Video Signal Processing

Yung-Lin Huang

1 Introduction

Nowadays, multimedia information systems become more popular. People love to watch images and videos on different devices such as television (TV), personal computer (PC), and mobile phone. The variety of display formats rapidly increase. This fact has resulted in a demand for efficiently converting between various video formats.

We briefly explain the format of the video signal. An image consists of two-dimensional (2D) signals called *pixels* which represent colors in a point of the picture, and a video is composed of a sequence of images called frames. That is, the consecutive frames form a video as shown in Fig. 1. Therefore, a video is a three-dimensional (3D) signal including x-direction, y-direction, and temporal domain. Frame rate, expressed in frame per second (FPS), is the frequency that the consecutive frames are shown on the display device. The frame rate differs between lots of video formats. For example, movie films are at 24 FPS, and the videos recorded by mobile phone are often at 30 or 60 FPS.

In addition to the frame rate, the refresh rate of display devices should be took into consideration. Liquid crystal display (LCD), which is one of the most important display techniques, is widely used in many display devices. To provide high visual quality videos, LCD is capable of displaying high frame rate videos at 120 FPS or more. However, the video frame rate differs from many sources and is often lower than the refresh rate.

Y.-L. Huang
National Taiwan University, Taipei City, Taiwan
e-mail: cary@media.ee.ntu.edu.tw

G.J. Dolecek (ed.), *Advances in Multirate Systems*,
DOI 10.1007/978-3-319-59274-9_6

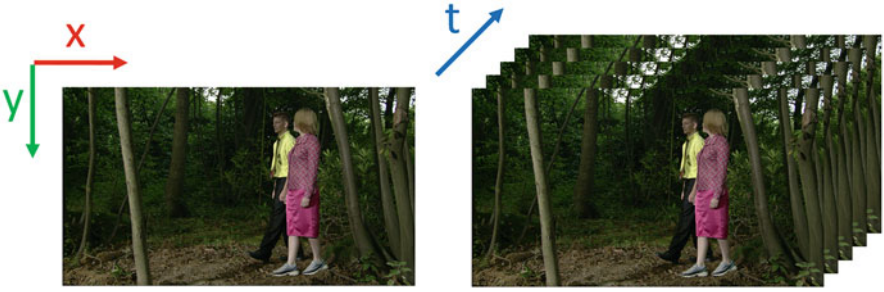


Fig. 1 Illustration of image and video signal. A video signal consists of a sequence of images. It is a three-dimensional signal including *x*-direction, *y*-direction, and temporal domain

Frame rate conversion (FRC) is a post-processing technique to convert the frame rate. The technique aims to increase or decrease the video frame rate for different applications. Because increasing the frame rate introduces smoother motion in videos, frame rate up-conversion (FRUC) is often operated to convert the frame rate from a lower number to a higher one. The technique is useful for lots of applications that are stated in the following section.

This chapter is organized as follows. First, we introduce several multirate video applications in Sect. 2 to show the importance of the multirate video techniques. Several fundamental FRC techniques are first presented in Sect. 3. Then, Sect. 4 explains the motivation of higher frame rate, which gives the reason why the FRUC techniques are most widely adopted. Then, we present the flow diagram and details in each part of state-of-the-art FRUC techniques in Sect. 5. The evaluation methods for the FRUC techniques are then introduced in Sect. 6. Moreover, Sect. 7 demonstrates the importance of hardware architecture design and recent research results. Finally, a conclusion and further discussion are given in Sect. 8.

2 Multirate Video Applications

There are several video applications that require the FRC techniques. Here we list and introduce four of them in the following.

2.1 Video Format Conversion

As mentioned previously, videos are required to be displayed on many different devices. In different digital video standards, the specification of frame rate differs. There are many variations, and new specifications are also introduced by emerging standards. It varies from 24, to 30, to 60, and even to 120 or higher FPS. Converting the frame rate between different standards is essential.

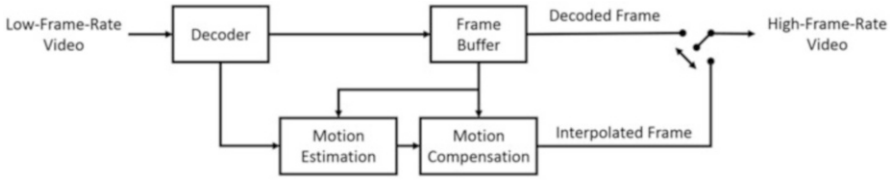


Fig. 2 The frame rate up-conversion technique for low bit rate video coding. The frames are skipped before transmission and interpolated after reception.

2.2 Low Bit Rate Video Coding

To efficiently transmit a video sequence, video coding such as H.264 [1] and high efficient video coding (HEVC) [2] should be applied. Moreover, decreasing the frame rate sufficiently lower the transmitted bit rate because some frames are skipped. After receiving the video, its frame rate can be up-converted to preserve the visual quality when displaying. The decoded and interpolated frames are alternatively displayed as shown in Fig. 2.

2.3 Video Editing

When editing a video, changing the frame rate creates impressive visual effects in a scene with moving objects. To be more specific, decreasing and increasing the frame rate creates time-lapse and slow-motion videos, respectively. Note that the edited video should be displayed at the original frame rate. Take slow-motion effects as an example; increasing the frame rate from 30 to 60 FPS means that the numbers of frame are doubled. The motion lasting 1 s becomes 2 s when displaying the video at the original 30 FPS. This indicates that the motion is slowed down.

2.4 High Refresh Rate Display Device

To provide better visual quality with smooth motion, some off-the-shelf display devices are able to display videos at 120 FPS and higher. Since most videos are at lower frame rate, the FRUC technique should be operated to fill the gap. Figure 3 shows an example of the application. Several frames are interpolated before the video is displayed.

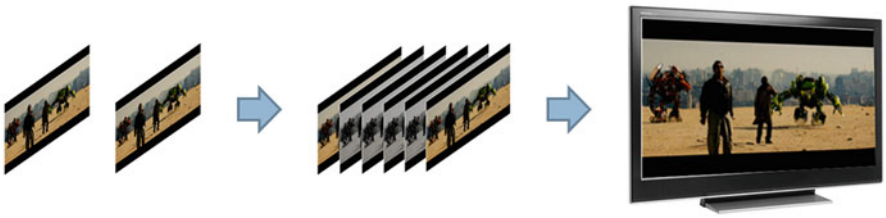


Fig. 3 The frame rate up-conversion technique increases video frame rate for high refresh rate display devices. The four intermediate frames are interpolated to achieve five times up-conversion

3 Fundamental Frame Rate Conversion

We first present several fundamental FRC techniques. Since implementing these techniques is simple, they are still employed in some systems with limited resources.

3.1 Frame Duplicate and Skipping

Different sampling methods introduce different frame rates. When up-converting the frame rate, a straightforward way is to duplicate frames. That is, sampling a frame more than once in the same period to achieve the target higher frame rate. For example, duplicating each frame once creates a video at doubled frame rate. On the contrary, skipping an appropriate number of frames down-converts the frame rate. Frames are often automatically skipped when the resources are limited.

3.2 Three-Two Pull Down

A popular method for changing the sample rate on an original video to achieve a different frame rate is the three-two pull down technique [3]. It converts videos from movie film at 24 FPS to the widely used 30 FPS as shown in Fig. 4. The frames are split into two fields noted as *a* and *b* in Fig. 4. One field contains odd rows and the other contains even rows of the frame. Then, the fields are recombined into frames for video at 30 FPS. The three-two pull down technique causes interlacing problems which is not discussed here.

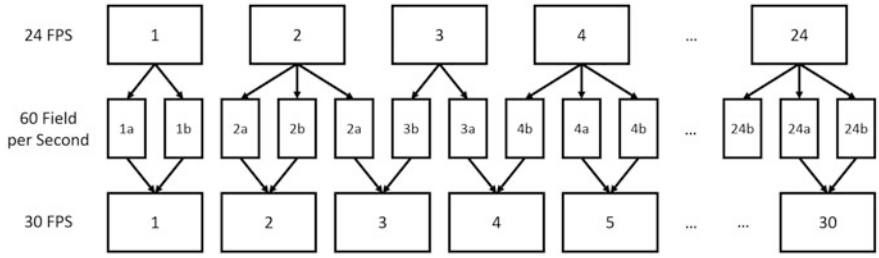


Fig. 4 Illustration of three-two pull down to convert video at 24 FPS to 30 FPS. Four frames are converted to five frames. For each original frame, it is separated into two fields noted as *a* and *b*. The fields are then recombined

3.3 Frame Insertion

Increasing the frame rate by inserting meaningful frames benefits visual quality. Some systems apply black frame insertion or frame blending. *Black frame insertion* means that a frame filled with black color is inserted. The purpose is to reduce the motion blur problem introduced in the next section. *Frame blending* inserts a frame $f(t)$ between frame $t-1$ and frame $t+1$. It utilizes the blending function $f(x, y, t)$ for the pixels of inserted frames,

$$f(x, y, t) = \frac{\lambda_1 f(x, y, t-1) + \lambda_2 f(x, y, t+1)}{2}$$

The (x, y, t) mean the spatial and temporal position of a pixel. It blends the pixels at the same position in the previous and next frames. The parameters λ_1 and λ_2 can be adjusted to be adaptive weighting.

Since videos always present moving objects or scene, blending the pixels at the same position often causes blurred pixels. Therefore, first estimating the motion in videos plays an important role for frame insertion. Motion-compensated FRUC, abbreviated as MC-FRUC or simply FRUC, is the essential technique. In the following section, we state more details of the motivation before introducing the FRUC techniques.

4 Motivation of Higher Frame Rate

As mentioned previously, the FRUC technique is important because video at higher frame rate can display smoother motion. This also indicates better visual quality. We explain this in the aspects of LCD motion blur problems. The visual quality of LCDs suffers from motion blur due to the physical property of the liquid crystal. In general, two types of motion blur occur in LCDs [4].

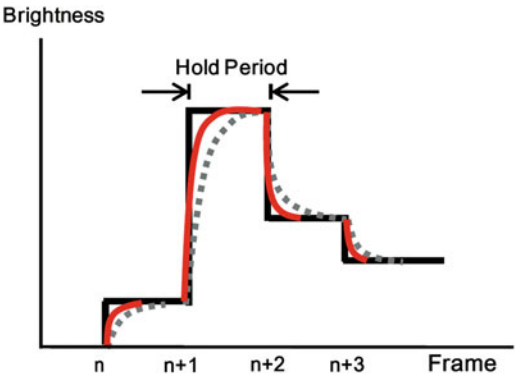


Fig. 5 Hold-type display with slow response. The black solid, the dotted, and the red solid lines represent the target, displayed, and overdrived brightness, respectively

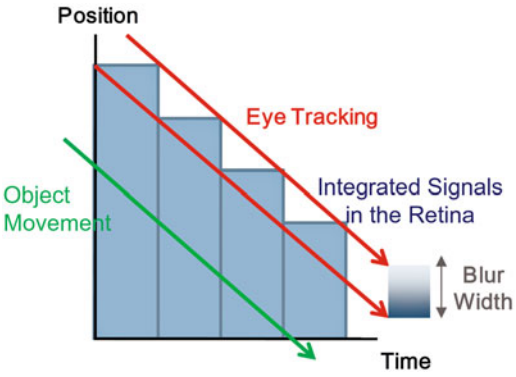


Fig. 6 Direct evaluation of the blur width. The blur width is caused by the integrated signals in the retina

The first motion-blur type is caused by the slow response of the liquid crystal. Figure 5 shows that the black solid line indicates the target brightness and the dotted line indicates the actual displayed brightness. The smooth variation in brightness appears blurred as perceived by human eyes. To overcome this problem, a popular solution called overdrive is applied, i.e., the voltage is first set higher or lower to approach the target brightness and is then set back to the ordinary value after the brightness approaches the target. The red solid line in Fig. 5 shows the resulting brightness, which demonstrates a reduction in the smooth variation in brightness.

The second type is called the hold-type motion blur. Figure 5 shows that maintaining the brightness is termed as the "hold period," which is equal to the inverse of the frame rate. Figure 6 shows that when human eyes track the movement of an object with velocity v, the intensity is continuously integrated in the retina, but the actual intensity discretely changes. This divergence causes the integrated signals in the object boundary in the retina to smoothly decrease or increase. The range of the decrease or increase is called the blur width and can be directly expressed as

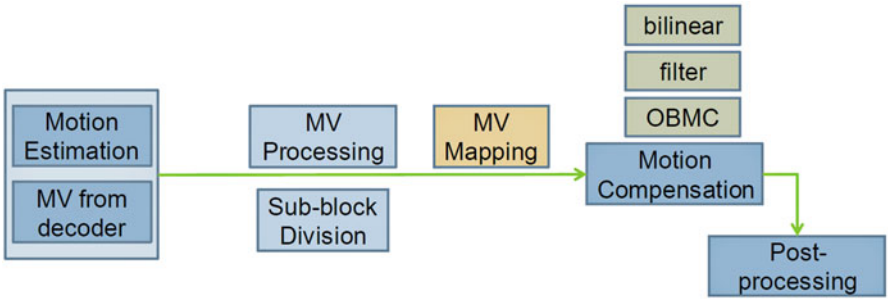


Fig. 7 General flow diagram of the FRUC technique. The MVs are estimated using the ME or retrieved from a video decoder, after which the MVs are refined and mapped to interpolate the intermediate frames using the MC

$$\text{Blur width}_{\text{direct}} = \frac{v}{\text{frame rate}}$$

Another method of evaluating the hold-type motion blur is based on the sampling and reconstruction theory of integrated signals in the retina [5]. In the case of an idle display, the blur width is equal to

$$\text{Blur width}_{\text{theoretical}} = 0.8 \times \frac{v}{\text{frame rate}}$$

Therefore, blur width is inversely proportional to the frame rate. Among the solutions for the hold-type motion blur, increasing the frame rate is regarded as the most efficient method because it can directly reduce the effect of motion blur without drop in visual quality [4].

5 Frame Rate Up-Conversion

In this section, we give an overview of FRUC techniques and the details in each part. Figure 7 shows the general FRUC flow. Initially, motion vectors (MVs) between existing frames are required. In general, MVs are derived from a video decoder or by performing motion estimation (ME). The MVs can be estimated block-based, sub-block-based, or even pixel-based. Then, all derived MVs in a frame form an MV field (MVF). To display a more realistic and detailed motion, further MV processing may be performed on the MVFs. After the MVFs are retrieved, they must be mapped from the existing frames to the target intermediate frames because of temporal mismatch. Thereafter, the intermediate frames are interpolated according to the mapped MVFs using motion compensation (MC) techniques. Finally, the interpolated frames are post-processed to achieve better visual quality.

The discussions of the functions related to each part are listed below. Five parts are included: ME, MV processing, MV mapping, MC, and post-processing. In each part, we first discuss related works and then introduce several representative and state-of-the-art methods.

5.1 Motion Estimation

Illustration of motion in an image is shown in Fig. 8. A motion in 3D space is projected into the image plane and becomes a 2D vector. The 2D vector corresponding to the 3D true motion is called true MV. True ME is the method aiming to estimate the true MVs between two frames. On the other hand, conventional ME required for residual minimization is the key technique in video encoder to find out the most similar blocks for video compression. As shown in Fig. 9, both the MVs are able to be used for compression. However, unlike the ME in a video encoder, performing true ME between frames aims to determine the true motion

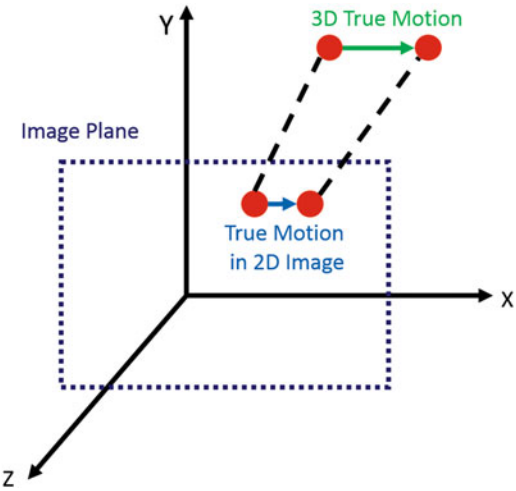


Fig. 8 Illustration of 3D motion in real world projected into image space

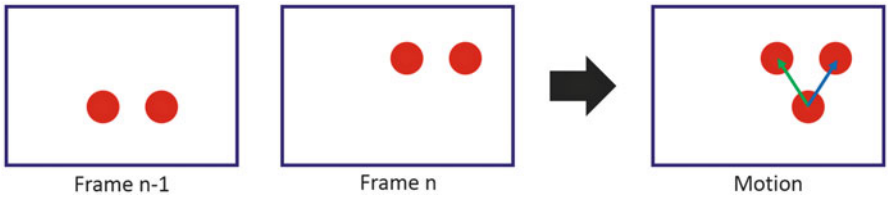


Fig. 9 Illustration of two consecutive frames with two moving red balls. This describes different goals of ME. For compression, ME using the two MVs performs the same because they both match the pixel values. However, only one MV represents the true motion of this object

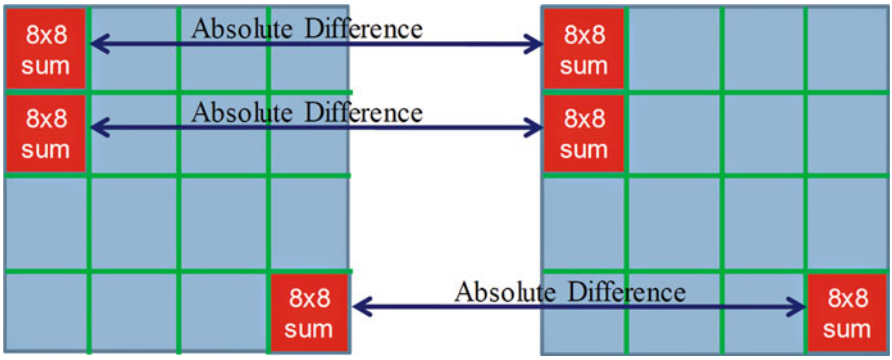


Fig. 10 An example of the 8 × 8 MSEA computation. The 32 × 32 block is composed of 16 sub-blocks, and each sub-block has 8 × 8 pixels. The absolute difference between the sub-block pairs are computed and then accumulated

that demonstrates the object movement [6], instead of reducing only the residual energy.

Many related works approximate the true MVFs using block-based ME with spatial and temporal predictions [6–9]. To achieve more accurate MVFs, Min and Sim [10] introduces a confidence level of blocks, Liu et al. [11] uses a multiple hypotheses Bayesian scheme, and Kaviani and Shirani [12] adopts optical flow estimation method. However, the true MVF is difficult to estimate, and the computational cost is usually high. On the other hand, we can possibly retrieve the MVF from a video decoder [13–15] and then perform MV processing to optimize the rough MVFs. Nevertheless, the decoding information is not always available for FRUC in the current LCD systems. We introduce three types of ME techniques in the following.

Block matching ME is a memory-efficient and hardware-friendly technique. Separating a frame into several blocks and then computing the difference between current block and neighboring blocks in the other frame. The conventional matching criterion is the sum of absolute difference (SAD). For current block A at *(x, y)*, discover the block B at $(x + s, y + t)$ minimizing the SAD computed as

$$\mathrm{SAD} = \sum_{\text{All pixels}} \left| A_{x,y} - B_{x+s,y+t} \right|$$

The vector *(s, t)* indicates the MV of the current block. To efficiently compute the difference, the down-sampled version of the SAD, multilevel successive elimination algorithm (MSEA) [16], can be adopted. An example of computing 8 × 8 MSEA between two 32 × 32 blocks is shown in Fig. 10.

To keep the robustness and consistency between neighboring blocks, the prediction and special search patterns are usually applied. A ME technique with median prediction and hybrid search pattern is introduced in [17]. The ability to reject the predictor and reestimate from the origin can prevent estimation of incorrect MVs due to incorrect predictors. In addition, the method is cost-efficient owing to its early convergence. The pseudocode is shown as follows. Here *SP* means square pattern, and ε means minimum distortion of the applied *SP*. *MV* is used as the center of *SP*, and the *threshold* is equal to 1024 in the implementation.

Step 1	Set *MV* = median of three neighboring blocks' *MVs*
Step 2	Apply *4-step SP* on *MV*
	If ε is found at the center or $\varepsilon <$ *threshold*
	Apply *2-step* and *1-step SPs* for converge
	Else
	Set *MV* = origin, go to Step. 3
Step 3	Apply *8-step SP* on *MV*
	If ε is not found at the center
	Set *MV* = the position with ε, repeat Step. 3
	Else
	Apply *4-step*, *2-step* and *1-step SPs* for converge

This search strategy is similar to a hybrid search algorithm with four-step search (4SS) [18] and three-step search (3SS) [19]. The square search pattern contains the centering block and eight neighboring blocks, and the distance between two blocks is represented by step-size. At first, a predictor is given by the median of three neighboring (left, up, and upper-right) MVs. Then a 4-step square search pattern centering on the predictor is employed. If the minimum distortion appears at the center or its value is smaller than the threshold, the predictor will be regarded as good and proceed to apply 2-step and 1-step square patterns for converge, like 3SS. Otherwise, it go back to the origin and search MV like 4SS but with an 8-step square pattern. If the minimum distortion is found at the center of an 8-step square pattern, 4-step, 2-step, and 1-step square patterns are employed for converge .

Optical flow estimation, a pixel-based ME method first proposed by Lucas and Kanade [20], can be used in place of block matching ME. This avoids blocky artifacts of block matching algorithms, but generates salt-and-pepper artifacts [21]. Recently, Lee et al. [22] proposed a FRUC framework which estimates four sets of MVF using a modified optical flow algorithm. Multiple intermediate frames are reconstructed and fused to obtain the final frame. The MVF $\boldsymbol{V_t}$ between frame $\boldsymbol{F_{t-1}}$ and frame $\boldsymbol{F_{t+1}}$ is estimated by minimizing the optical flow energy function

$$E_{\mathrm{OF}} = E_{\mathrm{OF,D}}(\boldsymbol{V_t}) + \alpha E_{OF,S}(\boldsymbol{V_t})$$

where the data term $E_{\mathrm{OF,D}}$ and smooth term $E_{\mathrm{OF,S}}$ are defined as

Fig. 11 An example of MVF estimated by optical flow estimation [23]. The MVF includes MVs of each pixel. The color coding of the MVF is explained in the evaluation method section

$$E_{\mathrm{OF,D}}(\boldsymbol{V_t}) = \int_{\Omega} \varphi\Big(|\boldsymbol{F}_{t-1}(\boldsymbol{x}) - \boldsymbol{F}_{t+1}(\boldsymbol{x})|^2 + \gamma\|\nabla \boldsymbol{F}_{t-1}(\boldsymbol{x}) - \nabla \boldsymbol{F}_{t+1}(\boldsymbol{x})\|_2^2\Big) d\boldsymbol{x}$$

$$E_{\mathrm{OF,S}}(\boldsymbol{V_t}) = \int_{\Omega} \varphi\Big(\|\nabla u_t\|_2^2 + \|\nabla v_t\|_2^2\Big) d\boldsymbol{x}$$

Here, ∇ denotes the gradient operator, and the α and γ denote the weighting parameters, respectively. Moreover, $\boldsymbol{V_t} = (u_t, v_t)$ and $\varphi(\mathrm{s}) = \sqrt{s + \varepsilon}$, where ε is a tiny constant. The data term $E_{\mathrm{OF,D}}$ preserves the intensity and gradient constancy of the data, while the smoothness term $E_{\mathrm{OF,S}}$ smooths the optical flow field. Both terms are measured in a regularized L1-norm via function φ. That is, the data term can improve the robustness to outliers, and the smoothness term can preserve motion boundaries. According to [22], the optical flow energy function is minimized by using a multilevel pyramidal scheme to cope with large displacements. However, the optical flows obtained in the previous level are often over-smoothed near motion boundaries. Therefore, the initial optical flow near motion boundaries should be refined at each level before the optimization. There are also many implementations available. Figure 11 shows an example of the pixel-based MVF estimated by [23].

Decoder-side MV extraction is an efficient way to retrieve MVF when the decoder information is available. This is often true when FRUC techniques are employed to the abovementioned low bit rate video coding application. [24] suggests that the decoder-side MVF is also useful for the FRUC techniques. Figure 12 shows an example of MVF extracted from H.264 decoder [25]. The color coding of MVF and the ground truth MVF are explained in the evaluation method section. Three different ME strategies are shown: full search (FS), fast full search (Fast FS), and enhanced predictive zonal search (EPZS). The general FS algorithm with two different block size is also shown for comparison. We can see that the MVFs extracted from H.264 decoder include accurate MVs in most regions. The further MV processing can be operated to achieve a MVF closer to the ground truth one.

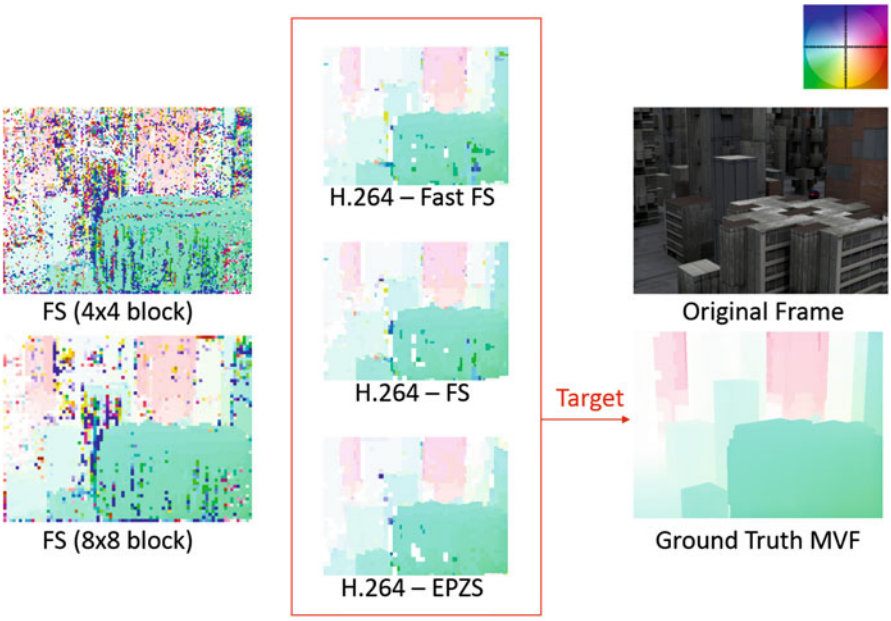


Fig. 12 An example of MVF extracted from H.264 decoder. The left two MVFs are computed using conventional full-search ME with two different block sizes. The middle three MVFs are extracted from H.264 decoder with different ME strategies. The target is to enhance the MVFs to approach the ground truth one

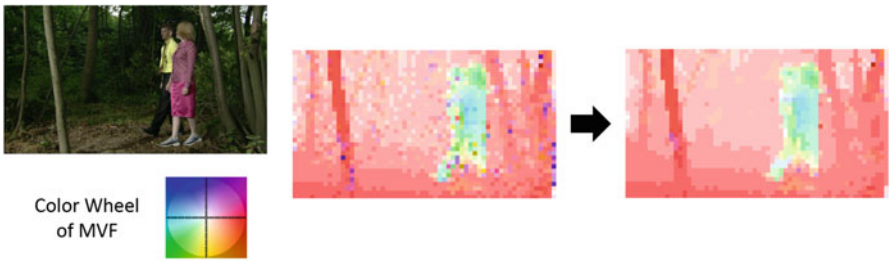


Fig. 13 Illustration of the MVF before and after MV processing. The outlier MVs are removed using MRF modeling

5.2 Motion Vector Processing

To restore the MV outliers which are misestimated as shown in Fig. 13, a further MV processing is often performed on the MVF. For example, a vector median filter [26] is often adopted owing to the important spatial and temporal coherence in true MVFs. In addition, more processing methods such as prediction-based MV

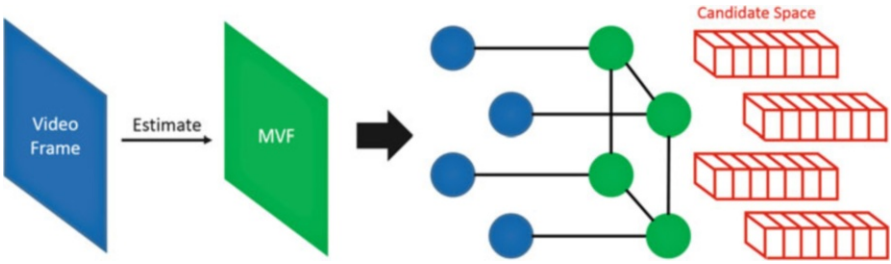


Fig. 14 Illustration of MRF modeling on MVF estimation. Estimating the MVF while observing the video frame. The edges between nodes show the connection modeled in the MRF energy function. The candidate space size depends on the algorithm

smoothing [9], motion smoothing via global energy minimization [27], and 3D Markov random field (MRF) modeling [28] are proposed to approximate the true MVF. A hardware-friendly MRF modeling method is also introduced in [17].

Vector median filter is a widely used technique to remove outlier vectors. Since MVs are vectors containing x-direction and y-direction, the filter can be applied straightforward. The median filter is a simple but effective choice. Using a 3×3, 5×5, or larger window for a block, its median MV is often highly consistent with its neighboring MVs. The median vector $\boldsymbol{v}_m$ can be derived by

$$\boldsymbol{v}_m = \underset{\boldsymbol{v}_m \in S_i}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{v}_m - \boldsymbol{v}_i\|_L$$

where i is the summation index, N is the number of members in the set, and $S_i = \{\boldsymbol{v}_i\}$ is the set of vectors. L denotes the order of the norm, and any proper norm is eligible to be used.

MRF modeling is a theoretical modeling method based on the Bayesian framework, and this modeling method has been applied to computer-vision algorithms for many years [29]. The framework can also be adopted to estimate MVF in the previous ME part as shown in Fig. 14. A node represents a pixel or block and its corresponding MV. The MVF is estimated with observed video frame. For each node, searching in the MV candidate space and choosing the MV can locally or globally minimize the MRF energy function.

A good reason to apply the framework after the ME is to reduce MV candidates to avoid heavy computation. [30] chooses only neighboring MVs as candidates to perform optimization, which can prevent over-smoothing and keep the computational complexity lower. For a block, eight neighboring MVs and the MV itself are chosen as the nine MV candidates. Thus, the corresponding MRF energy function for each MV candidate is calculated as follows,

$$\text{energy}_{\text{candidate}} = \text{cost}(\text{MV}_{\text{candidate}}) + \text{weight} \times \sum_{\forall \text{neighbor}} \left|\text{MV}_{\text{candidate}} - \text{MV}_{\text{neighbor}}\right|$$

The cost of a MV candidate can be measured in many ways such as the MSEA and optical flow data term previously introduced. The MV candidate that can minimize the MRF energy function is selected as the refined MV of the processed block as follows:

$$\text{Refined MV} = \underset{\text{MV}_{\text{candidate}}}{\text{argmin}}\ \text{energy}_{\text{candidate}}$$

5.3 *Motion Vector Mapping*

The abovementioned MVF generally represents the motion relation between two existing frames. To convert the frame rate to a higher one, new frames are interpolated between the two existing frames. The frame rate is doubled when an additional frame is interpolated as shown in Fig. 15. In this illustration, the MVF is divided by two to map the position from existing frames to the middle frame.

To our knowledge, three MV mapping methods are mainly available, and these are shown in Fig. 16. The first one is called the *traditional MV mapping* method,

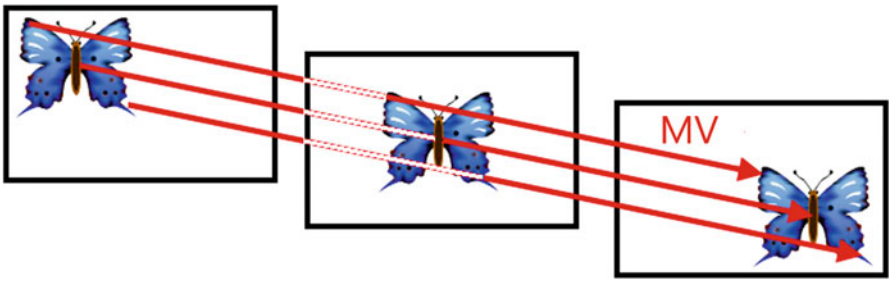


Fig. 15 Illustration of mapping MVs to the interpolated frames. Dividing the MVF by two maps the position to the interpolated frames for the twice up-converted video

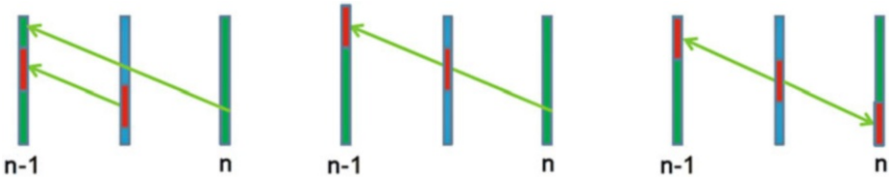


Fig. 16 Three general MV mapping methods. From left to right: traditional, forward, and bilateral MV mapping. Note that green, blue, and red represent existing frames, intermediate frames, and processing pixels/blocks, respectively

Fig. 17 The problems on overlaps and holes. The black regions are not pointed by reference pixels, which generate the holes

which maps a block MV in existing frames to a block at the same position in the intermediate frames. However, these two blocks exist at the same position but at different times; hence, their MVs are not exactly the same. The second one is called the *forward MV mapping* [31], which maps through the direction of an MV to the block where it is pointed. No temporal mismatch occurs in the use of the forward MV mapping, but in this case, some positions may be pointed out by many MVs or no MV at all, thus introducing problems on overlaps and holes. The third one is called the *bilateral MV mapping*, which performs ME on the intermediate frames [32, 33]. No problem on overlaps and holes, as mentioned earlier, but it usually fails to estimate the true MVs in flat regions. Recent researches often improve their MV mapping based on the abovementioned three mapping methods.

Forward MV mapping can avoid temporal mismatch but introduce problems on overlaps and holes as mentioned above. Figure 17 shows an example of the problems. Since the mapping operation is temporally correct and straightforward, it is widely adopted. However, to solve the problems on overlaps and holes, the further MC and post-processing techniques play important roles.

Bilateral MV mapping operation is shown in Fig. 18. The mapping method can avoid the problems on overlaps and holes because it operates on the target intermediate frame. For each block in the intermediate frame, it searches a pair of matched blocks in previous and next frames. The searching directions on two frames are opposite to match temporal domain.

5.4 Motion Compensation

Within the mapped MVF, MC is performed to interpolate the intermediate frame. The appropriate MVs are calculated as shown in Fig. 19. The adopted MC technique often depends on the ME and MV mapping techniques. For forward MV

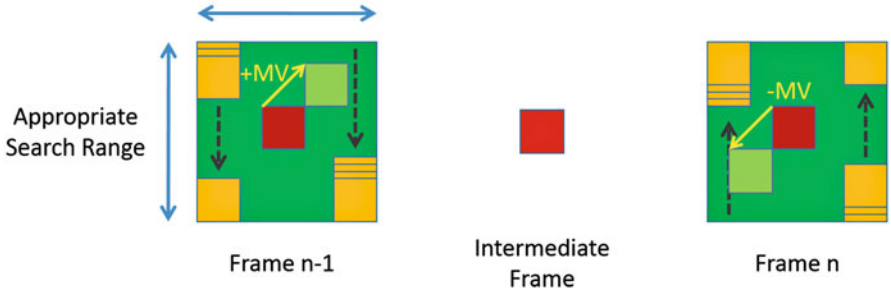


Fig. 18 The operation of bilateral MV mapping. In an appropriate search range of the previous and next frame, the blocks in the intermediate frame search for a best-matched pair of blocks

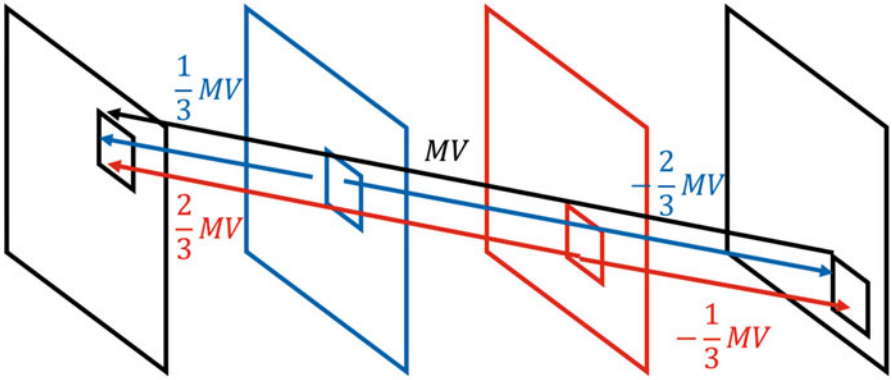


Fig. 19 MC is performed to interpolate the intermediate frame using the appropriate ratio of MVs

mapping, the problems of overlap and hole should be taken into account. For example, [34] uses mesh-based method, [12] employs patch-based reconstruction, and [30] applies block-based forward MC. For block-based operations, block artifacts should be avoided. Overlapped block MC (OBMC) [35] is a conventional method in this part and the further post-processing. Applying adaptive weighted interpolation for occlusion handling is also proposed [36]. In addition, because MC should be performed for each interpolated frame, the computational effort and hardware bandwidth consumption become significant problems when the higher frame rate is required and the number of interpolated frames increases.

Conventional *bilinear* and *adaptive weighting MC* calculate a pixel value in an interpolated frame. The pixel at the position $\boldsymbol{p}$ in frame t is computed as

$$f(\boldsymbol{p},t)=\frac{\lambda_1 f(\boldsymbol{p}-\boldsymbol{MV},t-1)+\lambda_2 f(\boldsymbol{p}+\boldsymbol{MV},t+1)}{2}$$

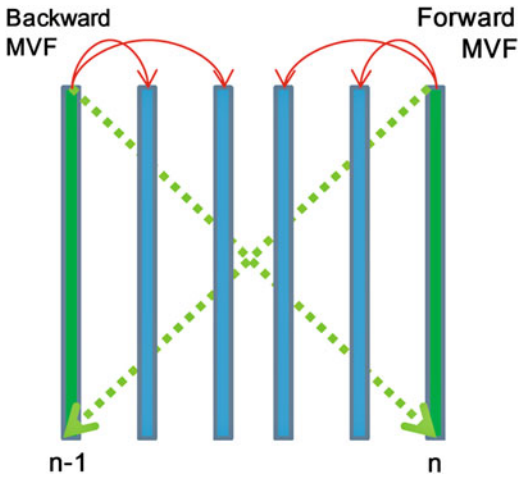


Fig. 20 Both forward and backward MVFs are estimated to achieve multirate up-conversion. The red arrows show that unidirectional interpolation is adopted for these four intermediate frames

That is, blending the pixels after taking the object motion into account. The parameters λ_1 and λ_2 can be adjusted to be adaptive weighting.

To achieve higher frame rates, the MVF should be divided by a larger number, and the MC technique should be performed more than once. However, the visual quality of up-converted frames is limited by the precision of MVF and computational complexity of the following MC. Therefore, twice up-conversion mapping is most adopted.

Estimating more MVFs creates possibilities to achieve higher frame rate. To achieve five times up-conversion, [30] performs low-complexity ME twice to retrieve the forward and backward MVFs, as indicated by the green dotted arrows in Fig. 20. Moreover, unidirectional interpolation is adopted to prevent blur and reduce complexity. In other words, the first and second intermediate frames are interpolated using the pixels in frame $n - 1$ with mapped backward MVF. Similarly, the third and fourth intermediate frames are interpolated using the pixels in frame n with mapped forward MVF.

5.5 Post-processing

The visual quality of interpolated frames always suffers due to unstable results from all of the above factors such as incorrect MVs and interpolation mismatch. Therefore, many studies choose to refine the interpolated frames via post-processing [36, 37]. Nevertheless, determining where the artifacts are and how to interpolate the blocks to obtain better visual quality are difficult.

The artifacts often happen near moving object boundaries because there exists occluded and uncovered regions. These regions cause misestimated MVs due to unmatched pixel values between two frames. [12] utilizes structure similarity and

Fig. 21 An example of possible regions with visual artifacts detected by [30]. These regions mostly reside near the boundaries of moving objects

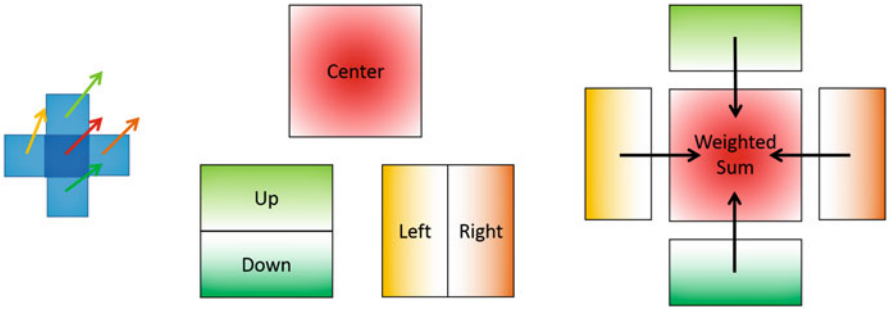


Fig. 22 Illustration of the OBMC operation. Five different colors represent five blocks. Each neighboring block contributes its half part near the processed block for blending. The more saturated colors represent the larger weighting numbers

edge information to generate a mismatch mask for further visual enhancement. [30] divides blocks into sub-blocks and takes the boundaries of the MV discontinuity as the possible regions with visual artifacts. Figure 21 shows an example of the detected result. [38] computes residual energy for blocks and then enhance their visual quality instead of finding occlusion regions.

A local refinement is often performed to enhance the visual quality of these regions. OBMC [35] and its variants are often adopted. The concept is to blend the regions with their neighboring regions by weighted sum. As shown in Fig. 22, four neighboring blocks are selected to perform OBMC on one block. We use colors to

Fig. 23 Results of post-processing to eliminate artifacts near boundaries of moving objects. The left and right images of each pair are before and after processing, respectively

represent the blocks and their weighted number, and the larger weighted numbers show more saturated color. The pixel values in these five blocks are blended by weighted sum. Performing weighted sum can achieve smooth visual quality between blocks because the pixel values across block boundaries are blended. Figure 23 shows several results before and after post-processing.

6 Evaluation Methods

To evaluate the performance of each technique, a universal dataset and measurement play important roles. In this section, we introduce the popular video datasets and evaluation criterion.

6.1 Test Video Sequences

Video coding techniques have been studied for many years. Therefore, several test video sequences are widely used in the research field [39, 40]. Figure 24 shows some snapshots of the well-known test video sequences. FRUC techniques are usually performed on these video sequences for evaluation. Since recording a video is a simple task nowadays, many researchers also have their own video sequences.

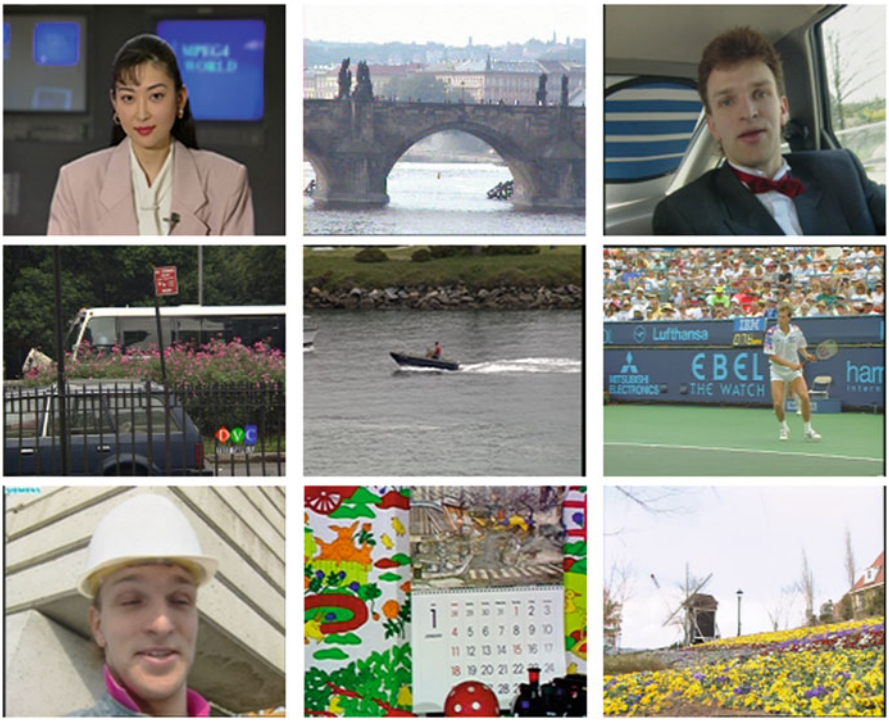


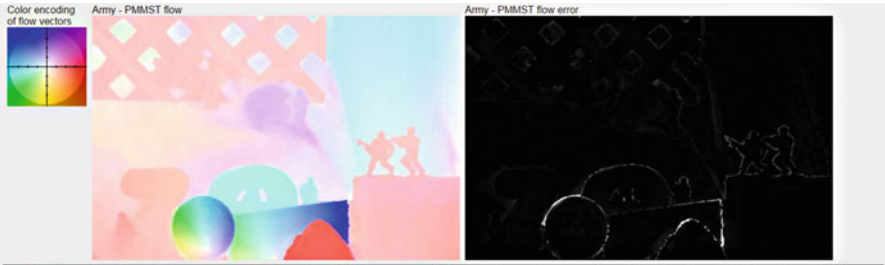
Fig. 24 An example of popular test video sequences in video coding

6.2 Motion Visualization and Evaluation

Since the estimated MVFs are crucial for FRUC, visualizing and evaluating MVFs are helpful. [41] provides a website [42] containing tools for visualizing MVF and several image pairs with ground truth MVF for evaluation. A snapshot of [42] is shown in Fig. 25. The MVFs are visualized using color coding. More specifically, a color wheel is utilized, and its center representing zero motion is white color. The ticks on the x-axe and y-axe denote a motion unit of one pixel. Therefore, the representation can visualize pixel-based MVF in floating-point precision. Note that the maximum magnitudes of the visualized MVF depend on different datasets and can be manually set.

6.3 Evaluation Criterion

The most popular evaluation method is to compute the difference between original and interpolated frames. The evaluation method using frame skipping is illustrated in Fig. 26. For example, the even frames in original video sequence are deleted, and



Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1 all	disc	untext	Mequon (Hidden texture) GT im0 im1 all	disc	untext	Schefflera (Hidden texture) GT im0 im1 all	disc	untext	Wooden (Hidden texture) GT im0 im1 all	disc	untext	Grove (Synthetic) GT im0 im1 all	disc	untext	Urban (Synthetic) GT im0 im1 all	disc	untext	Yosemite (Synthetic) GT im0 im1 all	disc	untext	Teddy (Stereo) GT im0 im1 all	disc	untext
NNF-Local [87]	3.4	**0.07** 1 0.20 3 **0.05** 1	0.15 2	0.51 4	0.12 5	0.18 2	0.37 2	0.14 2	0.10 2	0.49 5	0.06 3	**0.41** 1	**0.61** 1	0.21 2	0.23 3	0.66 3	0.19 2	0.10 5	0.12 12	0.17 13	0.34 3	0.60 6	0.23 2
PMMST [114]	9.2	0.09 32	0.21 5	0.07 16	0.18 11	0.51 4	0.16 28	0.21 9	0.42 8	0.17 16	0.10 2	**0.33** 1	0.06 12	0.51 5	0.74 4	0.28 7	0.24 3	0.65 2	0.20 5	0.11 17	0.12 12	0.17 13	0.37 4	0.74 2	0.35 4
OFLAF [77]	9.5	0.08 9	0.21 5	0.06 6	0.15 6	0.53 6	0.12 5	0.19 3	0.37 2	0.14 2	0.14 8	0.77 28	0.07 5	0.51 5	0.78 7	0.25 4	0.31 10	0.76 4	0.25 15	0.11 17	0.12 12	0.21 39	0.42 10	0.78 4	0.63 17
MDP-Flow2 [68]	10.3	0.08 9	0.21 5	0.07 16	0.15 2	**0.48** 1	**0.11** 1	0.20 5	0.40 5	0.14 2	0.15 21	0.80 36	0.08 12	0.63 19	0.93 19	0.43 20	0.26 4	0.76 4	0.23 8	0.11 17	0.12 12	0.17 13	0.38 6	0.79 5	0.44 6
NN-field [71]	11.6	0.08 9	0.22 17	**0.05** 1	0.17 8	0.55 10	0.13 11	0.19 3	0.39 4	0.15 7	**0.09** 1	0.48 4	**0.05** 1	**0.41** 1	**0.61** 1	**0.20** 1	0.52 59	**0.64** 1	0.26 18	0.13 41	0.13 35	0.20 33	0.35 3	0.83 8	**0.21** 1
ComponentFusion [96]	13.6	**0.07** 1	0.21 5	**0.05** 1	0.16 6	0.55 10	0.12 5	0.20 5	0.44 9	0.15 7	0.11 4	0.65 8	0.06 2	0.71 35	1.07 40	0.53 37	0.32 14	1.06 26	0.28 21	0.11 17	0.13 35	0.15 8	0.41 9	0.88 13	0.54 9
TC/T-Flow [76]	19.1	**0.07** 1	0.21 5	**0.05** 1	0.19 18	0.68 33	0.12 5	0.28 25	0.66 30	0.14 2	0.14 8	0.86 46	0.07 5	0.67 29	0.98 29	0.49 31	**0.22** 1	0.82 9	0.19 2	0.11 17	0.11 2	0.30 64	0.50 27	1.02 30	0.64 19
WLIF-Flow [93]	19.5	0.08 9	0.21 5	0.06 6	0.18 11	0.55 10	0.15 23	0.25 18	0.56 20	0.17 16	0.14 8	0.68 9	0.08 12	0.61 16	0.91 17	0.41 18	0.43 34	0.96 15	0.29 27	0.13 41	0.12 12	0.21 39	0.51 32	1.03 33	0.72 37
NNF-EAC [103]	21.0	0.09 32	0.22 17	0.07 16	0.17 8	0.53 6	0.13 11	0.23 11	0.49 12	0.15 7	0.16 35	0.80 36	0.09 27	0.60 13	0.89 13	0.40 16	0.39 24	0.78 6	0.28 21	0.12 30	0.12 12	0.18 25	0.57 45	1.24 49	0.69 31
Layers++ [37]	21.5	0.08 9	0.21 5	0.07 16	0.19 18	0.56 13	0.17 34	0.20 5	0.40 5	0.18 27	0.13 7	0.58 6	0.07 5	0.48 3	0.70 3	0.33 9	0.47 46	1.01 19	0.33 46	0.15 64	0.14 57	0.24 52	0.46 17	0.88 13	0.72 37
LME [70]	22.4	0.08 9	0.22 17	0.06 6	0.15 2	0.49 2	**0.11** 1	0.30 34	0.64 25	0.31 87	0.15 21	0.78 32	0.09 27	0.66 25	0.96 24	0.53 37	0.33 15	1.18 43	0.28 21	0.12 30	0.12 12	0.18 25	0.44 12	0.91 16	0.61 14
IROF++ [58]	22.5	0.08 9	0.23 24	0.07 16	0.21 31	0.68 33	0.17 34	0.28 25	0.63 24	0.19 39	0.15 21	0.73 21	0.09 27	0.60 13	0.89 13	0.42 19	0.43 34	1.08 29	0.31 36	0.10 5	0.12 12	0.12 4	0.47 19	0.96 23	0.68 30
nLayers [57]	23.2	**0.07** 1	**0.19** 1	0.06 6	0.22 39	0.59 16	0.19 55	0.25 18	0.54 16	0.20 48	0.15 21	0.84 42	0.08 12	0.53 7	0.78 7	0.34 11	0.44 38	0.84 10	0.30 32	0.13 41	0.13 35	0.20 33	0.47 19	0.97 22	0.67 28

Fig. 25 An example of visualizing and evaluating MVFs. The vectors are visualized according to their positions in the color wheel. The results by selected algorithm are shown, and the ranked algorithms are listed below

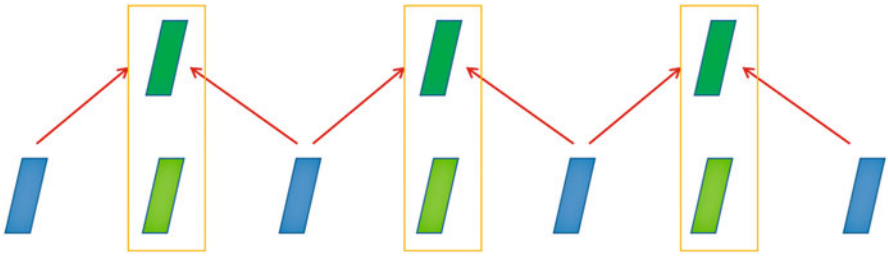


Fig. 26 Illustration of skipping frames for FRUC evaluation. The difference between skipped and interpolated frames is computed for evaluation

FRUC techniques are then performed to interpolate new even frames. The FRUC technique is evaluated based on the difference between the original and interpolated even frames.

Heinrich et al. [43] suggests an evaluation method called double interpolation without altering the original frame rate of the test video sequence. The first interpolation takes place between original frames and the second one on the interpolated result as illustrated in Fig. 27. Since the interpolation errors caused by the evaluated FRUC technique are possible to be amplified, it allows a better performance to discriminate between different FRUC methods.

The difference between frames is usually measured by the peak signal-to-noise ratio (PSNR). The PSNR between two images I^{A} and I^{B} is computed as

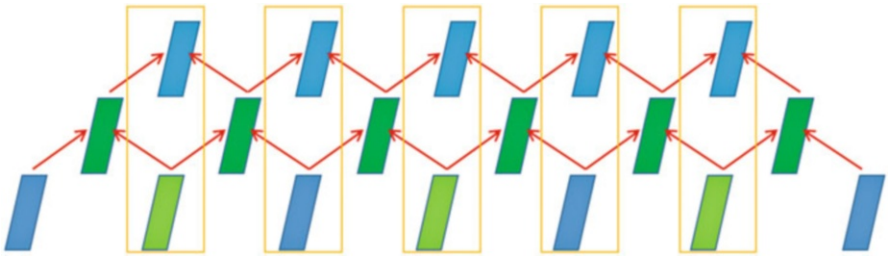


Fig. 27 Illustration of double interpolating frames for FRUC evaluation. The frames are interpolated twice and compared to the original frames for evaluation

$$\mathrm{PSNR}\left(I^{\mathrm{A}}, I^{\mathrm{B}}\right) = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right)$$

$$\mathrm{MSE} = \frac{\sum_{n=1}^{\text{Frame Size}} \left(I_n^{\mathrm{A}} - I_n^{\mathrm{B}}\right)^2}{\text{Frame Size}}$$

Although the PSNR values cannot totally represent the perceptual visual quality, frames with higher PSNR values still appear better at most times. In addition to the objective evaluation, some researchers also perform subjective evaluation to measure the perceptual visual quality. Displaying the original and processed videos, several people rate the videos. Instead, some researchers perform the structural similarity index (SSIM) [44] to evaluate the results since the index shows more relation with perceptual visual quality. The SSIM between two blocks B_1 and B_2 is defined as

$$\mathrm{SSIM}(B_1, B_2) = \frac{2(m_1 m_2 + C_1)(2\sigma_{1,2} + C_2)}{\left(m_1^2 + m_2^2 + C_1\right)\left(\sigma_1^2 + \sigma_2^2 + C_2\right)}$$

where m_i and σ_i^2 are the mean and variance of the luminance value in the corresponding block B_i, respectively. $\sigma_{1,2}$ is the covariance between values in two blocks B_1 and B_2. C_1 and C_2 are constants which stabilize the division.

7 Hardware Implementation Issues

Since FRUC becomes a crucial technique in display devices, integrating it into display systems is an efficient solution for consumer electronics. However, the required amounts of computational cost and bandwidth consumption are massive. The higher video resolution also introduces new challenges. Three challenges are encountered for the hardware architecture design. The first challenge is the requirement of a large on-chip SRAM. The on-chip SRAM arrangement is significant to

Table 1 Comparison of the FRUC hardware architecture specifications

	Kang [45]	Cetin [46]	Wang [48]	Hsu [49]	Lee [47]	Huang [30]
Technology	Xilinx Virtex 4	FPGA 90-nm	UMC 90-nm	UMC 90-nm	Altera Cyclone III	TSMC 90-nm
Clock rate (MHz)	168.33	63	200	133	148.5	300
Total gate count	6899 slices	89,000 slices	292,732	1,627,900	28,091 LEs	410,356
SRAM size (Bytes)	N/A	14,070	3036	12,365	336,444	9984
FRUC mode (FPS)	60–120	168–336	60–120	60–120	60–120	24–120 60–120
Frame size	352 × 288	1280 × 720	1920 × 1080	1920 × 1080	1920 × 1080	3840 × 2160

support a larger resolution operation. Second, because interpolating multiple frames is essential to achieve multirate FRUC, the required bandwidth consumption for the ME and MC becomes larger. Third, the cycles for data fetching must be as few as possible to achieve multirate up-conversions. Consequently, efficiently utilizing the available hardware resources is very important.

We list six hardware implementations of FRUC techniques for reference in Table 1. Three [45–47] are implemented using a field-programmable gate array (FPGA), and three [30, 48, 49] are implemented using an application-specific integrated circuit (ASIC). Despite the fact that comparison between hardware implementations is difficult, it shows that the requirement of hardware resource becomes higher. [48, 30] lower the requirement but maintain the competitive performance because they apply more efficient hardware utilization. To support the larger frame size and multirate video mode for current display devices, recent researchers should address more hardware implementation issues.

8 Conclusion and Discussion

In this chapter, we present several applications and techniques of multirate video systems. This shows that multirate property for video systems is essential, and the related research topics are important issues. We start from the fundamental FRC techniques, and selected conventional methods are presented. Then, we divide the FRUC techniques into five parts and state several popular methods of each part. The evaluation methods and hardware implementation issues are also discussed. We give a thorough presentation on the whole stage of the FRC techniques. However, many video processing techniques such as scene change detection, videotext localization, and perceptual video processing can also be applied to enhance the visual quality of converted videos. Moreover, the FRC techniques can be

cooperated with other techniques such as video coding, super resolution, and virtual view synthesis. We hope the relative techniques and implementations become well developed and enhance the video viewing experience for people.

References

1. Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the H. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(7), 560–576.
2. Sullivan, G. J., Ohm, J., Han, W. J., & Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1649–1668.
3. Poynton, C. (2012). *Digital video and HD: Algorithms and interfaces*. Amsterdam: Elsevier.
4. Pan, H., Feng, X., & Daly, S. (2005). 51.4: Quantitative analysis of LCD motion blur and performance of existing approaches. In *SID symposium digest of technical papers* (Vol. 36, No. 1, pp. 1590–1593). Blackwell Publishing Ltd, UK.
5. Pan, H., Feng, X. F., & Daly, S. (2005). LCD motion blur modeling and analysis . In *Proceedings of IEEE international conference of image processing* (*ICIP)*. (Vol. 2, pp. II–21).
6. De Haan, G., Biezen, P. W., Huijgen, H., & Ojo, O. A. (1993). True-motion estimation with 3-D recursive search block matching. *IEEE Transactions on Circuits and Systems for Video Technology, 3*(5), 368–379.
7. Wang, J., Wang, D., & Zhang, W. (2003). Temporal compensated motion estimation with simple block-based prediction. *IEEE Transactions on Broadcasting, 49*(3), 241–248.
8. Tourapis, A. M. (2002, January). Enhanced predictive zonal search for single and multiple frame motion estimation. In *Proceedings of SPIE visual communications and image processing (VCIP)* (pp. 1069–1079).
9. Kim, U. S., & Sunwoo, M. H. (2014). New frame rate up-conversion algorithms with low computational complexity. *IEEE Transactions on Circuits and Systems for Video Technology, 24*(3), 384–393.
10. Min, K. Y., & Sim, D. G. (2013). Confidence-based adaptive frame rate up-conversion. *EURASIP Journal on Advances in Signal Processing, 2013*(1), 1–12.
11. Liu, H., Xiong, R., Zhao, D., Ma, S., & Gao, W. (2012). Multiple hypotheses Bayesian frame rate up-conversion by adaptive fusion of motion-compensated interpolations. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(8), 1188–1198.
12. Kaviani, H., & Shirani, S. (2016). Frame rate up-conversion using optical flow and patch-based reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology, 26*(9), 1581–1594.
13. Yang, Y. T., Tung, Y. S., & Wu, J. L. (2007). Quality enhancement of frame rate up-converted video by adaptive frame skip and reliable motion extraction. *IEEE Transactions on Circuits and Systems for Video Technology, 17*(12), 1700–1713.
14. Huang, A. M., & Nguyen, T. Q. (2008). A multistage motion vector processing method for motion-compensated frame interpolation. *IEEE Transactions on Image Processing, 17*(5), 694–708.
15. Liu, Y. N., Wang, Y. T., & Chien, S. Y. (2011). Motion blur reduction of liquid crystal displays using perception-aware motion compensated frame rate up-conversion. In *Proceedings of IEEE workshop on signal processing systems (SiPS)* (pp. 84–89).
16. Gao, X. Q., Duanmu, C. J., & Zou, C. R. (2000). A multilevel successive elimination algorithm for block matching motion estimation. *IEEE Transactions on Image Processing, 9*(3), 501–504.

17. Chen, F. C., Huang, Y. L., & Chien, S. Y. (2012). Hardware-efficient true motion estimator based on Markov Random Field motion vector correction. In *Proceedings of IEEE international symposium on VLSI design, automation, and test (VLSI-DAT)* (pp. 1–4).
18. Li, R., Zeng, B., & Liou, M. L. (1994). A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 4*(4), 438–442.
19. Po, L. M., & Ma, W. C. (1996). A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 6*(3), 313–317.
20. Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2 (IJCAI'81)* (Vol. 2, pp. 674–679). San Francisco, CA: Morgan Kaufmann Publishers Inc.
21. Tang, C. W., & Au, O. C. (1998, May). Comparison between block-based and pixel-based temporal interpolation for video coding. In *Proceedings of IEEE international symposium on circuits and systems (ISCAS)*, (Vol. 4, pp. 122–125).
22. Lee, W. H., Choi, K., & Ra, J. B. (2014). Frame rate up conversion based on variational image fusion. *IEEE Transactions on Image Processing, 23*(1), 399–412.
23. Liu, C. (2009). *Beyond pixels: exploring new representations and applications for motion analysis*. Doctoral dissertation, Massachusetts Institute of Technology.
24. Huang, Y. L., Liu, Y. N., & Chien, S. Y. (2010, October). MRF-based true motion estimation using H. 264 decoding information. In *IEEE workshop on signal processing systems (SIPS)* (pp. 99–104).
25. H.264/AVC Software Coordination. http://iphome.hhi.de/suehring/tml/
26. Astola, J., Haavisto, P., & Neuvo, Y. (1990). Vector median filters. *Proceedings of the IEEE, 78*(4), 678–689.
27. Dane, G., & Nguyen, T. Q. (2004). Smooth motion vector resampling for standard compatible video post-processing. In *Proceedings of IEEE Asilomar conference on signals, systems and computers* (Vol. 2, pp. 1731–1735).
28. Wang, D., Zhang, L., & Vincent, A. (2010). Motion-compensated frame rate up-conversion—Part I: Fast multi-frame motion estimation. *IEEE Transactions on Broadcasting, 56*(2), 133–141.
29. Li, S. Z. (1994, May). Markov random field models in computer vision. In *European conference on computer vision* (pp. 361–370). Berlin/Heidelberg: Springer.
30. Huang, Y. L., Chen, F. C., & Chien, S. Y. (2016). Algorithm and architecture design of multi-rate frame rate up-conversion for ultra-HD LCD systems. *IEEE Transactions on Circuits and Systems for Video Technology, PP*(99), 1–1. doi:10.1109/TCSVT.2016.2596198.
31. Jeon, B. W., Lee, G. I., Lee, S. H., & Park, R. H. (2003). Coarse-to-fine frame interpolation for frame rate up-conversion using pyramid structure. *IEEE Transactions on Consumer Electronics, 49*(3), 499–508.
32. Choi, B. T., Lee, S. H., & Ko, S. J. (2000). New frame rate up-conversion using bi-directional motion estimation. *IEEE Transactions on Consumer Electronics, 46*(3), 603–609.
33. Choi, B. D., Han, J. W., Kim, C. S., & Ko, S. J. (2007). Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology, 17*(4), 407–416.
34. Min, K. Y., Ma, J. H., Sim, D. G., & Bajic, I. V. (2015). Bidirectional mesh-based frame rate up-conversion. *IEEE Multimedia, 22*(2), 36–45.
35. Orchard, M. T., & Sullivan, G. J. (1994). Overlapped block motion compensation: An estimation-theoretic approach. *IEEE Transactions on Image Processing, 3*(5), 693–699.
36. Ling, Y., Wang, J., Liu, Y., & Zhang, W. (2008). A novel spatial and temporal correlation integrated based motion-compensated interpolation for frame rate up-conversion. *IEEE Transactions on Consumer Electronics, 54*(2), 863–869.
37. Hsu, K. Y., & Chien, S. Y. (2008). Frame rate up-conversion with global-to-local iterative motion compensated interpolation. In *Proceedings of IEEE international conference on multimedia and expo (ICME)*, (pp. 161–164).

38. Huang, A. M., & Nguyen, T. (2009). Correlation-based motion vector processing with adaptive interpolation scheme for motion-compensated frame interpolation. *IEEE Transactions on Image Processing, 18*(4), 740–752.
39. Xiph.org Video Test Media. https://media.xiph.org/video/derf/
40. YUV Video Sequences. http://trace.eas.asu.edu/yuv/
41. Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision, 92*(1), 1–31.
42. Middlebury Optical Flow Dataset. http://vision.middlebury.edu/flow/
43. Heinrich, A., de Haan, G., & Cordes, C. N. (2008). A novel performance measure for picture rate conversion methods. In *Proceedings of IEEE international conference on consumer electronics (ICCE)*, (pp. 1–2).
44. Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing, 13*(4), 600–612.
45. Kang, S. J., Yoo, D. G., Lee, S. K., & Kim, Y. H. (2008, November). Hardware implementation of motion estimation using a sub-sampled block for frame rate up-conversion. In *IEEE international SoC design conference* (Vol. 2, pp. II-101).
46. Cetin, M., & Hamzaoglu, I. (2011). An adaptive true motion estimation algorithm for frame rate conversion of high definition video and its hardware implementations. *IEEE Transactions on Consumer Electronics, 57*(2), 923–931.
47. Lee, G. G., Chen, C. F., Hsiao, C. J., & Wu, J. C. (2014). Bi-directional trajectory tracking with variable block-size motion estimation for frame rate up-convertor. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), 4*(1), 29–42.
48. Wang, Y.T. (2010). *Algorithm and hardware architecture design of perception-aware motion compensated frame rate up-conversion*. Master's thesis, National Taiwan University.
49. Hsu, K. Y., & Chien, S. Y. (2011). Hardware architecture design of frame rate up-conversion for high definition videos with global motion estimation and compensation. In *Proceedings of IEEE workshop on signal processing systems (SiPS)*, (pp. 90–95).