ell_{\mathrm{E2E}} = \alpha_{\mathrm{S}}\ell_{\mathrm{S}} + \alpha_{\mathrm{M}}\ell_{\mathrm{M}} + \alpha_{\mathrm{F}}\ell_{\mathrm{F}}\,. \tag{5}$$

We use $\alpha_{\mathrm{M}} = 0.5$ while keeping $\alpha_{\mathrm{S}} = \alpha_{\mathrm{F}} = 1.0$ in our experiments to balance the losses. A batch size of 16 is used for end-to-end fine-tuning due to memory constraint. All learning rates are linearly decayed from $2\mathrm{e}{-5}$ following the learning rate of the per-frame stereo model.

The results on different benchmarks are summarized in Tab. 3, where the end-to-end training (E2E) result is comparable to the stage-wise fine-tuning results in Sect.5.3. Other than TartanAir, end-to-end fine-tuning leads to better results. In either training setup, our results are favorable compared to the per-frame results. The results suggest that our framework is not sensitive to the specific training strategies we have used.

C.3. Zero-shot Generalization

Dataset We report zero-shot generalization results of our pre-trained network from FlyingThings3D on the TartanAir, KITTI Depth and MPI Sintel dataset finalpass [2]. MPI Sintel contains animated characters with deformation. It contains 23 video sequences, totaling 1064 images.

Table 3: End-to-end fine-tuning results. Ours: Fine-tune stereo network first and then freeze its parameters. Ours (E2E): end-to-end fine-tuning.

		TEPE	δ^t_{3px}	$TEPE_r$	$\delta^t_{100\%}$	EPE	δ_{3px}
Tartan Air dataset [16]	Stereo [11]	0.876	0.053	9.039	0.339	0.934	0.055
	CODD	**0.751**	**0.045**	**6.206**	**0.240**	**0.904**	0.053
	CODD (E2E)	0.763	0.047	6.976	0.270	0.905	**0.052**
KITTI Depth dataset [14]	Stereo [11]	0.289	**0.001**	3.630	0.156	0.423	0.004
	CODD	0.258	**0.001**	2.764	0.132	0.418	**0.003**
	CODD (E2E)	**0.251**	**0.001**	**2.408**	**0.129**	**0.409**	**0.003**
KITTI 2015 dataset [8]	Stereo [11]	0.570	0.026	10.672	0.126	0.811	0.033
	CODD	0.507	**0.022**	8.740	0.112		
	CODD (E2E)	**0.505**	0.024	**8.132**	**0.111**	**0.806**	**0.032**

Table 4: Zero-shot generalization experiments on MPI Sintel [2] datasets. All networks are trained only on FlyingThings3D [7].

	TEPE	δ^t_{3px}	$TEPE_r$	$\delta^t_{100\%}$	EPE	δ_{3px}
Stereo [11]	2.621	0.127	298.674	0.515	5.028	0.199
Kalman Filter [17]	2.583	0.126	287.456	0.460	5.027	0.199
CODD (Ours)	**2.270**	**0.092**	**191.445**	**0.439**	**5.009**	0.199

Results As shown in Tab. 4, our model is able to generalize well onto a new data domain and significantly improves over the per-frame stereo [11] and Kalman filter [17] in terms of temporal metrics, up to 36% ($TEPE_r$: from 298.674 to 191.445). CODD also outperforms the competing approaches in terms of EPE.

C.4. Additional Fusion Network Ablation

We provide additional ablation experiments to further understand how different design choices may affect the performance in Tab. 5.

Correlation window We experiment with different correlation window size W to see the effect of increasing receptive field. We increase the correlation window size of correlation from 3 to 5 or 7, with a constant dilation of 2. We do not observe significant benefits of increasing the window size. The result suggests that a window size of 3 with dilation 2 already provides enough information for the network. Our final model uses a window size of 3 for efficiency.

Correlation type In addition to our proposed pixel-to-patch correlation, we also experiment with: 1) pixel-to-pixel correlation, where correlation values are obtained pixel-wise, and 2) patch-to-patch correlation, where correlation values are computed pixel-wise over the whole patch. We find that pixel-to-patch correlation performs best across all metrics.

D. Dataset Details

For long sequences (several thousands), we chunk the video into sub-sequences of 50 frames following [2].

Table 5: Additional ablation studies of fusion network. Underline: the base configuration.

		TEPE	δ^t_{3px}	$TEPE_r$	$\delta^t_{100\%}$	EPE	δ_{3px}
Correlation window W	3	0.756	0.035	15.013	0.211	0.604	0.029
	5	0.756	0.035	15.005	0.210	0.603	0.029
	7	0.756	0.035	15.014	0.210	0.603	0.029
Correlation Type	pixel-to-patch	0.756	0.035	15.013	0.212	0.604	0.029
	pixel-to-pixel	0.758	0.036	15.103	0.211	0.610	0.030
	patch-to-patch	0.759	0.035	15.134	0.210	0.609	0.030

Table 6: Comparison of training schemes for stereo depth network on the FlyingThings3D datset [7].

	TEPE	$TEPE_r$	EPE
default training	**0.821**	**16.450**	**0.609**
multi-frame training	0.998	19.199	0.715

FlyingThings3D: We follow the official split information.

TartanAir: We validate on the *seasonforest* sequence and test on the *carwelding* sequence.

KITTI Depth [14][3]: We follow the official split on the KITTI Depth dataset, except the *People* scene for training due to little variation. We use the last training sequence *2011_10_03_drive_0042_sync* for validation. Due to the missing ground truth optical flow information, we use an off-the-shelf network to generate the optical flow information [12] and use forward-backward flow consistency to determine flow occluded regions and outliers. The disparity change is then computed from optical flow.

KITTI 2015 [8][4]: We use a 5-fold split that splits the data into train, validation, test data.

E. Training Stereo Depth Network in Multi-frame Setting

One of the simplest idea we have tried to improve temporal consistency of stereo depth network is to train stereo networks in a multi-frame setting instead of randomly sampling images from the whole dataset. For fair comparison, we reduce the cropping height by two times and insert a subsequent frame from the same video sequence, such that the total number of pixels that the network sees is the same. However, we find this does not perform well compared to original training scheme as shown in Tab. 6, which may be attributed to the reduced diversity in the sample size.

F. Optical Flow as Alignment Mechanism

One alternative to our motion network is to predict an inter-frame optical flow (*i.e.* without disparity change compared to scene flow) to provide past information. However,

[3] http://www.cvlibs.net/datasets/kitti/eval_depth_all.php
[4] http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php

since depth is not translation invariant, the past information can only provide past trending information instead of directly aggregated. As a proof-of-concept, we replace the motion network with the ground truth optical flow provided to test the optical flow idea. However, we find that the temporal stability often doesn't benefit from the past information, and in fact, becomes worse on the FlyingThings3D dataset.

References

[1] Gualtiero Badin and Fulvio Crisciani. Variational formulation of fluid and geophysical fluid dynamics. *Mech. Symmetries Conservation Laws*, 2018. 2

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 2, 3, 4

[3] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 2

[6] Zhaoshuo Li, Amirreza Shaban, Jean-Gabriel Simard, Dinesh Rabindran, Simon DiMaio, and Omid Mohareri. A robotic 3d perception system for operating room environment awareness. *arXiv preprint arXiv:2003.09487*, 2020. 2

[7] 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. 1, 2, 3, 4

[8] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2:427, 2015. 4

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 1, 2

[10] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 2

[11] Vladimir Tankovich, Christian Hane, Yinda Zhang, Adarsh Kowdle, Sean Fanello, and Sofien Bouaziz. Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14362–14372, 2021. 1, 4

[12] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 4

[13] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021. 1, 2

[14] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017. 4

[15] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 1

[16] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 4

[17] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995. 4

[18] Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. Consistent depth of moving objects in video. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021. 2