Literature Review: Video Frame Rate Up-Conversion (FRUC)

Naim Govani Olly Larkin Lin Hao Lee Jialong Yu Navid Zandpour Zhiyuan Zhang
Intel Video Interpolation 3rd Year Group Project
Department of Electrical and Electronic Engineering, Imperial College London.
Email: {*naim.govani17,olly.larkin17,lin.lee17,jialong.yu17,navid.zandpour19,zhiyuan.zhang17*}*@imperial.ac.uk*

***Abstract*—Video Frame Rate Up-Conversion (FRUC) utilises video frame interpolation to synthesize nonexistent frames in-between the original frames, resulting in a video of higher frame-rate. In this literature review, we discuss the conventional MEMC (motion estimation and motion compensation) FRUC model before moving on to more state-of-the-art methods. A focus is put on the implementation of these algorithms on FPGAs.**

***Index Terms*—video frame interpolation method, frame rate up conversion, image and video synthesis, video signal processing, object motion**

I. Introduction

Currently, video FRUC research is both vast and deep, however, very few resources exist collecting that information into a single place and removing the high amount of overlap between modern papers. The purpose of this review is to collate what we have deemed to be the most relevant and helpful research, specifically for an FPGA platform, so that in the future a team planning to develop a FRUC IP will have a solid foundation from which they can begin.

The earliest algorithms, explored further in Section II, are computationally simple–they can be used in real-time systems while maintaining a low resource footprint. Unfortunately, they tend to produce low-fidelity frames with a variety of issues such as very evident judder or blurred frames.

On the opposite end of the spectrum lies the more recently developed algorithms (referred in this review as "*advanced algorithms*") which incorporate neural nets into the FRUC systems [37], [46], [47], [50], [55], either as the entire system or as a subcomponent for a larger structure. These algorithms are able to interpolate very accurate frames. The issue, of course, is training these neural nets which could take a long time and once trained the sheer size of the networks makes them very resource intensive. It is doubtful that these specific algorithms will soon be adapted for FPGAs; however, their inclusion in Section V-A of this review is more of a measure of completeness to make sure that, if this problem is approached at a point in time where implementing large neural nets is tractable on FPGAs, this review would still be a helpful resource.

The third and most extensively researched option is MEMC FRUC. The core philosophy behind this approach is to measure the motion between two frames in a video and, from that information, estimate what a frame in between said frames would look like–a methodology also used by codecs to compress videos. As shown in Fig 4a, MEMC FRUC can be broken down into two steps, motion estimation (ME) and motion compensated interpolation (MCI), both of which are explored further in Section III and Section IV.

II. Basic Interpolation Methods

In this section, we discuss basic interpolation methods that are trivially implementable. To exemplify these methods, we will use 24-to-60 fps FRUC.

A. Frame repetition

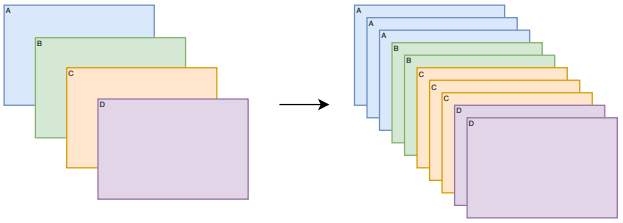

Fig. 1: Frame repetition visualisation for 24-to-60 fps FRUC.

Frame repetition is one of the earliest methods applied in FRUC. Commonly used, 3:2 pulldown is a frame repetition implementation for 24-to-60 fps FRUC. To match the $\frac{60}{24} = 2.5$ ratio of up conversion, two repeats of the first frame and one repeat of the second frame are added to the video and so on. Fig. 1 shows a visualisation of frame repetition in this case.

This algorithm can be simply implemented on hardware by repeating corresponding frames in sync with the reference clock. However, the inconstant duration of each frame can cause discontinuous motion in the video which is known as motion judder [1].

B. Frame averaging

Frame averaging, also known as "oversampling", is another method of FRUC similar to frame repetition. For the common 24-to-60 fps case, instead of displaying the two adjacent frames with a 3:2 ratio, each frame is displayed 2.5 times. The middle frame is obtained by blending the two adjacent frames. Fig. 2 shows a visualisation of frame averaging in this case.

The output video of this method appears to be smoother than the result of the frame repetition method. In addition,

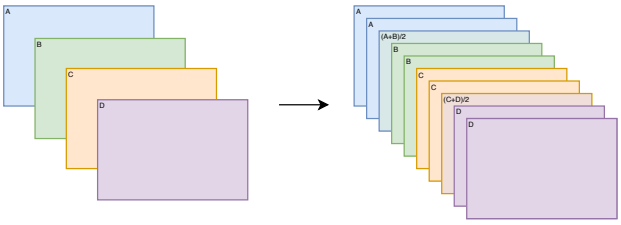

Fig. 2: Frame averaging visualisation for 24-to-60 fps FRUC. (A+B)/2 represents the blending of frames A and B.

the calculation of blending the frames is not computationally intensive; therefore, it is feasible to be implemented on hardware for real-time FRUC. Although the quality of the output video is improved by using this method, the problem of motion judder is not yet solved [1].

C. Linear interpolation

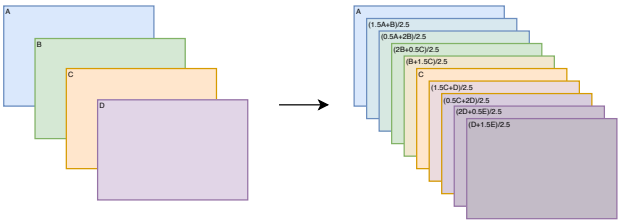

Fig. 3: Linear interpolation visualisation for 24-to-60 fps FRUC.

Linear interpolation is another FRUC method. This algorithm creates interpolated frames by linearly blending the video frames. The interpolated frames are blended depending on the weights of frames which is linearly distributed. Fig. 3 shows a visualisation of linear interpolation in this case.

The problem of this method is that many blended frames, which look blurry, are introduced to the output video. [1], [2]. Moreover, there is an edge case relating to the final frames to be accounted for. Fig. 3 shows the last two frames darkening– this is because frame E is not present (and is taken as an empty black frame); however, this can be handled by simply repeating the last frame as required.

III. Motion Estimation

This section describes different methods for ME, where the main objective is to estimate motion of moving objects by calculating motion vectors (MVs).

The trade-off between accuracy and speed of ME is a common issue for MEMC FRUC, especially in real-time applications. A variety of ME algorithms were proposed by researchers in an attempt to address the problem of accurately estimate motion at a low computational cost.

A. Block Matching Algorithms

Block matching algorithms (BMA) are widely used for motion estimation in video coding due to its relatively low complexity and its compatibility with hardware implementation [58]. The objective of these algorithms is to estimate MVs by matching blocks from subsequent frames. Given a $K \times K$ block in the current frame, the goal is to find the best matching $K \times K$ block in a reference frame that minimises some cost function. Sum of Absolute Differences (SAD) is commonly used as a cost function due its low computational cost [32], [36], [58]. Calculating the SAD between two $K \times K$ blocks requires $2K^2 - 1$ additions. The difference in spatial location between the two matched blocks then represents the MV.

In theory, it is necessary to perform Full Search (FS) in order to find the block that produce the global minimum of the cost function. FS is a simple algorithm that compares the $K \times K$ block from the current frame with all possible $K \times K$ blocks in the reference frame. Finding the best matching $K \times K$ block in a frame of size $N \times M$ based on SAD therefore requires $NM(2K^2 - 1)$ additions. As described, the FS algorithm requires a large number of computations and therefore a variety of algorithms were proposed to reduce the computational complexity of the search process.

1) Three Step Search Algorithm: The three step search algorithm (TSS) is a simple algorithm with near optimal performance and low computational cost [59]. TSS benefits from an efficient search scheme which only requires 25 block comparisons, hence $25(2K^2 - 1)$ additions, which is a significant computational reduction compared to FS.

Further, modifications of TSS were proposed in order to improve the performance by modifying the search pattern. The New Three Step Search (NTSS) [63] employs a center-biased checking point pattern and a halfway-stop technique to reduce the computational cost. The Predictive three step search presented in [60] reduces the number of searched blocks by using information from neighbouring blocks. Chau *et al.* [61] propose a modified search scheme that aims to solve the problem of getting trapped in a local minimum, a problem of TSS.

2) Diamond search: Diamond search (DS) is closely related to TSS. This algorithm was first presented in [62] and consists of two diamond-shaped search patterns which are applied recursively until the best matching block is found. According to [62], the DS greatly outperforms the standard TSS and works better than the NTSS in terms of computational complexity and matching accuracy. However, it should be mentioned that DS does not provide a fixed number of searching points, which is the case for TSS.

3) Hierarchical Block Motion Estimation: Hierarchical Block Motion Estimation (HBME) consists of algorithms where the block size, frame resolution and/or the search pattern can change depending on the level of the hierarchy. The idea is to start searching at the highest level and use a small frame-to-block size ratio in order to prevent convergence to local minima. This can be achieved by either increasing the block size or down-sampling the frame. The motion vectors from the higher level is used as a starting point for searching blocks in the lower level. As the hierarchy level reduces, the frame-to-block size ratio is increased in order to find more accurate motion vectors [68]. Different search methods were presented together with HBME algorithms, and in some

algorithms, methods change depending on the hierarchical level. T. Lee and D.V. Anderson [73] propose a HBME algorithm that uses a binary tree architecture with variable block sizes. In [74], a three level HBME is proposed that applies the FS at the highest level, a modified DS algorithm in the second level and finally TSS in the first level to obtain the final motion vectors.

IV. Motion Compensated Interpolation

The approach taken during motion compensated interpolation (MCI) largely depends on the type of ME used and whether the resulting motion vectors are unidirectional or bidirectional.

Unidirectional MCI is the conventional approach to motion compensated interpolation [69]–[71]. Blocks in the interpolated frame can be found by taking a block in the initial frame and multiplying its motion vector (by 0.5 to get an interpolated frame timed directly between the two input frames) to find its location in the interpolated frame. This technique can result in blocking artifacts (the final image appearing segmented by blocks), overlapping blocks (when multiple motion vectors point to the same location), and holes (when there is no motion to the location or background is uncovered) in the interpolated image.

To combat overlapping pixels, a depth approach [70] can be used to determine which pixel should be used: the pixel belonging to the object closer to the camera should remain visible in the interpolated frame. This approach, however, can be computationally expensive and therefore may have limited application to hardware design.

Irregular-grid expanded-block weighted motion compensation [71] is an algorithm aiming to reduce both blocking artifacts and resolve overlaps. It expands the size of the interpolated blocks and applies a weighted filter to them before normalising to generate pixel values for the interpolated frame. The typical method of dealing with overlaps is to simply take the pixel from the motion vector with the lowest corresponding SAD score but this method can lead to rapid changes between contiguous pixels as they switch between two or more motion vectors [71]. This method shows obvious improvements over the standard method in sample images with a reasonably low complexity.

A typical method used to fill in holes is the application of a median filter [70]. This method is preferred largely because of its simplicity but because the filter is only applied over areas with no pixel information, it is not as viable for hardware or real time solutions. Block-wise directional hole interpolation (BDHI) [71] is an algorithm to combat holes. It is more complex than a simple median filter, but due to its block-wise nature, it may be better suited to hardware implementations. Additionally, it is able to interpolate features from surrounding pixels, as apposed to averaging them. This leads to superior results over median filters.

Another factor leading to poor interpolation of frames is scene change. If there is a change in scene between two frames, then interpolating those two frames using motion estimation will not produce desirable results. Scene detection can be used to determine when this occurs and then instead of interpolating from the surrounding frames, the frame can be constructed by extrapolating either the two previous frames or the two following [72]. This design specifically may not translate well to hardware, as more reference frames need storing leading to high resource cost, but the algorithm may be able to be adapted to be more appropriate.

V. Advanced Methods

In this section, advanced FRUC methods, which refer to modern and state-of-the-art algorithms, are studied. It is worth noting that even though most of these algorithms are currently not suited for real-time hardware applications, the algorithms serve as a look into future possibilities as hardware improves. Further, some methods involve a combination of previous methods discussed in this review, hence demonstrating how different and simpler methods can be combined to produce better results.

A common theme in a number of modern and advanced algorithms is the use of Machine Learning (ML), pioneered by Long *et al.* [5]. More specifically, fast, scalable and end-to-end trainable Convolutional Neural Networks (CNNs) [15] are used. The motivation for using ML arises from the availability of high frame-rate videos [3], [6]–[8], [57], [77]. These videos can be down-converted to a lower frame-rate by dropping frames or synthesising blurred frames to simulate motion-blurred low frame-rate video [4], [8]. The resulting low frame-rate video can then be used to train an ML model.

We outline advanced implementations of motion estimation and motion compensated interpolation in sections V-A and V-B respectively. In section V-C, we discuss novel FRUC methods, some of which comprise parts of the former two sections. Finally, section V-D delineates a quantitative performance comparison amongst the best, state-of-the-art models at time of writing.

A. Motion Estimation: Advanced implementations

1) Optical Flow: Optical flow as a motion estimation method is heavily researched and have led to impressive performance on challenging benchmarks [10]–[12]. Top-performing classical methods usually adopt the energy minimisation developed by Horn *et al.* [13]. With the advancement of CNNs, Dosovitskiy *et al.* [14] proposed FlowNetS and FlowNetC, which, while performing below state-of-the-art, demonstrates feasibility of directly estimating optical flow from raw images using a U-Net CNN architecture [16]. Improving upon this, Ilg *et al.* [17] created FlowNet2, which comprise stacked FlowNetS models. FlowNet2 runs faster and performs on par with state-of-the-art models, but is prone to over-fitting during training (as it is a large network) and is very resource-intensive. SpyNet [18] solves the model size issue by adopting coarse-to-fine estimation in the network design– it residually updates the flow across the levels of a spatial pyramid with individual trainable weights. While smaller and faster, SpyNet performs below FlowNet2. LiteFlowNet

[20] and PWC-Net [19] further combine the coarse-to-fine strategy with multiple ideas from both classical methods and recent deep learning approaches. PWC-Net in particular outperformed all published methods on the common public benchmarks [11], [12], [21]. To alleviate the need for training data with ground truth in a specific domain, unsupervised [22]–[27], semi-supervised [28], [29] and self-supervised [30] methods have also been developed. To improve these networks, Hur *et al.* [31] proposed an iterative residual refinement (IRR) scheme that can be combined with several backbone networks, notably FlowNet and PWC-Net.

2) Multi-directional Motion Estimation: Traditionally, ME is calculated in forwards direction in time, which from t to $t+1$. It is also possible to assume the flow backwards which estimates the motion from $t+1$ to t. Moreover, these two flows can be combined to synthesise a more accurate target flow [64]. Compare dto the unidirectional method, bidirectional ME performs better when dealing with overlapped areas and holes caused by occlusion [65].

3) Motion Vector Smoothing: The two other estimation methods discussed prior to this method require large computation power. Motion vector smoothing is a way with reduced computational complexity. The basic idea is to have a smoothing filter to eliminate wrong motion vectors and only process the correct candidates [66]. The wrong motion vectors are defined as the vectors that break the continuity in a certain motion field. However, having smaller block sizes of filters makes the method more vulnerable to noise. A corresponding solution is proposed in [67].

B. Motion Compensated Interpolation: Advanced implementations

Compared to ME, MCI has fewer new methods proposed in recent papers. Further study can mainly be divided into two paths: the first one is changing the assumptions, which involves taking more details into account, and the other one relates to using neural networks. For the former, using more details can generate a slightly better result at the expense of increasing computational complexity. This increase is not suitable for resource-constrained hardware. Methods involving neural nets are usually coupled together with ME in an end-to-end model, and they would be discussed further in section V-C.

1) High-order Model: In optical flow, object motion between two frames is assumed to have linear motion. If a motion is non-linear (e.g. a curve), using a high-order polynomial to model the motion yields more accurate estimations. For example, estimating fast and large circular motions using a linear model would give very inaccurate estimations [75]; this would yield an estimation that looks as if the motion traces a straight line and ends with a sudden change in direction.

2) Multiple Inter Frame Interpolation: When interpolating more than one frame between two successive frames in conventional methods, the weights of original frames are used to calculate the ratios of each successive original frame required in the interpolated frames. This is covered in section II-C. Gracewell and John [76] proposed calculating the new frame in the middle first and then repeating the algorithm using the new frame and original frames to get the required frames, which allows for advanced mathematical optimisations, reducing computational complexity by 85%. Fig. 5 shows an illustration of this method. A drawback of this method that it can only be applied in cases where the number of interpolated frames required between two original frames is $2^n - 1$ where n is a positive integer; in other words, this method only works for FRUC by a factor of 2.

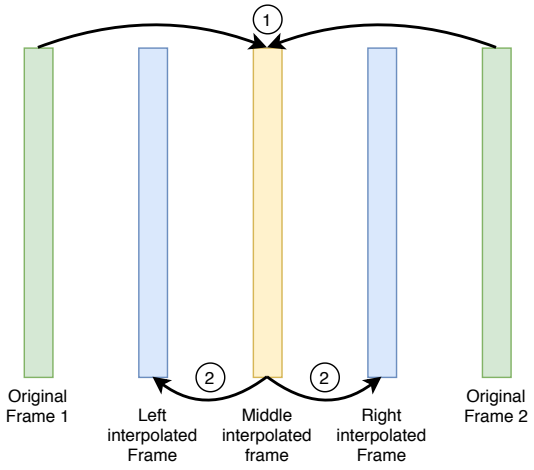

Fig. 5: Multiple interpolated frame between two successive original transmitted frames [76].

C. Novel FRUC implementations

In this section, we outline novel FRUC implementations categorised into six categories. We highlight the novelty in each approach, with–where present–its main drawback. Fig. 4 outlines components of different frameworks.

1) Novel MEMC-based methods (Fig. 4a): Conventional MEMC-based FRUC methods produce visual artifacts due to relative motions and occlusion between objects with different depth, leading to flow vectors causing incorrect interpolation results with hole regions [37]. Hence, advanced MEMC-based methods include a post-processing step to minimise these artifacts [32]–[36]. Kim *et al.* [36] utilise a hole interpolation method to restore missing pixels, while Wang *et al.* [33] propose a trilateral filtering method to fill holes and smooth compensation errors. A drawback of the mentioned methods, besides producing results that are below state-of-the-art, is that the post-processing step proposed is not end-to-end trainable.

2) Phase-based methods (Fig. 4b): Leveraging developments in phase-based methods that represent motion in the phase shift of individual pixels [38], [39], Meyer *et al.* [40] developed a phase-based method which manipulates pixel-phase information within a multi-scale pyramid for frame interpolation. Improving upon this, PhaseNet [41], a learning-based implementation, was proposed. However, phase-based methods are less effective in handling large motion in complicated scenarios, resulting in visual artifacts.

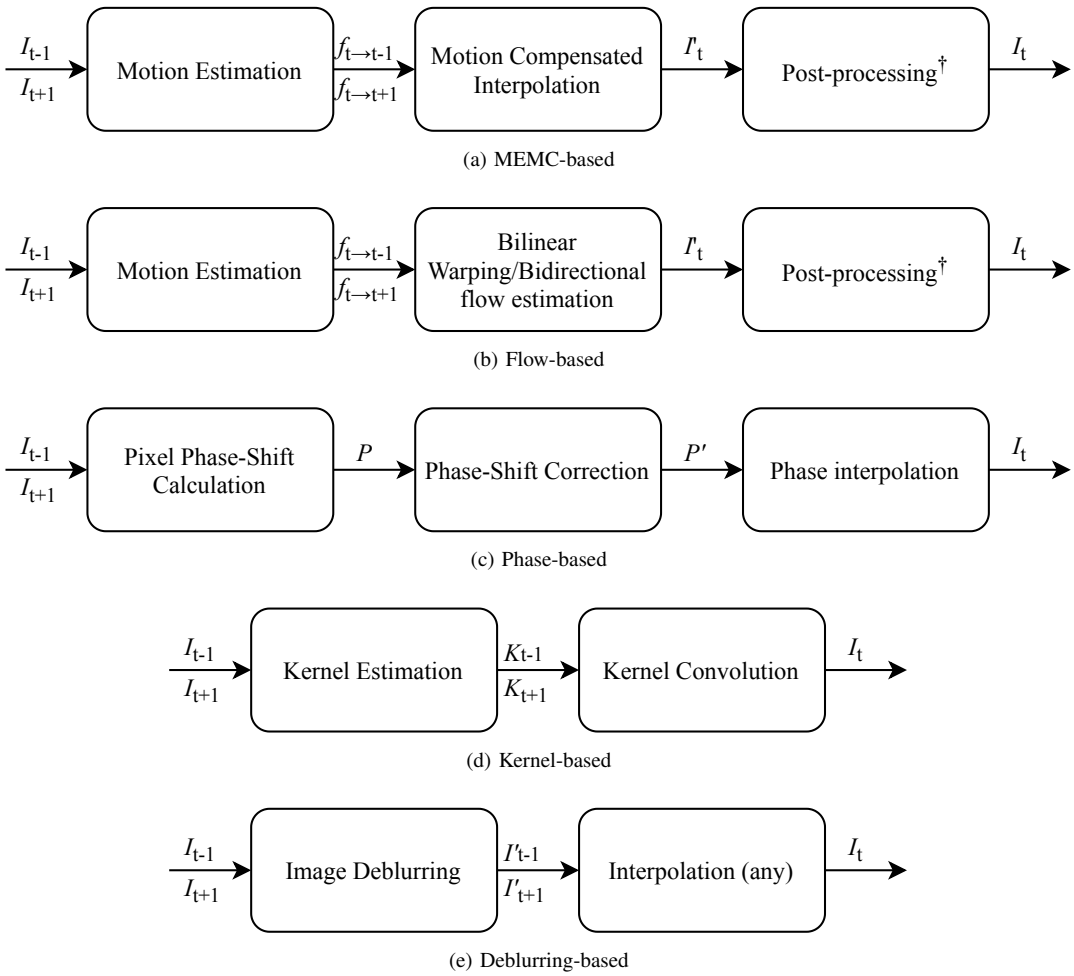

Fig. 4: Frameworks of different FRUC methods, taking in two consecutive images I_{t-1} and I_{t+1} to produce an interpolated frame I_t. I, f, P and K correspond to image, motion flow, phase-shift, and kernel respectively, while the subscripts denote the point in time.

3) Flow-based methods (Fig. 4c): Based on advancements in optical flow estimation by deep CNNs detailed in section V-A1, several methods based on end-to-end deep models have been developed. These approaches either predict bidirectional flow [42] or use bilinear warping operations to align input frames based on linear motion models [43]–[45]. A common technique utilised to synthesise output images is estimating an occlusion mask to adaptively blend warped frames. As bilinear warping blends neighbour pixels based on sub-pixel shifts, when the input frames are not aligned enough, the flow-based methods generate blurry artifacts [37].

4) Kernel-based methods (Fig. 4d): Frame interpolation can be formulated as convolution operators over patches instead of relying on optical flow, therefore ridding the disadvantages of optical flow methods. Niklaus *et al.* [46], [47] developed two models: (1) the AdaConv [46] model which estimates spatially-adaptive convolutional kernels for each output pixel and (2) the SepConv [47] model which is an improvement upon the AdaConv model with lower memory requirements and better interpolation results. Experimentally, Niklaus *et al.* [47] were able to show that the formulation of video interpolation as a single convolution gracefully handles challenges like occlusion, blur and abrupt brightness change. A drawback of these models is that they cannot handle motion larger than a pre-defined kernel size, therefore producing artifacts with large motion [37].

5) Deblurring-based methods (Fig. 4e): In section V, methods for preparing low frame-rate were mentioned. One method involved adding motion blur to the high frame-rate video to better simulate low frame-rate video. The reverse of this process–deblurring the video then interpolating the frames–is a method of interpolation that accounts for the nature of high shutter-speed in high frame-rate video. Proprietary implementations of this interpolatioon technique exist in industry [49], [53], [54]. Further, Shen *et al.* [48] proposed BIN (Blurry video frame INterplation) which is a model that simultaneously reduces motion blur and up-converts the frame rate. A disadvantage of this methodology is that the deblurring

TABLE I: Quantitative Performance Results of State-of-the-Art FRUC Models.

Method	Benchmark						
	Vimeo90k [42]		*UCF101 [51]*		*Middlebury [10]*	Efficiency	
	PSNR	SSIM	PSNR	SSIM	IE	#Parameters (millions)	Runtime (seconds)
SoftSplat - $\mathscr{L}_{Lap}$ [56]	**36.10**	0.970	**35.39**	0.952	**4.22**	-	-
MEMC-Net* [37]	34.40	**0.974**	35.01	**0.968**	5.24	70.31	0.12
DAIN [50]	34.70	0.964	35.00	0.950	4.86	24.02	0.13
RRIN [55]	35.22	0.964	34.93	0.950	-	**19.19**	**0.08**
SepConv - $\mathscr{L}_1$ [47]	33.80	0.956	34.79	0.947	5.61	21.60	0.20

Where applicable, the efficiency of methods are outlined: the number of model parameters (millions) and runtime (seconds per 480 × 640 image)–reported by [55]–are shown.

step produces images that are unnaturally and overly sharp.

6) Other methods: Methods outlined in this category either do not fit into any of the above categories, or use a combination of the former methods.

MEMC-Net [37] is a method combining MEMC-based, flow-based, and kernel-based methods. It is a data-driven end-to-end trainable model, which includes the artifact-fixing post-processing step as part of the trainable model, therefore solving the flow-based model problem of having a separate post-processing step. With regard to the kernel-based large motion artifacts problem, MEMC-Net proposes a fix by learning spatially-varying interpolation kernels for each pixel, thus better accounting for occlusion and dis-occlusion.

DAIN (Depth-Aware video frame INterpolation) [50] is another method exploiting advantages of various methods by combining them. DAIN proposes a model comprising four components: (1) a flow-based optical flow component, (2) a depth estimation component, (3) a context extraction component and (4) a kernel-based component. DAIN forward-warps the optical flow then backward-warps the input images to the target location according to the warped optical flow. Utilising the depth component to detect occlusion for flow aggregation rids the artifacts problem with optical flow. However, in challenging cases, DAIN produces blurred results with unclear boundaries.

RRIN (Residue Refinement video frame Interpolation Network) [55] exploits residue learning [52] and adaptive weight map for accurate video frame interpolation. Implementing submodules of RRIN using U-Net [16] with less depths effectively reduces the model size and computation complexity.

SoftSplat [56] utilises softmax splatting to estimate an optimal flow, which is then used to forward-warp frames and their feature pyramid representations. SoftSplat is end-to-end trainable and enables optimising of feature pyramids for image synthesis as well as fine-tuning of the optical flow estimator. SoftSplat is similar to DAIN [50]; however, SoftSplat is conceptually simpler due to not warping the flow and not incorporating depth- or kernel-estimates. DAIN also uses linear splatting in contrast to SoftSplat's softmax splatting which is translational invariant and yields better results.

D. Quantitative performance comparison of state-of-the-art models

Table I gives a comparison of the quantitative performance and efficiency results of state-of-the-art models on three video frame interpolation benchmarks: Vimeo90k [42], UCF101 [51] and Middlebury [10]. PSNR, SSIM and IE denote Peak Signal-to-Noise Ratio (higher is better), Structural Similarity Index (higher is better) and Interpolation Error (lower is better) respectively. See section V-D1 for details on these metrics.

As some papers train more than one model to suit different applications, where applicable, the model with highest mean score across the benchmarks is chosen; this is the case for SoftSplat - $\mathscr{L}_{Lap}$ [56], SepConv - $\mathscr{L}_1$ [47] and MEMC-Net* [37]. Bolded numbers denote best scores in each section.

An overview of these performance results point to the fact that models rely on multiple different methods and algorithms to achieve top-performing results. In the five models, SepConv [47], a kernel-based method and the only method relying on one paradigm–detailed in section V-C4–was outperformed by the other models which combine novel methods, detailed in section V-C6. Moreover, the range of years in which the mdoels were developed is 2017-2020, showing that there is very active research and development in this field. As advanced techniques (e.g. ML), computing power and datasets continue to improve, there is much to expect in the future as paradigm-shifting methods are introduced.

1) Definitions of IE, PSNR and SSIM: IE [10], also known as mean squared error (MSE) is computed by:

$$\text{IE} = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N}$$

where I_1 is the interpolated frame and I_2 is the ground truth. M and N are the number of rows and columns respectively in the input images.

PSNR is developed from IE and expressed in decibels:

$$\text{PSNR} = 10log_{10}(\frac{\max_i^2}{\text{IE}})$$

where $\max_i$ is the maximum range in the input image data type. For example if the image uses 8 bits per sample, $\max_i$ is $2^8 = 255$.

With more work into quality assessment methods using human visual system (HSV), Structural SIMilarity (SSIM) Index was proposed [78]. To compute it, firstly, SSIM is defined as:

$$\text{SSIM}(x,y) = [l(x,y)]^\alpha \times [c(x,y)]^\beta \times [s(x,y)]^\gamma$$

where $l(x,y)$ is the luminance comparison between signal x and signal y, $c(x,y)$ is the contrast comparison and $s(x,y)$

stands for structural comparison. α, β and γ are the weights of each comparison, normally defaulted to 1.

Luminance is computed by:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

where μ is the average intensity of the signal. C_1 is a small constant to prevent division by zero errors when $\mu_x^2 + \mu_y^2$ is close to zero. (Similar constants C_2 and C_3 are also used in the next two equations).

Contrast comparison is calculated as:

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

where σ is the standard deviation of a signal.

And lastly, the structural comparison is computed as:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

.

To simplify the equation, C_3 is chose to be half of C_2, and this is the formula used in the implementation:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

ACKNOWLEDGMENT

We would like to thank Kieron Turkington (Intel Corporation), industrial supervisor of this group project, and Professor George Constantinides (Imperial College London), academic supervisor of this project.

REFERENCES

[1] N. Haasn, "Interpolation techniques", https://github.com/haasn/mpvhq-old/wiki/Interpolation, 2015.

[2] K.T. Gribbon, D.G. Bailey,"A Novel Approach to Real-time Bilinear Interpolation", 2004.

[3] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich and O. Wang, "Deep Video Deblurring for Hand-Held Cameras," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 237-246.

[4] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, J. Davis, "Synthetic shutter speed imaging", Comput. Graph. Forum, vol. 26, no. 3, pp. 591-598, 2007.

[5] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In ECCV, 2016

[6] A. Mackin, F. Zhang and D. R. Bull, "A Study of High Frame Rate Video Formats," in IEEE Transactions on Multimedia, vol. 21, no. 6, pp. 1499-1512, June 2019.

[7] M. Emoto, Y. Kusakabe and M. Sugawara, "High-Frame-Rate Motion Picture Quality and Its Independence of Viewing Distance," in Journal of Display Technology, vol. 10, no. 8, pp. 635-641, Aug. 2014.

[8] A. Handa, R. Newcombe, A. Angeli and A. Davison, "Real-Time Camera Tracking: When is High Frame-Rate Best?" in Proc. of the European Conference on Computer Vision(ECCV), Oct. 2020

[9] M. Ghanbari, "Video Coding: An Introduction to Standard Codecs," Institution of Electrical Engineers, 1999.

[10] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. International Journal of Computer Vision (IJCV), 2011.

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In European Conference on Computer Vision (ECCV), 2012.

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[13] B. Horn and B. Schunck. Determining optical flow. Artificial Intelligence, 1981.

[14] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. FlowNet: Learning optical flow with convolutional networks. In IEEE International Conference on Computer Vision (ICCV), 2015.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989

[16] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2015.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[18] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[19] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In CVPR, pages 8934-8943, 2018.

[20] T.-W. Hui, X. Tang, and C. C. Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In CVPR, pages 8981-8989, 2018.

[21] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In CVPR, pages 3061-3070, 2015.

[22] A. Ahmadi and I. Patras. Unsupervised convolutional neural networks for motion estimation. In ICIP, pages 1629-1633, 2016.

[23] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In AAAI, pages 7251-7259, 2018.

[24] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha. Unsupervised deep learning for optical flow estimation. In AAAI, pages 1495-1501, 2017.

[25] Y. Wang, Y. Yang, Z. Yang, L. Zhao, and W. Xu. Occlusion aware unsupervised learning of optical flow. In CVPR, pages 4884-4893, 2018.

[26] J. J. Yu, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In ECCV Workshops, volume 3, pages 3-10, 2016.

[27] Y. Zhu and S. Newsam. DenseNet for dense flow. In ICIP, pages 790-794, 2017.

[28] Y. Zhu, Z. Lan, S. Newsam, and Alexander G. Hauptmann. Guided optical flow learning. In CVPR 2017 Workshops, 2017.

[29] W.-S. Lai, J.-B. Huang, and M.-H. Yang. Semi-supervised learning for optical flow with generative adversarial networks. In NIPS*2017, pages 354-364.

[30] P. Liu, M. Lyu, I. King and J. Xu, "SelFlow: Self-Supervised Learning of Optical Flow," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 4566-4575.

[31] J. Hur and S. Roth, "Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 5747-5756, doi: 10.1109/CVPR.2019.00590.

[32] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Motioncompensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 4, pp. 407–416, 2007.

[33] C. Wang, L. Zhang, Y. He, and Y.-P. Tan, "Frame rate upconversion using trilateral filtering," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 6, pp. 886–893, 2010.

[34] M. Biswas and V. Namboodiri, "On handling of occlusion for frame rate up-conversion using video in-painting," in IEEE International Conference on Image Processing. IEEE, 2010, pp. 785–788.

[35] W. H. Lee, K. Choi, and J. B. Ra, "Frame rate up conversion based on variational image fusion," IEEE Transactions on Image Processing, vol. 23, no. 1, pp. 399–412, 2014.

[36] U. S. Kim and M. H. Sunwoo, "New frame rate up-conversion algorithms with low computational complexity," IEEE Transactions on

Circuits and Systems for Video Technology, vol. 24, no. 3, pp. 384–393, 2014.
[37] W. Bao, W. Lai, X. Zhang, Z. Gao and M. Yang, "MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2019.2941941.
[38] P. Didyk, P. Sitthi-amorn, W. T. Freeman, F. Durand, and W. Matusik. Joint view expansion and filtering for automultiscopic 3D displays. ACM Trans. Graph., 32(6):221, 2013.
[39] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman. Phase-based video motion processing. ACM Trans. Graph., 32(4):80, 2013.
[40] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. SorkineHornung, "Phase-based frame interpolation for video," in IEEE Conference on Computer Vision and Pattern Recognition, 2015.
[41] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross and C. Schroers, "PhaseNet for Video Frame Interpolation," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 498-507, doi: 10.1109/CVPR.2018.00059.
[42] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," arXiv preprint arXiv:1711.09078, 2017
[43] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in IEEE International Conference on Computer Vision, 2017.
[44] J. van Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero, "Frame interpolation with multi-scale deep loss functions and generative adversarial networks," arXiv preprint arXiv:1711.06045, 2017.
[45] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[46] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[47] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in IEEE International Conference on Computer Vision, 2017
[48] W. Shen, W. Bao, G. Zhai, L. Chen, X. Min and Z. Gao, "Blurry Video Frame Interpolation," in CVPR 2020.
[49] Sony.co.uk. 2020. What Is Motionflow XR And X-Motion Clarity? — Sony UK. [online] Available at: ¡https://www.sony.co.uk/electronics/support/articles/00232431¿ [Accessed 5 May 2020].
[50] W. Bao, W. Lai, C. Ma, X. Zhang, Z. Gao and M. Yang, "Depth-Aware Video Frame Interpolation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 3698-3707, doi: 10.1109/CVPR.2019.00382.
[51] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In CRCV-TR-12-01, 2012.
[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[53] Samsung Electronics America. 2020. Motion Smoothing And The Soap Opera Effect On Samsung Tvs. [online] Available at: ¡https://www.samsung.com/us/support/answer/ANS00080741/¿ [Accessed 5 May 2020].
[54] Trumotion Overview — LG Canada. [online] Lg.com. Available at: ¡https://www.lg.com/ca_en/support/product-help/CT32004644-1411521672907-others¿ [Accessed 5 May 2020].
[55] H. Li, Y. Yuan and Q. Wang, "Video Frame Interpolation Via Residue Refinement," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 2613-2617, doi: 10.1109/ICASSP40776.2020.9053987.
[56] S. Niklaus and F. Liu (2020). Softmax Splatting for Video Frame Interpolation.
[57] J. Janai, F. Guney, J. Wulff, M. Black, and ¨ A. Geiger. Slow Flow: Exploiting High-Speed Cameras for Accurate and Diverse Optical Flow Reference Data. In IEEE Conference on Computer Vision and Pattern Recognition, 2017
[58] K. Laidi and M. Nibouche, "On the Performance of FPGA Implementation of Block Matching Algorithms for Video Motion Estimation.," 2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM), Algiers, 2018, pp. 1-5, doi: 10.1109/CISTEM.2018.8613430.
[59] H. Amirpour, A. Mousavinia and N. Shamsi, "Predictive Three Step Search (PTSS) algorithm for motion estimation," 2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP), Zanjan, 2013, pp. 48-52, doi: 10.1109/IranianMVIP.2013.6779948.
[60] H. Amirpour, A. Mousavinia and N. Shamsi, "Predictive Three Step Search (PTSS) algorithm for motion estimation," 2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP), Zanjan, 2013, pp. 48-52, doi: 10.1109/IranianMVIP.2013.6779948.
[61] Lap-Pui Chau and Xuan Jing, "Efficient three-step search algorithm for block motion estimation in video coding," 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)., Hong Kong, 2003, pp. III-421, doi: 10.1109/ICASSP.2003.1199501.
[62] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block matching motion estimation," Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat., Singapore, 1997, pp. 292-296 vol.1, doi: 10.1109/ICICS.1997.647106.
[63] Reoxiang Li, Bing Zeng and M. L. Liou, "A new three-step search algorithm for block motion estimation," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 438-442, Aug. 1994, doi: 10.1109/76.313138.
[64] X.Xu, L.Siyao, W.Sun, Q.Yin and M.Yang, "Quadratic video interpolation," in arxiv.org/abs/1911.00627, 2019
[65] Songhyun Yu and Jechang Jeong, "Multidirectional motion estimation algorithm for frame rate up-conversion," MATEC Web Conf. Volume 125, 2017, 21st International Conference on Circuits, Systems, Communications and Computers (CSCC 2017)
[66] J.Zhai, K.Yu, J.Li, S.Li, "A low complexity motion compensated frame interpolation method," in 2005 IEEE International Symposium on Circuits and Systems
[67] H.Yin, X.Fang, H.Yang, S.Yu, X.Yang, "Motion Vector Smoothing for True Motion Estimation," in 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings
[68] M. Bierling, "Displacement Estimation By Hierarchical Blockmatching," Proc. SPIE 1001, Visual Communications and Image Processing '88: Third in a Series, (25 October 1988)
[69] C.-L. Huang and T.-T. Chai, "Motion-compensated interpolation for scan rate up-conversion", Optical Engineering, vol. 35, no. 1, pp. 166-176, Jan. 1996.
[70] J. Benois-Pineau and H. Nicolas, "A new method for region-based depth ordering in a video sequence: Application to frame interpolation", Journal of Visual Communication and Image Representation, vol. 13, pp. 363-385, 2002.
[71] D. Wang, A. Vincent, P. Blanchfield and R. Klepko, "Motion-Compensated Frame Rate Up-Conversion—Part II: New Algorithms for Frame Interpolation," in IEEE Transactions on Broadcasting, vol. 56, no. 2, pp. 142-149, June 2010, doi: 10.1109/TBC.2010.2043895.
[72] D. Hui-Ping, Y. Li, X. Wei, H. Qing-Di and L. Rong, "Adaptive Interpolation/Extrapolation and Motion Vector Processing Method for Frame Rate Up Conversion," 2009 Fifth International Conference on Image and Graphics, Xi'an, Shanxi, 2009, pp. 18-22, doi: 10.1109/ICIG.2009.167.
[73] Teahyung Lee and D. V. Anderson, "Architecture for Hierarchical Block Motion Estimation Using Variable Block Sizes," 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, 2006, pp. III-III, doi: 10.1109/ICASSP.2006.1660815.
[74] T. Sebastian and J. Anitha, "Hybrid hierarchical search motion estimation for Video Compression," 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, 2016, pp. 88-92, doi: 10.1109/AEEICB.2016.7538403.
[75] W. Bao, X. Zhang, L.Chen, L.Ding, Z.Gao, "High-Order Model and Dynamic Filtering for Frame Rate Up-Conversion," in IEEE Transactions on Image Processing (Volume: 27 , Issue: 8 , Aug. 2018)
[76] J. Gracewell, M. John, "Motion Compensation based Multiple Inter Frame Interpolation", in 2016 International Conference on Signal Processing and Communication (ICSC)
[77] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," Accepted to ACM Multimedia Syst. Conf., Istanbul, Turkey, June 2020.
[78] Z. Wang ; A.C. Bovik ; H.R. Sheikh ; E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing (Volume: 13 , Issue: 4 , April 2004)