RANGEINET: FAST LIDAR POINT CLOUD TEMPORAL INTERPOLATION

Lili Zhao[⋆]*, Xuhu Lin*[⋆]*, Wenyi Wang*[⋆]*, Kai-Kuang Ma*[†]*, Jianwen Chen*[⋆]

[⋆]University of Electronic Science and Technology of China, Chengdu, China
[†]Nanyang Technological University, Singapore

ABSTRACT

Due to the low scan rate of LiDAR sensors, LiDAR point cloud streams usually have a low frame rate, which is far below that of other sensors such as cameras. This could incur frame rate mismatch while conducting multi-sensor data fusion. LiDAR point cloud temporal interpolation aims to synthesize the non-existing intermediate frame between input frames to improve the frame rate of point clouds. However, the existing methods heavily depend on 3D scene flow or 2D flow estimation, which yield huge computational complexity and obstacles in real-time applications. To resolve this issue, we propose a fast and non-flow involved method, which analyzes the LiDAR point cloud by exploiting its corresponding 2D range images (RIs). Specifically, we develop a Siamese context extractor containing asymmetrical convolution kernels to learn the shape context and spatial feature of RIs, and the 3D space-time convolutions are introduced to precisely capture the temporal characteristics. Experimental results have clearly shown that our method is much faster than the state-of-the-art LiDAR point cloud temporal interpolation methods on various datasets, while delivering either comparable or superior frame interpolation performance.

***Index Terms*—** LiDAR, point cloud, temporal interpolation, frame interpolation.

1. INTRODUCTION

Light detection and ranging (LiDAR) sensors have become widely-used in many applications such as self-driving cars, robots, drones. LiDAR sensors can acquire high-precision three-dimensional (3D) representations of the environment in term of 3D point clouds. The temporal interpolation aims to increase the frame rate of a point cloud sequence by generating the non-existing intermediate frame between consecutive frames. It can be encountered in many applications, including frame prediction in point cloud compression [1], sensor data synthesis for self-driving simulation [2], and frame rate upsampling for multi-sensor data fusion [3]. Especially, the multi-sensor data fusion is often encountered in cars usually equipped with a suite of sensors, and there exists the incorrect time synchronization between LiDAR sensors (typically 10 Hz to 20 Hz) and cameras (typically above 30 Hz). LiDAR point cloud temporal interpolation can overcome this issue by improving the frame rate of point clouds. In addition, point clouds with a high frame rate can improve the related perception performance, such as 3D object tracking, 3D key point detection and multi-frame ICP [4].

Due to its practical importance, some algorithms are proposed for increasing the frame rate of LiDAR point clouds [4–6], which can be classified into two categories: the 3D space-based method [4] and 2D space-based methods [5, 6]. In [4], the frame interpolation is conducted in 3D space by utilizing the existing 3D scene flow estimation method (FlowNet3D [7]), along with their proposed adaptive point fusion module. However, the operations in 3D domain introduce high computational complexity and incur severe problems in applications demanding real-time response. To address this drawback, our previous works [5, 6] exploited the 2D range image (RI) representation of point clouds, and modified the existing optical flow estimation method (PWC-lite [8]) with the warping technique used in video frame interpolation for LiDAR point cloud frame interpolation. However, these methods, in principle, did not consider the fact that the spatial-temporal characteristics of RIs greatly differ from those of the typical videos. Specifically, the RI is not an intensity image, but a one-channel image with the pixel value representing the distance (depth). Therefore, the pixel value distribution of RIs is different from that of typical images. In the temporal domain, consider that the LiDAR sensor is normally mounted on a moving vehicle (e.g., cars) that usually moves in a fast speed, the frames of a RI sequence in temporal dimension are relatively discontinuous compared with common videos. All these major differences between the RI sequence and a common video may cause inaccurate interpolated results, if the LiDAR point cloud interpolation algorithm follows the principles in typical video frame interpolation. Moreover, the aforementioned methods [4–6] heavily depend on the quality of 3D scene flow or 2D flow estimation, which are often time-consuming and sometimes not accurate in occlusion or blur regions.

To address these limitations, we propose a fast temporal interpolation approach for LiDAR point clouds, namely *RangeINet*. For reducing complexity, the input of *RangeINet* is the 2D RIs generated by a spherical projection from the full LiDAR scan. Experimental results on the KITTI and

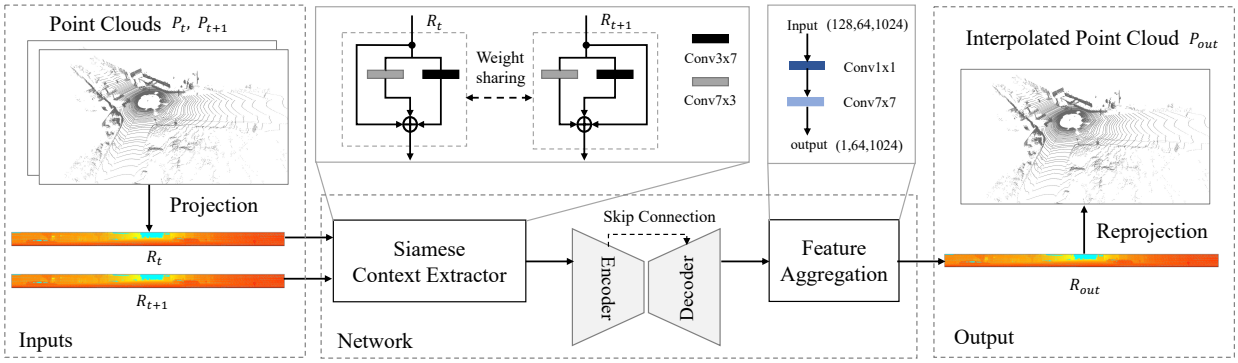

Fig. 1. The overall architecture of the proposed LiDAR point cloud temporal interpolation network: *RangeINet*.

Table 1. The decoder architecture of our encoder-decoder.

Layer	3D Convolution	Decoder
1	Conv3D	$[3 \times 3 \times 3, 256]$
2	ConvTranspose3D	$[3 \times 4 \times 4, 128]$
3	ConvTranspose3D	$[3 \times 4 \times 4, 64]$
4	Conv3D	$[3 \times 3 \times 3, 64]$
5	ConvTranspose3D	$[3 \times 4 \times 4, 64]$

nuScenes datasets show that the proposed network *RangeINet* is comparable or outperforms other state-of-the-art LiDAR point cloud frame interpolation methods, in terms of point-wise accuracy while requiring less computation time.

2. THE PROPOSED METHOD

2.1. Problem Formulation

In this paper, the consecutive two frames of point clouds are denoted as P_t and P_{t+1}, respectively, where $t \in \mathbf{Z}$ is the frame index. They can be reversibly projected into a set of 2D RIs via the function $\Pi : \mathbf{R}^3 \mapsto \mathbf{R}^2$ [9, 10], denoted as $R_t = \Pi(P_t)$ and $R_{t+1} = \Pi(P_{t+1})$, respectively. Our goal is to synthesize the intermediate new frame R_{out} by finding the relation $\mathbf{f}$:

$$R_{out} = \mathbf{f}(R_t, R_{t+1}). \tag{1}$$

In our work, $\mathbf{f}$ is presented by a novel point cloud temporal interpolation network. As shown in Fig. 1, the proposed approach consists of three stages: 1) 2D RI generation ($\mathbf{R}^3 \mapsto \mathbf{R}^2$): projecting 3D point clouds into a set of 2D RIs; 2) conducting temporal interpolation by using our proposed network; 3) re-projecting the interpolated frame of the stage 2 into the final point cloud ($\mathbf{R}^2 \mapsto \mathbf{R}^3$).

2.2. Range Image Generation

It is well-known that the LiDAR's native range image, is a 2D compact representation for 3D sparse LiDAR point clouds. For each point (x, y, z) in LiDAR point clouds, it can be reversibly projected via the function $\Pi : \mathbf{R}^3 \mapsto \mathbf{R}^2$ into a pixel (u, v) in the 2D image coordinate as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right] w \\ \left[1 - \left(\arcsin\left(z, \rho^{-1}\right) + f_{\text{down}}\right) f^{-1}\right] h \end{pmatrix}, \tag{2}$$

where h and w represent the height and width of the projected range image, respectively. $\rho = \sqrt{x^2 + y^2 + z^2}$ is the value of the pixel (u, v), and $f = |f_{\text{down}}| + |f_{up}|$ denotes the vertical field of view (FOV) for the LiDAR sensor. Here, we set $w = 1024$ and $h = 64$ as used in [9, 10].

2.3. The Proposed Temporal Interpolation Network

Siamese Context Extractor. The proposed Siamese context extractor (SCE) contains two same sub-nets sharing weights. Each sub-net containing the asymmetrical convolutions (3×7 and 7×3) takes one of the two RIs as input, as depicted in Fig. 1 (top). The principle behind the design of SCE can be clarified from two aspects. First, the Siamese structure can jointly learn the local and global descriptions, while considering the spatial relationship of two input range images (RIs). Second, the shapes of objects (e.g., car, bus, pole, tree) in RIs are often presented in rectangle regions. The asymmetrical convolutions can better fit the point (pixel) distribution of objects to learn the shape context of RIs.

Encoder-Decoder. We follow the U-Net architecture [13] to build our encoder-decoder. Differently, all 2D convolutions are replaced by 3D space-time convolutions, which can model the complex dynamical change in the temporal dimension without any flow information. Motived by FLAVR [14], we use 18-layer ResNet-3D (R3D-18) [15] as our encoder, where the last full-connected layer is not included. In Table 1, we present the decoder structure, in which 3D transpose convolutions are used.

Feature Aggregation. The output of the encoder-decoder has a dimension of (c, t, h, w), where c denotes the output channel number, and t is the number of frames, while h and w show

Table 2. Quantitative comparisons with state-of-the-art methods for 3D LiDAR point cloud frame interpolation on the KITTI and nuScenes datasets. The average runtime of generating a frame containing the following point number is also evaluated.

Method	Inputs	KITTI [11]		nuScenes [12]		Point Number (KITTI)	Runtime (KITTI)
		SNN-RMSE↓	CD↓	SNN-RMSE↓	CD↓	per frame	s/frame
PointINet [4]	xyz+Intensity+3D scene flow	**0.220**	**0.101**	0.452	0.444	51032	0.45
RAI-Net [6]	xyz+range+2D optical flow	0.253	0.137	0.558	0.729	51404	0.09
RangeINet (Ours)	range	0.234	0.117	**0.394**	**0.362**	51403	**0.04**

the height and width of the feature map, respectively. Then, the 3D feature map is split along the temporal dimension and concatenated in the channel dimension to generate 2D feature maps, which are then followed by 2D convolutions to obtain the 2D interpolated frame.

3. EXPERIMENTAL RESULTS

3.1. Datasets

We use the KITTI odometry dataset [11] and nuSecenes [12] dataset to train and evaluate our model. These two datasets provide the point clouds acquired by 64-beam and 32-beam LiDAR sensors, respectively. The KITTI odometry dataset consists of 11 sequences (indexed by 00-10). For each sequence, following the method in [4], we downsample the frame rate of the point clouds. Then, we divide the KITTI dataset into two parts: the training set (11676 frames) and the test set (4659 frames). The frames in these two sets are randomly selected from 11 sequences covering all scenes. The nuScenes dataset provides 850 scenes, in which the first 100 scenes are selected as the training set and the others as our test set.

3.2. Implementation Details

Loss Function. Our frame interpolation network aims to generate the intermediate range image which has less differences with its corresponding ground-truth frame. Given the output of our model is R_{out} and the ground truth is R_{gt}, we use ℓ_1 norm for the loss:

$$\mathcal{L}_1 = \|R_{out} - R_{gt}\|_1 . \tag{3}$$

The ℓ_2 norm is also a choice, but it has been proven that ℓ_2 norm could incur blurry results for image synthesis [16]. We also train our model by ℓ_2 norm, whose results are provided in ablation studies.

Training Strategy. We first train our model using the training set from the KITTI dataset for 40 epochs by using Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as optimizer. The initial learning rate is set to 1×10^{-4}, which is reduced by a factor of 0.5 while encountering the training plateau. The batch size is 16. Afterwards, we fine-tune the model using the training set from nuScenes dataset for another 20 epochs. It took about 6 days to converge on an NVIDIA GTX 1080Ti GPU.

3.3. Evaluation Metrics

The difference between the interpolated point cloud and the ground truth one is evaluated by the commonly-used evaluation metrics for point clouds: symmetric nearest neighbor root mean square error (SNN-RMSE) [6] and Chamfer distance (CD) [17]. It should be noted that our method may interpolate a frame with different point number from that of the ground-truth one, and therefore the Earth mover's distance (EMD) [4] is not considered in this paper. Given two point clouds P_A and P_B ($P_A, P_B \subseteq \mathbf{R}^3$) with N_A and N_B points respectively, the SNN-RMSE and CD can be computed as follows.

Chamfer distance (CD). The CD between P_A and P_B is

$$\mathcal{L}_{\text{CD}} = \frac{1}{N_A} \sum_{a \in P_A} \min_{b \in P_B} \|a - b\|_2^2 + \frac{1}{N_B} \sum_{b \in P_B} \min_{a \in P_A} \|b - a\|_2^2. \tag{4}$$

The first term represents the sum of the minimum distances for any point a in P_A with respect to its nearest neighbor b in P_B , and the second term represents the sum of the minimum distances for any point b in P_B with respect to its nearest neighbor a in P_A .

SNN-RMSE. The mean squared error (MSE) between P_A, P_B is first computed by calculating the Euclidean distance from each point $a \in P_A$ to its nearest point $b \in P_B$ via KD-tree. Similarly, the distance from each point $b \in P_B$ to its nearest point $a \in P_A$ can be computed as well. Lastly, SNNR-RMSE can be computed by

$$\mathcal{L}_{\text{RMSE}} = \sqrt{\frac{1}{2}\left(\frac{1}{N_A} \sum_{a \in P_A} (a-b)^2 + \frac{1}{N_B} \sum_{b \in P_B} (b-a)^2\right)}. \tag{5}$$

3.4. Results and Analysis

3.4.1. Comparison with the state-of-the-art LiDAR point cloud frame interpolation methods

We compare our method against the existing state-of-the-art learning-based LiDAR point cloud temporal interpolation methods: the 3D space-based PointINet [4] and the 2D space-based RAI-Net [6]. Note that the work [5] is the flow estimation module adopted in RAI-Net [6], and therefore it is not taken into consideration for comparison. In Table 2, we compare the performances on the KITTI odometry dataset and nuScenes dataset. From the perspective of accuracy, it

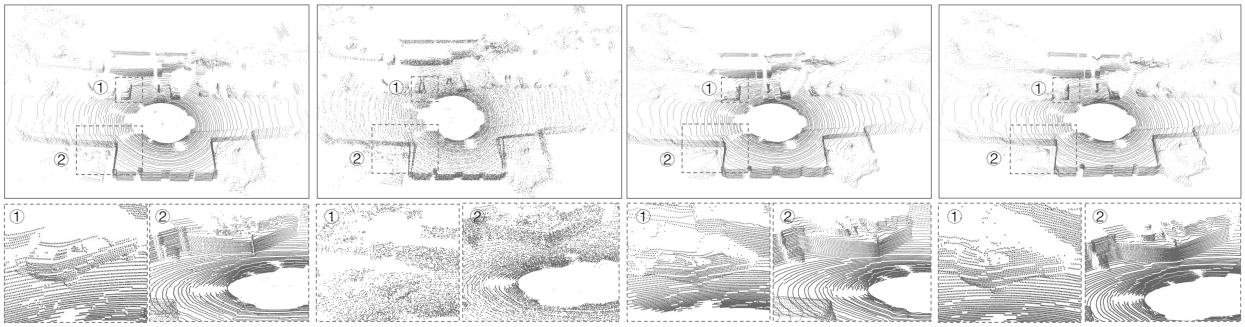

Fig. 2. Visual comparisons on the KITTI dataset among LiDAR point cloud frame interpolation methods.

Table 3. Quantitative comparisons with video frame interpolation (VFI) methods, in which 2D range images are inputs.

Method	Category	L1 loss↓
SuperSlomo [18]	flow-based VFI	1.564
DAIN [19]	flow-based VFI	1.948
AdaCoF [20]	kernel-based VFI	3.914
FLAVR [14]	flow-agnostic VFI	2.196
RangeINet (Ours)	point cloud frame interpolation	**0.930**

Table 4. Impacts of the Siamese context extractor (SCE), loss function and kernel sizes in our model.

Model	SNN-RMSE↓	CD↓
w/o SCE	0.243	0.127
Conv 1x3 in SCE	0.301	0.195
Conv 3x5 in SCE	0.312	0.209
L2 loss	0.303	0.198
The full network	**0.234**	**0.117**

can be found that our method outperforms RAI-Net [6] for all test datasets. Compared with PointINet [4], our method delivers a comparable performance on KITTI dataset and a better performance on nuScenes dataset.

From the perspective of the computational cost, the inference is made by using a NVIDIA GTX 1080Ti GPU. Table 2 shows the runtime of generating an interpolated frame containing 51400 points for the KITTI odometry dataset. It is worth noting that our proposed method is more than ten times faster than PointINet [4], while two times faster than RAI-Net [6]. This can be explained by the fact that PointINet [4] and RAI-Net [6] require additional 3D scene flow and 2D optical flow estimation, respectively, while our method does not need any flow information and only uses the range value as input. Thus our method provides the possibility of developing a real-time temporal interpolation algorithm. In addition, Fig. 2 shows the visual interpolation results from different methods. It can be found that our method deliver better results, especially for object details and sharp edges.

3.4.2. Comparison with video frame interpolation methods

Given that one step of our algorithm is to generate the interpolated 2D RI, we also compare the proposed method on the projected 2D RIs against several representative video frame interpolation methods: SuperSloMo [18], DAIN [19], AdaCoF [20], and FLAVR [14]. All methods are fine-tuned by our training set. It should be noted the pixel value in RIs is the distance and in the 32-bit floating-point precision with only one channel. We aims to get the high precision other than pleasant human vision experience. Therefore, we adopt L1 loss as the evaluation metric. From Table 3, it can be observed that our method delivers the most accurate frame interpolation results.

3.5. Ablation Studies

In Table 4, we present the ablation studies on the KITTI odometry dataset in terms of the Siamese context extractor (SCE), the kernel sizes in SCE, and the loss functions. We evaluate the performance with SNN-RMSE and Chamfer distance. Table 4 proves the effectiveness of different modules in our network.

4. CONCLUSION

In this work, we present a novel and fast temporal interpolation framework for LiDAR point clouds, By our method, LiDAR point clouds can be upsampled to a high frame rate. Our main contribution is a non-flow involved network which exploits 2D RIs as the immediate representation, and introduces the Siamese context extractor to learn the spatial feature distribution and 3D space-time convolutions to model temporal dynamics of RIs, without any flow information. Quantitative experimental results have shown that our proposed scheme can deliver comparable or superior results to the existing methods with less runtime.

5. ACKNOWLEDGEMENTS

This work was supported in part by the Sichuan Science and Technology Program (2019YJ0190, 2020YFG0149), and in part by the NTU-WASP Joint Project under Grant M4082184.

6. REFERENCES

[1] Y. Xu, W. Zhu, Y. Xu, and Z. Li, "Dynamic point cloud geometry compression via patch-wise polynomial fitting," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process (ICASSP)*, May 2019, pp. 2287–2291.

[2] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar, "SurfelGAN: Synthesizing realistic sensor data for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020.

[3] T. N. Mundhenk, K. Kim, and Y. Owechko, "Frame rate fusion and upsampling of EO/LIDAR data for multiple platforms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 762–769.

[4] F. Lu, G. Chen, S. Qu, Z. Li, Y. Liu, and A. Knoll, "PointINet: Point cloud frame interpolation network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 3, May 2021, pp. 2251–2259.

[5] X. Guo, X. Lin, L. Zhao, Z. Zhu, and J. Chen, "An unsupervised optical flow estimation for LiDAR image sequences," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sept. 2021, pp. 2613–2617.

[6] L. Zhao, Z. Zhu, X. Lin, X. Guo, Q. Yin, W. Wang, and J. Chen, "RAI-Net: Range-adaptive LiDAR point cloud frame interpolation network," in *IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Aug. 2021, pp. 1–6.

[7] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 529–537.

[8] L. Liu, J. Zhang, R. He, Y. Liu, Y. Wang, Y. Tai, D. Luo, C. Wang, J. Li, and F. Huang, "Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2020, pp. 6489–6498.

[9] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet ++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 4213–4220.

[10] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 1–19.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.

[12] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11 618–11 628.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation." in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*, Oct. 2015.

[14] T. Kalluri, D. Pathak, M. Chandraker, and D. Tran, "FLAVR: Flow-agnostic video representations for fast frame interpolation," *arXiv preprint arXiv:2012.08512v2*, Apr. 2021.

[15] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6450–6459.

[16] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016.

[17] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2463–2471.

[18] H. Jiang, D. Sun, V. Jampani, M. Yang, and J. Kautz, "Super SloMo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 9000–9008.

[19] W. Bao, W. Lai, C. Ma, X. Zhang, Z. Gao, and M. Yang, "Depth-aware video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3703–3712.

[20] H. Lee, T. Kim, T. Chung, D. Pak, Y. Ban, and S. Lee, "AdaCoF: adaptive collaboration of flows for video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5316–5325.