ISSN 1213-6972 (print)
ISSN 1213-6964 (online)
Journal of WSCG
http://www.wscg.eu
Vol.29, No-1-2, 2021

FastRIFE: Optimization of Real-Time Intermediate Flow Estimation for Video Frame Interpolation

Malwina Kubas
Warsaw University of Technology
Faculty of Electrical Engineering
ul. Koszykowa 75
00-662 Warszawa, Poland
malwina.kubas.stud@pw.edu.pl

Grzegorz Sarwas
Warsaw University of Technology
Faculty of Electrical Engineering
ul. Koszykowa 75
00-662 Warszawa, Poland
grzegorz.sarwas@pw.edu.pl

Abstract

The problem of video inter-frame interpolation is an essential task in the field of image processing. Correctly increasing the number of frames in the recording while maintaining smooth movement allows to improve the quality of played video sequence, enables more effective compression and creating a slow-motion recording. This paper proposes the FastRIFE algorithm, which is some speed improvement of the RIFE (Real-Time Intermediate Flow Estimation) model. The novel method was examined and compared with other recently published algorithms. All source codes are available at:
https://gitlab.com/malwinq/interpolation-of-images-for-slow-motion-videos.

Keywords

Video Interpolation, Flow Estimation, Slow-Motion

1 INTRODUCTION

Video frame interpolation is one of the most important issues in the field of image processing. Correctly reproduced or multiplied inter-frame information allows its use in a whole range of problems, from video compression [Wu18], through improving the quality of records, to generating slow-motion videos [Men20] or even view synthesis [Fly16]. Mid-frame interpolation performed in real-time on high-resolution images also increases the image's smoothness in video games or live broadcasts. With fast and accurate algorithms, we can reduce the costs associated with the construction of high-speed video cameras or provide services to users with limited hardware or transmission resources.

Algorithms of this class should deal with complex, nonlinear motion models, as well as with changing parameters of the real vision scene, such as shadows, changing the color temperature, or brightness. Conventional approaches are based on motion prediction and compensation [Haa93; Bao18], which are used in various display devices [Wu15].

Motion estimation is used to determine motion vectors in a block or pixel pattern between two frames. Block-based methods [Haa93] assume that pixels in a block have the same inter-frame shift and use search strategies [Zhu00; Gao00], and selection criteria [Haa93; Wan10] to obtain the optimal motion vector. Other methods of movement estimation are optical flow algorithms that estimate the motion vector for each pixel separately, which makes them more computationally expensive. In recent years there has been noticed significant progress in this area. New algorithms have been developed based on the optimization of variance [Bro04], searching for the nearest neighbors [Che13], filtering the size of costs [Xu17] and deep convolutional neural networks (CNN) [Ran17; Dos15]. However, very often, the quality of the obtained interpolations is burdened with an increase in the required computing resources and often limits their use in the case of users working on standard equipment or mobile phones.

Flow-based video frame interpolation algorithms achieve very good results [Jia18; Nik18; Bao19; Xue19; Hua21]. These solutions include two steps: warping the input frames according to approximated optical flows and fusing and refining the warped frames using CNN. Flow-based algorithms can be divided into forward warping methods and more often used back-warping methods. A common operation of these methods is to calculate bidirectional optical flow to generate an intermediate flow. These operations require a large number of computing resources and are not suitable for real-time scenarios.

DOI:10.24132/JWSCG.2021.29.3

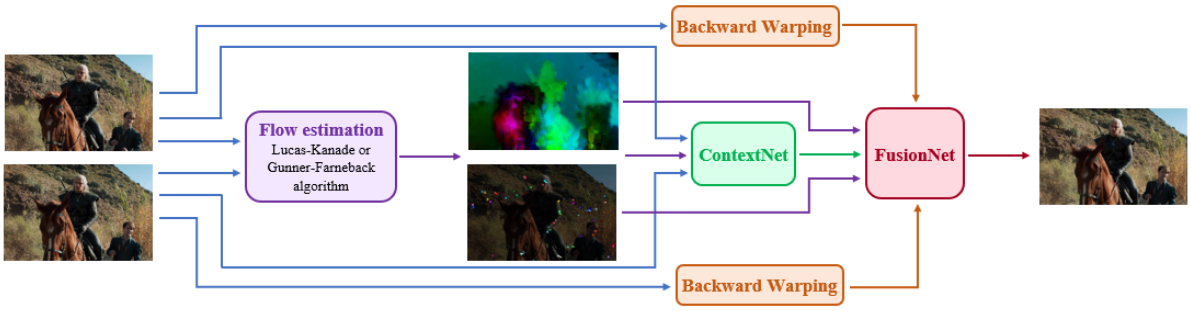


Figure 1: Architecture of the proposed video frame interpolation algorithm. We estimate the optical flow with analytical methods and use fine-tuned neural network models from RIFE: ContextNet and FusionNet.

This work aims to create an efficient video frame interpolation model, which can be implemented on many devices and run in real-time. Authors will compare state-of-the-art techniques, best in image interpolation field at the moment, and propose an improvement in terms of execution time in one of them. Algorithms which will be analyzed and compared are: MEMC-Net [Bao21], SepConv [Nik17], Softmax Splatting [Nik20], DAIN [Bao19] and RIFE [Hua21], all using different approaches.

2 RELATED WORK

The topic of video frame interpolation has been widely discussed in the research papers [Kim14; Dos15; Nik17; Bao21; Jia18]. The first groundbreaking method was proposed in [Lon16]. Long *et al.* were the first to use the convolutional neural network, which aimed to estimate the result frame directly. Next generations of image interpolation algorithms brought significantly better results, each using more extensive neural network structures.

Some algorithms, like SepConv [Nik17], were built and trained end-to-end using only one pass through the encoder-decoder structure. However the majority of methods use an additional sub-networks, of which the most important is the optical flow estimation. Having the information about pixels transition allows generating more accurate results and boost benchmark metrics. This approach was used e.g. in the MEMC-Net method, which combines both motion estimation, motion compensation and post-processing by four sub-networks [Bao21].

Bao *et al.* composed an improvement of MEMC-Net model called DAIN (Depth-Aware INterpolation) [Bao19], where the information of pixels depth was used, improving dealing with the problems related to occlusion and large object motion by sampling closer objects better. Depth estimation is one of the most challenging tasks in the image processing field of study, so the whole process takes a noticeable amount of time. Authors used a fine-tuned model called hourglass [Che16] network learned on the MegaDepth dataset [Li18]. DAIN method estimates the bi-directional optical flow using an off-the-shelf solution called PWC-Net [Xue19].

Another excellent flow-based method is Softmax Splatting [Nik20]. Niklaus and Liu proposed a solution of forward warping operator, which works conversely to backward warping and is used in most methods. Softmax Splatting uses an importance mask and weights all pixels in the input frame. Authors used fine-tuned PWC-Net for estimating optical flow, similarly to DAIN.

One of the newest algorithm is RIFE - Real-Time Intermediate Flow Estimation [Hua21], which is able to achieve comparable results on standard benchmarks when compared to previous methods but also works significantly faster. More details about this solution are presented in the next section.

3 PROPOSED SOLUTION

Most state-of-the-art algorithms generate intermediate frames by combining bi-directional optical flows and additional networks, e.g. estimating depth, which makes these methods unable to real-time work. Huang *et al.* proposed in RIFE [Hua21] a solution of simple neural network for estimating optical flows called IFNet and a network for generating interpolated frames called FusionNet.

The IFNet comprises three IFBlocks, each built with six ResNet modules and operating on increasing image resolutions. The output flow is very good quality and runs six times faster than the PWCNet and 20 times faster than the LiteFlowNet [Hui18]. After estimating the flow, coarse reconstructions are generated by backward warping and then the fusion process is performed. The process includes context extraction and the FusionNet with an architecture similar to U-Net, both consisting four ResNet blocks.

In this paper authors will analyze the RIFE model with different module for estimating optical flow. Although the IFNet gives excellent results and runs very fast, authors wanted to test analytical methods and compare the

Overlayed inputs | Optical flow by GF method | Optical flow by LK method | Interpolated frame with GF method | Interpolated frame with LK method | Ground-truth frame

Figure 2: Example of video frame interpolation results. Authors propose a simplified RIFE model with optical flow estimated using analytical methods: Gunnar-Farnebäck and Lucas-Kanade.

results of such a simplified model in terms of runtime and quality. This change can speed up the algorithm even more, making the whole process capable of running real-time on many more devices.

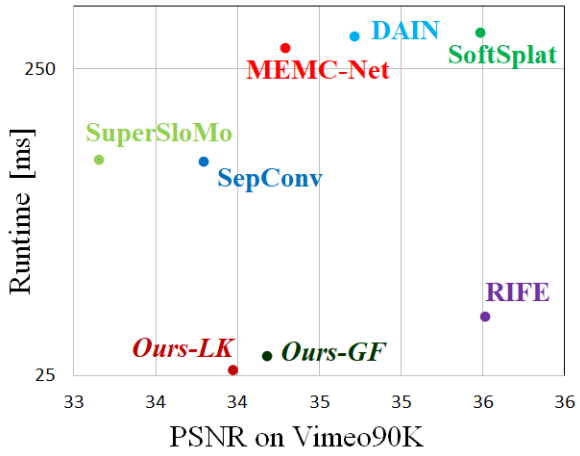


Figure 3: Speed and accuracy trade-off measured on RTX 2080 Ti GPU. Our models compared with previous frame interpolation methods: RIFE [Hua21], Softmax Splatting [Nik20], DAIN [Bao19], MEMC-Net [Bao21], SuperSloMo [Jia18] and SepConv [Nik17]. Comparison done on Vimeo90K test dataset. Please note the logarithmic runtime scale.

Authors replaced the IFNet part with analytical methods described below. Then, new FusionNet and ContextNet models were fine-tuned on the proposed solution. The full architecture is shown in Figure 1. Authors decided to test two of the most widely used algorithms, one generating dense flow and one which estimates sparse results. The example of optical flows and frame interpolation results are shown in Figure 2 and the speed/accuracy trade-off is presented in Figure 3.

3.1 Gunnar-Farnebäck algorithm

The Gunnar-Farnebäck method [Far03] gives dense results, which means that flow values are generated for every pixel. The algorithm detects changes in pixel intensity between two images using polynomial expansions and highlights pixels with the most significant changes.

The idea depends on an approximation of neighbourhood pixels in both frames by quadratic polynomials. Pixels' displacements are estimated from the differentiation of transforms of polynomials under translations. The algorithm is implemented by hierarchical convolutions what makes the process very efficient.

After the tests authors set the parameters of the Gunnar-Farnebäck (GF) method as follows:

- scale of image pyramids: 0.2,
- number of pyramid layers: 3,
- size of the pixel neighborhood: 5,
- size of averaging windows: 15×15.

The above sets give the best runtime to accuracy trade-off.

3.2 Lucas-Kanade algorithm

Dense methods can give very accurate results. However, sometimes the speed to accurate ratio might be more favorable for sparse methods. These algorithms calculate optical flow only for some number for pixels, extracted by the detector as feature points. To feed the results to the FusionNet it was needed to change the values from sparse to dense form.

Authors used Lucas-Kanade (LK) algorithm, which is the most commonly used analytical method for generating optical flow. It takes a 3×3 patch around the point and assumes all nine pixels will have the same flow. The solution is calculated with the least squares method. It is proven that the Lucas-Kanade works better with corner points, so for feature points detections authors used the Shi-Tomasi algorithm. The function calculates corner quality measure in each pixel's region using minimum eigenvalues and Harris Corner Detection.

Parameters which were used in the Lucas-Kanade algorithm are:

- size of search window: 15×15,
- number of pyramids: 1 (single level),

- termination criteria: epsilon and criteria count,

and for the Shi-Tomasi detector:

- maximum number of corners: 100,
- minimum accepted quality of corners: 0.1,
- minimum possible distance between the corners: 10,
- size of computing block: 7.

3.3 Runtime

Runtime comparison can be found in Table 1. IFNet works much faster than any other state-of-the-art method, but anyway optical flow can be obtained by analytical methods in less time. One of the important issues is that most of the standard flow methods have to run the estimation process twice if a bi-directional flow is desired. The results were measured again on RTX 2080 Ti GPU.

Method	Runtime [ms]
PWCNet	125
LiteFlowNet	370
IFNet	34
Ours - GF	**9**
Ours - LK	**7**

Table 1: Runtime comparison of optical flow estimation algorithms

4 EXPERIMENTS

This section will provide evaluation results, comparing the last and best methods with our proposal. Comparison is made based on quantitative and visual results using common metrics and datasets as well as algorithms runtime.

4.1 Learning strategy

Our solution was trained on Vimeo90K dataset with optical flow labels generated by the LiteFlowNet [Hui18] in PyTorch version [Nik19]. Authors used the AdamW optimizer, the same as used in RIFE [Hua21].

4.1.1 Loss function

To address the problem of training the IFNet module, Huang *et al.* proposed a solution of leakage distillation schema, which compares the network result with the output of the pre-trained optical flow estimation method. The training loss is a linear combination of reconstruction loss L_{rec}, census loss L_{cen} and leakage distillation loss L_{dis}. Our solution was trained using the same loss function:

$$L = L_{rec} + L_{cen} + \lambda L_{dis}, \tag{1}$$

with the weight of leakage distillation loss 10 times smaller than the two others ($\lambda = 0.1$).

4.1.2 Training dataset

Vimeo90K is a video dataset containing 73,171 frame triplets with 448×256 pixels image resolution [Xue19]. This dataset was used as a training set in all compared methods.

4.1.3 Computational efficiency

All analyzed algorithms are implemented in PyTorch [Pas17] using CUDA, cuDNN, and CuPy [Che14]. The proposed solutions was implemented using OpenCV functions, and RIFE model was fine-tuned on Nvidia RTX 2080 Ti GPU with 11GB of RAM for 150 epochs which took 15 days to converge.

4.2 Evaluation datasets

For evaluation, the following publicly available datasets have been used:

4.2.1 Middlebury

Middlebury contains two subsets: Other with ground truth interpolation results and Evaluation, which output frames are not publicly available [Bak07]. It is the most widely used dataset for testing image interpolation algorithms. The resolution of frames is 640×480 pixels.

4.2.2 UCF101

UCF101 provides 379 frame triplets from action videos with 101 categories [Soo12] [Liu17]. Images resolution is 256×256 pixels.

4.2.3 Vimeo90K

Vimeo90K was used for training, but it contains also the test subset - 3,782 triplets with 448×256 pixels image resolution [Xue19].

4.3 Metrics

The following metrics have evaluated each algorithm. PSNR (Peak Signal-to-Noise Ratio):

$$PSNR(i,j) = 10 \cdot \log_{10} \frac{255^2}{MSE(i,j)}. \tag{2}$$

SSIM (Structural Similarity, l - luminance, c - contrast and s - structure):

$$SSIM(i,j) = l(i,j) \cdot c(i,j) \cdot s(i,j). \tag{3}$$

IE (Interpolation Error) is the arithmetic average of a difference between the interpolated image and the ground truth frame:

$$IE(i,j) = \overline{|i-j|}, \tag{4}$$

where i is interpolated frame and j is the ground truth image [Wan04]. On UCF101 and Vimeo90K datasets we used PSNR and SSIM (higher values mean better results), on Middlebury Evaluation and Other we evaluated IE (lower values mean better results). This set of benchmarks is used in the majority of frame interpolation articles and datasets websites.

Figure 4: Qualitative comparison on UCF101 dataset

Method	Middlebury IE	UCF101 PSNR	UCF101 SSIM	Vimeo90K PSNR	Vimeo90K SSIM	Parameters (million)
SepConv	2.27	34.78	0.967	33.79	0.970	21.6
MEMC-Net	2.12	35.01	0.968	34.29	0.970	70.3
DAIN	2.04	34.99	0.968	34.71	0.976	24.0
SoftSplat	1.81	35.10	0.948	35.48	0.964	7.7
RIFE	1.96	35.25	0.969	35.51	0.978	9.8
Ours: FastRIFE - GF	**2.89**	**34.84**	**0.968**	**34.18**	**0.968**	**4.1**
Ours: FastRIFE - LK	**2.91**	**34.26**	**0.967**	**33.97**	**0.967**	**4.1**

Table 2: Quantitative comparison of described methods on Middlebury Other, UCF101 and Vimeo90K test

4.4 Quantitative comparison

Table 2 shows a comparison of the results obtained on datasets: Middlebury Other, UCF101 and Vimeo90K, red marks the best performance, blue marks second best score. Results of evaluation on Middlebury benchmark are not available yet.

The proposed model performs comparably to other methods, scoring very similar results in both PSNR and SSIM. The most significant gap is between the results of Interpolation Error in Middlebury Other dataset. FastRIFE generates better values than the SepConv model in UCF101 and Vimeo90K benchmarks and worse results than other algorithms, but a significant reduction is shown on the parameters column.

Evaluation of both optical flow methods gives similar results. However, the model which uses the Gunnar-Farnebäck algorithm performs slightly better in every benchmark, which means that the FusionNet behaves better with dense flow estimation.

4.5 Visual Comparison

Figure 4 shows examples of frame estimation with comparison made on the DAIN and RIFE methods. These images show dynamic scenes, which are more challenging to analyze for frames interpolation methods. The resolution of images is small (256×256), but both GF and LK methods generate acceptable results, similar to DAIN. The worst smudging is shown on RIFE results, which works better with high-resolution videos.

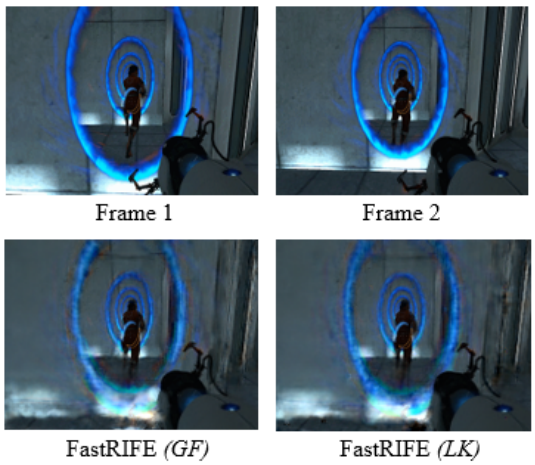


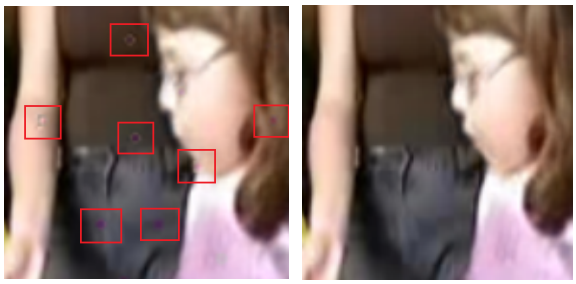
Figure 5: Visual comparison on HD frames

Method	Inference time on 448×256 image [ms]	Time of learning one epoch [h]
SepConv	125	-
MEMC-Net	293	-
SoftSplat	329	-
DAIN	320	9
RIFE	39	4.5
Ours: FastRIFE - GF	**29**	**2.4**
Ours: FastRIFE - LK	**26**	**2.4**

Table 3: Runtime comparison of described methods tested on 448×256 image (device: RTX 2080 Ti GPU)

The results of high-resolution image interpolation are shown in Figure 5. Both FastRIFE models generate very blurry frames. It is probably caused by including only small resolution videos in the training process and setting constant parameters when calculating optical flow. The issue could be repaired as future work. Other methods: DAIN and RIFE work well with HD resolution frames.

The advantage of the proposed models is the size of occupied memory. For most methods, analyzing HD images results in an out-of-memory error on our GPU, while RIFE allocates only 3 GB and FastRIFE only 1.5 GB of RAM.



FastRIFE - LK FastRIFE - GF

Figure 6: Sparse optical flow issue. Sometimes output frames suffers from spots visible in zoom

One important issue related to using the sparse optical flow method (Lucas-Kanade) is overlapping feature points as visible spots, as shown in Figure 6. The spots are noticeable after zooming small resolution images. Such a problem can be reported only for a model with the LK method. FastRIFE which uses the Gunnar-Farnebäck algorithm is this issue free.

4.6 Runtime Comparison

The usage of analytical methods and simplified RIFE model caused that FastRIFE performs favourably faster than any other state-of-the-art algorithm. Our models have 4 million parameters, which is the smallest neural network structure from all video frame interpolation methods. The runtime comparison is shown in Table 3.

Compared to any other model, except for RIFE, our solution runs up to 10 times faster. FastRIFE has slightly better benchmark results from SepConv, but runtime has been compressed from 125 to less than 30 milliseconds. The replacement of the optical flow module in RIFE saved 25% of the time needed to interpolate frames. As shown in Table 3, also the time of learning one epoch was significantly reduced.

5 CONCLUSION

In this paper, we proposed a solution to RIFE model simplification in terms of optical flow estimation. FastRIFE uses well-known analytical methods instead of additional neural network modules, which results in excellent runtime improvement with an acceptable slight drop in the quality of output frames.

Another advantage of model compression is the reduction of memory usage. Thanks to the small number of parameters, FastRIFE is able to generate output frames with the allocation of max 1.5 GB of RAM in the case of HD videos. Low memory consumption with short execution time makes the model possible to implement on many devices, including mobile phones.

The FastRIFE algorithm was analyzed with two method types for estimating optical flow: sparse (Lucas-Kanade) and dense (Gunnar-Farnebäck). Sparse flow performs faster, but better-quality results were always generated with the GF method. FastRIFE works appropriately in small resolution images but introduces prominently blurry regions when interpolating HD videos.

Future work that may be performed includes the improvement of the FastRIFE model in order to obtain better results on high resolution videos. It can probably be achieved by including HD images in the training dataset or by adjusting optical flow algorithm parameters on the fly based on video resolution or other criteria. Our proposed method can result in future research on increasing the efficiency of video frame interpolation algorithms.

REFERENCES

[Bak07] S. Baker et al. “A Database and Evaluation Methodology for Optical Flow”. In: *2007 IEEE 11th International Conference on*

Computer Vision. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4408903.

[Bao18] W. Bao et al. "High-Order Model and Dynamic Filtering for Frame Rate Up-Conversion". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3813–3826. DOI: 10.1109/TIP.2018.2825100.

[Bao19] W. Bao et al. "Depth-Aware Video Frame Interpolation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3698–3707. DOI: 10.1109/CVPR.2019.00382.

[Bao21] W. Bao et al. "MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.3 (2021), pp. 933–948. DOI: 10.1109/TPAMI.2019.2941941.

[Bro04] T. Brox et al. "High Accuracy Optical Flow Estimation Based on a Theory for Warping". In: *Computer Vision - ECCV 2004*. Ed. by T. Pajdla and J. Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 25–36. ISBN: 978-3-540-24673-2.

[Che13] Z. Chen et al. "Large Displacement Optical Flow from Nearest Neighbor Fields". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2443–2450. DOI: 10.1109/CVPR.2013.316.

[Che14] S. Chetlur et al. "CuDNN: Efficient primitives for deep learning". In: *International Journal of Computer Vision*. 2014. DOI: arXiv:1410.0759.

[Che16] W. Chen et al. "Single-Image Depth Perception in the Wild". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 730–738.

[Dos15] A. Dosovitskiy et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

[Far03] G. Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis*. 2003. DOI: 10.1007/3-540-45103-X_50.

[Fly16] J. Flynn et al. "Deep Stereo: Learning to Predict New Views from the World's Imagery". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5515–5524. DOI: 10.1109/CVPR.2016.595.

[Gao00] X. Q. Gao, C. J. Duanmu, and C. R. Zou. "A multilevel successive elimination algorithm for block matching motion estimation". In: *IEEE Transactions on Image Processing* 9.3 (2000), pp. 501–504. DOI: 10.1109/83.826786.

[Haa93] G. de Haan et al. "True-motion estimation with 3-D recursive search block matching". In: *IEEE Transactions on Circuits and Systems for Video Technology* 3.5 (1993), pp. 368–379. DOI: 10.1109/76.246088.

[Hua21] Z. Huang et al. *RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation*. 2021. arXiv: 2011.06294 [cs.CV].

[Hui18] T.-W. Hui, X. Tang, and C. C. Loy. "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[Jia18] H. Jiang et al. "Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9000–9008. DOI: 10.1109/CVPR.2018.00938.

[Kim14] U. S. Kim and M. H. Sunwoo. "New Frame Rate Up-Conversion Algorithms With Low Computational Complexity". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.3 (2014), pp. 384–393. DOI: 10.1109/TCSVT.2013.2278142.

[Li18] Z. Li and N. Snavely. "MegaDepth: Learning Single-View Depth Prediction from Internet Photos". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2041–2050. DOI: 10.1109/CVPR.2018.00218.

[Liu17] Z. Liu et al. "Video Frame Synthesis Using Deep Voxel Flow". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4473–4481. DOI: 10.1109/ICCV.2017.478.

[Lon16] G. Long et al. “Learning Image Matching by Simply Watching Video”. In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe et al. Cham: Springer International Publishing, 2016, pp. 434–450. ISBN: 978-3-319-46466-4.

[Men20] H. Men et al. “Visual Quality Assessment for Interpolated Slow-Motion Videos Based on a Novel Database”. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 2020, pp. 1–6. DOI: 10.1109/QoMEX48832.2020.9123096.

[Nik17] S. Niklaus, L. Mai, and F. Liu. “Video Frame Interpolation via Adaptive Separable Convolution”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 261–270. DOI: 10.1109/ICCV.2017.37.

[Nik18] S. Niklaus and F. Liu. “Context-Aware Synthesis for Video Frame Interpolation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1701–1710. DOI: 10.1109/CVPR.2018.00183.

[Nik19] S. Niklaus. *A Reimplementation of LiteFlowNet Using PyTorch*. https://github.com/sniklaus/pytorch-liteflownet. 2019.

[Nik20] S. Niklaus and F. Liu. “Softmax Splatting for Video Frame Interpolation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5436–5445. DOI: 10.1109/CVPR42600.2020.00548.

[Pas17] A. Paszke et al. “Automatic differentiation in PyTorch”. In: *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*. 2017.

[Ran17] A. Ranjan and M. J. Black. “Optical Flow Estimation Using a Spatial Pyramid Network”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2720–2729. DOI: 10.1109/CVPR.2017.291.

[Soo12] K. Soomro, A. R. Zamir, and M. Shah. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *International Journal of Computer Vision*. 2012. DOI: arXiv:1212.0402.

[Wan04] Z. Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[Wan10] C. Wang et al. “Frame Rate Up-Conversion Using Trilateral Filtering”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.6 (2010), pp. 886–893. DOI: 10.1109/TCSVT.2010.2046057.

[Wu15] J. Wu et al. “Enabling Adaptive High-Frame-Rate Video Streaming in Mobile Cloud Gaming Applications”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.12 (2015), pp. 1988–2001. DOI: 10.1109/TCSVT.2015.2441412.

[Wu18] C.-Y. Wu, N. Singhal, and P. Krahenbuhl. “Video Compression through Image Interpolation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Xu17] J. Xu, R. Ranftl, and V. Koltun. “Accurate Optical Flow via Direct Cost Volume Processing”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5807–5815. DOI: 10.1109/CVPR.2017.615.

[Xue19] T. Xue et al. “Video Enhancement with Task-Oriented Flow”. In: *International Journal of Computer Vision (IJCV)* 127.8 (2019), pp. 1106–1125.

[Zhu00] S. Zhu and K.-K. Ma. “A new diamond search algorithm for fast block-matching motion estimation”. In: *IEEE Transactions on Image Processing* 9.2 (2000), pp. 287–290. DOI: 10.1109/83.821744.