IET Research Journals

Special Issue Paper

IET Journals
The Institution of Engineering and Technology

Motion Based Frame Interpolation for Film and Television Effects

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

A.Kokaram[1] *Davinder Singh*[1] *Simon Robinson*[2] *Damien Kelly*[3] *Bill Collis*[4] *Kim Libreri*[5]

[1] *Trinity College Dublin*
[2] *The Foundry, London*
[3] *Google Research*
[4] *Disgiuse.one, London*
[5] *EPIC Games*
* *E-mail: anil.kokaram@tcd.ie*

Abstract: Frame interpolation is the process of synthesising a new frame in-between existing frames in an image sequence. It has emerged as a key algorithmic module in motion picture effects. This paper provides a review of the technology used to create inbetween frames and presents a new Bayesian framework for designing solutions with the concept of motion interpolation. We also compare performance using the industrial toolkits used in the post production industry. We find that all successful techniques employ motion based interpolation. In addition the latest Convolutional Neural Network approaches do not outperform the explicit motion based techniques but provide complementary improvements in certain types of sequences.

1 Introduction

Frame interpolation is the process of synthesising a new frame in-between existing frames in an image sequence. It has emerged as a key algorithmic module in motion picture effects like slow motion (Slo-Mo) and Timeslice*. It is also a key component in television standards conversion used for instance in high frame rate TV sets to upsample the frame rate of received signals, and to convert between worldwide television standards e.g. EU and US formats (25 fps and 30 fps respectively). These processes revolve around the manipulation of time and space in a sequence of frames hence frame interpolation is implicated in them all.

Slow-Motion was the earliest form of time manipulation effect used to bring a kind of emphasis to fast moving objects in a scene. Two early appearances as an effect included the film The wild Bunch and the television series in the 1970's The Six Million Dollar Man. The viewer is meant to believe that the speed and significance of events is heightened by the artificially slowed down sequence. The effect was created "in camera" by shooting on set at a higher frame rate than would be used in the cinema i.e. > 24fps. Shooting at 48fps and playing back at 24fps for instance yielded motion that was 2x slower than normal. Slow-motion generated in this way has now become popular in mobile phone apps (iPhone, Casio etc) and YouTube (slo-mo guys). Today the manipulation of time can be quite flexible as illustrated in figure 1.

The TimeSlice effect, invented by Tim Macmillan in the 1990's was made famous in the series of episodes of the movie The Matrix. Known thereafter as BulletTime, it has been used as a standard effect in many movies since then. In contrast to slow motion, TimeSlice or BulletTime requires the instantaneous recording of multiple views of the scene in a number of different cameras. When those frames captured at the same time instant but from multiple views are played back as a temporal sequence, the effect of a "frozen moment in time" is conveyed and the viewer is presented with a camera path that would be impossible with a single camera. The challenge in BulletTime is to convey a sense of a smooth camera track. That is not possible by playback direct from the on set image capture, because the camera bodies are too large to get images from viewpoints close enough together. Frames therefore have to be synthesized in-between the existing views. Used in this way, frame interpolation is known in the Computer Vision community as novel view generation or viewpoint interpolation.

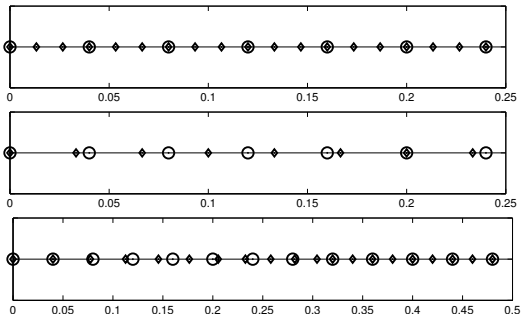

Fig. 1: *Timelines (secs) showing the relationship between input (circles, 25fps) and output frame (diamonds) locations. Conversion to 75fps (Top), 30fps (Middle) and linear deceleration (Bottom) from 25fps to 50fps.*

In both the cases of slow-motion and BulletTime the effect can be created by synthesizing frames in-between the existing images in the sequence. Hence, given a 30fps sequence, interpolating 4 frames in-between each existing pair, then playing back at 30fps, yields a 4x Slo-mo effect. In the case of BulletTime, the choice of the number of in-between frames generated depends on an artistic decision taken together with the viewpoint separation on-set. Frame interpolation has also become important for the effect of motion blur* [1] or shutter angle simulation. In that effect, several frames are interpolated close to the time instant of the actual frame and then averaged to yield the effect of a longer exposure time. This is quite an important visual effect that is associated with the film look and used regularly not only in live action shots but also in animated productions.

1.1 Motion

The simplest mechanism for converting between one frame or field (interlace) rate and another is to repeat (zero-order hold) or drop

*http://timeslicefilms.com/

*https://revisionfx.com/products/rsmb/

(subsample) frames. Thus in converting 30fps (60 fields per second) interlaced video to 25fps (50 fields per second) interlaced video we can drop 50 fields out of every 300 fields from the 30fps source. This means dropping one field every 6 fields from the source. To convert from 25fps interlaced to 30fps interlaced we can do the opposite, repeating one field in every 6 from the source. This simple technique leads to very poor quality converted pictures since one frame in every 6 would have the wrong field being merged into a frame. That leads to poorly represented motion which is perceived like a stutter effect in the converted material. A better technique is to estimate the missing field by interpolating it from the given video data. In the same time instant, for example, we can estimate an odd field from an even one by averaging lines vertically. Then we can repeat that field as needed in the next time interval, or drop the original one. Some of the earliest work on this kind of interpolation was done by the BBC R&D unit in the 1970's [2]. Since then there has been a huge body of work on this kind of non-motion compensated conversion both for interlaced and progressive format video [3]. These are all based on the general idea of repeating or dropping fields or frames accordingly.

In the motion picture effects industry frame interpolation has become synonymous with optic flow estimation. That connection was popularised by the journalist Mike Seymour in his influential **fxPhD**[*] online review in 2006 which contains an extremely comprehensive historical account of the evolution of the technology in moviemaking. The reason is because in all the effects discussed above it is essential that the interpolated frames do not disrupt the motion in the existing sequence. Hence a knowledge of the existing motion in the sequence is required. However as most researchers in this area would realise, optic flow is only a means to an end in frame interpolation. In fact to generate high quality interpolated frames requires attention to post-processing and the robust handling of poor motion estimates.

Principally because of the high level of redundancy between frames, frame interpolation can be achieved by first interpolating the motion at the time instant of the missing frame. Given that motion both into the future and the past, a new frame can be generated. The effectiveness of motion interpolation for solving this problem is actually better understood through an image sequence model as follows.

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) + \epsilon(\mathbf{x}) \quad (1)$$

where the pixel intensity in frame n at site $\mathbf{x}$ is $I_n(\mathbf{x})$ and the motion at that site from frame n into frame $n-1$ is $\mathbf{d}_{n,n-1}(\mathbf{x})$. This is the equation from which classic optic flow and motion estimation algorithms are derived. Given $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ then the pixels in the current frame n can be created by rearranging those in the previous frame $n-1$. This is also illustrated in Figure 2. Of course occlusion and uncovering plays a role, hence the need for a forward prediction model as well. Intuitively, because motion is generally smooth in space and time, we can employ motion smoothness constraints to infer motion at the missing instant in time.

Alternative strategies have proposed the notion that we can *push* image data from existing frames *into* the time instant that the data is required *along contours of least gradient* between the relevant images [4–6]. This results in convincing frame interpolation, but not necessarily along directions of motion hence the motion fidelity of the conversion is not guaranteed when the sequence is viewed at the required frame rate.

In this paper we review motion synthesis based approaches for frame interpolation and present a framework based on what would now be viewed as traditional image sequence modeling. We compare this approach to a range of existing approaches including commercial implementations of the motion synthesis idea and convolutional neural networks (CNNs). Our results show that our model based approach outperforms the most recent convolutional neural network models as well as implementations of a popular phase based technique. Commercial implementations of the motion synthesis concept all perform well in our tests including the FFMEG open source software.

[*] *https://www.fxguide.com/fxfeatured/art_of_optical_flow/*

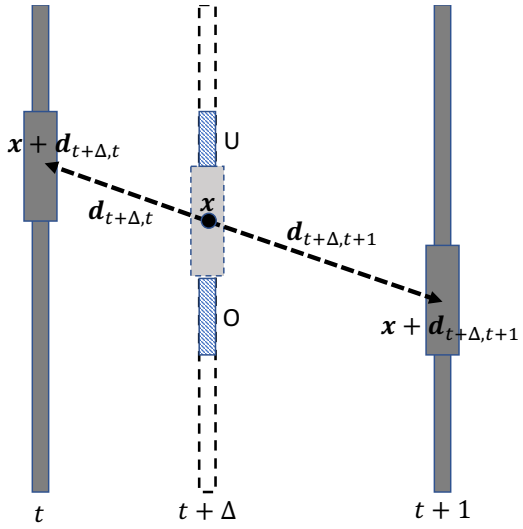

Fig. 2: *In frame interpolation, the frame at time instant $t+\Delta$ is to be interpolated from the existing frames at time instants $t, t+1$. An object is shown moving through the frames. The pixel datum at $\mathbf{x}$ in the frame at $(t+\Delta)$ can be reconstructed if we know the motion backward in time from $t+\Delta$, $\mathbf{d}_{t+\Delta,t}$, and the corresponding forward motion $\mathbf{d}_{t+\Delta,t+1}$. Regions of uncovering (U) and occlusion (O) around the object in the new frame are also shown. The interpolated frame must preserve this motion as well as the regions of occlusion and uncovering.*

2 A general formulation

Figure 3 shows four existing input frames, and one (missing) frame to be synthesised at location $t+\Delta$ in time. Proceeding in a probabilistic fashion, we require to manipulate the p.d.f. $p(I_{t+\Delta}, \mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1} | I_t, I_{t-1}, I_{t+1}, I_{t+2}, \mathbf{D}_{-\Delta})$ to generate an estimate $\hat{I}_{t+\Delta}$ of the missing frame. Here, $\mathbf{D}_{t+\Delta,\cdot}$ denotes the missing motion fields and $\mathbf{D}_{-\Delta}$ is used to denote motion fields between the *existing* frames. In this example it includes the motion field mapping frame t into $t-1$ ($\mathbf{D}_{t,t-1}$), t into $t+\Delta$ ($\mathbf{D}_{t,t+\Delta}$), $t+1$ into t ($\mathbf{D}_{t+1,t}$) and $t+1$ into $t+2$ ($\mathbf{D}_{t+1,t+2}$). We assume that $\mathbf{D}_{-\Delta}$ has been estimated between existing frames in a pre-process using any existing motion estimation technique. For the moment, we ignore the effect of occlusion and uncovering for simplicity.

Jointly solving for motion and image data is possible but more efficient algorithms result by first factorising the posterior as follows

$$p(I_{t+\Delta}, \mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1} | \cdot) = p(I_{t+\Delta} | \mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1}, \cdot) p(\mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1} | \cdot) \quad (2)$$

Maximising the two factors in turn then leads to an estimate of the missing motion, followed by the interpolated image using that estimated motion.

Considering the motion factor, and proceeding in a Bayesian fashion we can write

$$p(\mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1} | \cdot) \propto p_{LD}(\cdot | \mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1}) p_{PD}(\mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1}) \quad (3)$$

where $p_{LD}()$ is the motion likelihood and $p_{PD}()$ the motion prior. $p_{LD}()$ ensures that the interpolated motion explains the observed image frames and the observed motion $\mathbf{D}_{-\Delta}$ in the rest of the sequence. p_{PD} inserts information about the spatial and temporal

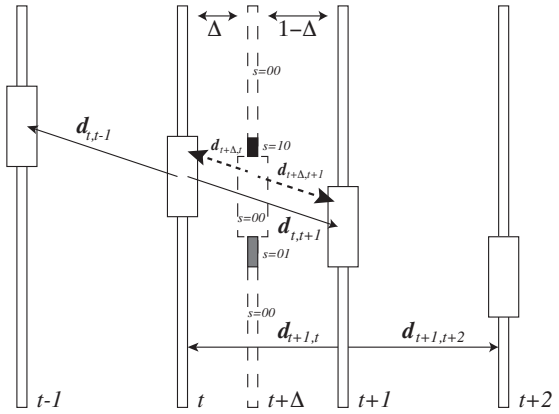

Fig. 3: *Four existing frames at $t-1, t, t+1, t+2$ are shown, as well as the missing frame at $t+\Delta$ (dashed). The missing frame is offset from t by Δ of a time interval and hence from $t+1$ by $1-\Delta$. An object is shown translating between the frames, and the correct manifestation for that object at $t+\Delta$ is shown in the dashed rectangle. The motion $\mathbf{d}_{t,t-1}$ between frames at t and $t-1$ is shown using one example motion vector as is the forward motion $\mathbf{d}_{t,t-1}$. Motion $\mathbf{d}_{t+1,t}, \mathbf{d}_{t+1,t+2}$ is also shown using a vector in the background of frame $t+1$ as an example. The correct interpolated motion at one position in $t+\Delta$ is shown as dashed arrows. The association of each position in frame $t+\Delta$ with an occlusion state s is also indicated in the strip representing $t+\Delta$. For instance, in the region $s=10$, the interpolated image data does not exist in frame t but it does in frame $t+1$. The other states $s=00, 01$ are illustrated similarly.*

smoothness of motion. Liu et al [7] and Wang et al [8] both also formulate the problem in a similar Bayesian fashion but go on to include only spatial priors and operate only on a block basis.

The missing image p.d.f. follows a similar pattern.

$$p(I_{t+\Delta}|\mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1}, \cdot) \propto p_{LI}(\mathbf{D}_{t+\Delta,t}, \mathbf{D}_{t+\Delta,t+1}, \cdot|I_{t+\Delta}) p_{PI}(I_{t+\Delta}) \quad (4)$$

where again $p_{LI}(),\ p_{PI}()$ are respectively the likelihood and prior distributions for the missing image data.

The different approaches to motion based interpolation proposed in the past can be explained as resulting from particular choices for the various likelihood and prior terms above. We use this framework to review the previous work next.

3 Previous work

We may divide the historical evolution of this area into the development of model based algorithms and learning algorithms. Learning based algorithms have evolved rapidly since about 2010 but before this model based algorithms were the most common.

3.1 Motion interpolation

Deinterlacing and field rate conversion work in the early 1980's [3, 9, 10] all employed more than two fields (usually 3 or 4) to generate the missing motion field. Deinterlacing will not be discussed further here, but it is important to note that the idea of motion interpolation originated there. Most of the early work can be derived by specifying the motion likelihood as a function of observed or measured motion between existing frames as follows.

$$p_{LD}(\cdot|\mathbf{d}_{t+\Delta,t}(\mathbf{x}), \mathbf{d}_{t+\Delta,t+1}(\mathbf{x})) \propto \exp -\frac{1}{2\sigma_t^2}\left[||\mathbf{d}_{t+\Delta,t} - \mathbf{d}_b|| + ||\mathbf{d}_{t+\Delta,t+1} - \mathbf{d}_f||\right] \quad (5)$$

where $\mathbf{d}_b, \mathbf{d}_f$ are estimated from the observed motion between frame t and $t+1$ and σ_t^2 measures the confidence in the model to some extent. The main body of work uses a Maximum likelihood approach for motion interpolation in the sense that no explicit prior for the interpolation is employed i.e. $p_{PD}(\cdot)$ is constant. In detail, however, since $\mathbf{d}_b, \mathbf{d}_f$ are derived from $\mathbf{d}_{t,t+1}$ and/or $\mathbf{d}_{t+1,t}$ there is an implicit notion of temporal motion smoothness. Assuming no acceleration across the frames it is typical [5, 8, 10–18] to state $(1-\Delta)\mathbf{d}_{t+\Delta,t} = \Delta\mathbf{d}_{t+\Delta,t}$. Hence the ML estimates for the interpolated motion become $\hat{\mathbf{d}}_{t+\Delta,t} = \mathbf{d}_b$ and $\hat{\mathbf{d}}_{t+\Delta,t+1} = \mathbf{d}_f$. Note that the complication of motion between the frames is not treated consistently in the literature. In fact, the observed motion should be compensated for motion itself before being used as a constraint in the motion interpolation process. This is not always the case in previous work. For example in Jiang et al [19] the initial motion interpolant is not derived from compensated motion.

By far the majority of work after about 2000, uses a symmetric motion constraint to estimate the forward and backward interpolated motion fields directly from the observed image data [7, 8, 20–29]. The first example of its use is probably as far back as 1996 Castagno et al [30]. This is is variously referred to in the literature as *bidirectional, bilateral, or symmetric* motion estimation. The likelihood in this case is expressed as a function of observed image frames as follows.

$$p_{LD}(\cdot|\mathbf{d}_{t+\Delta,t}(\mathbf{x}), \mathbf{d}_{t+\Delta,t+1}(\mathbf{x})) \propto \exp -\frac{1}{2\sigma_e^2}\left[(I(\mathbf{x} = \mathbf{d}_{t+\Delta,t}(\mathbf{x})) - I(\mathbf{x} + \mathbf{d}_{t+\Delta,t+1}(\mathbf{x})))\right] \quad (6)$$

Hence estimation of the interpolated motion becomes a problem of motion estimation or optic flow estimation from a location $\mathbf{x}$ situated at the *interpolated* time instant $t+\Delta$. Most of the previous work use a block matching scheme of some description to estimate this motion. This entails searching for matching blocks in frame $t, t+1$ using typically the sum absolute error as the objective function E_{SAD} defined as follows.

$$E_{\text{SAD}}(\mathbf{d}_{t+\Delta,t}(\mathbf{x})) = \sum_{\mathbf{x}\in\mathcal{B}} |I(\mathbf{x} + \mathbf{d}_{t+\Delta,t}(\mathbf{x})) - I(\mathbf{x} + \mathbf{d}_{t+\Delta,t+1}(\mathbf{x}))| \quad (7)$$

where β denotes the sites in a block of pixels. Despite the wide success of these techniques leading to use in real time applications like frame rate upconversion in television sets, they do not explicitly incorporate motion smoothness in time.

3.2 Motion compensated frame interpolation

Having estimated an interpolated motion field it remains to actually synthesise the interpolated frame. All previous work at this stage acknowledges that the motion interpolated in the previous step will contain errors and so some kind of robust interpolation process is necessary. Furthermore, occlusion and uncovering tend to be taken in to account in this step rather than in the motion interpolation step.

Taking a maximum likelihood approach to image interpolation, ignoring occlusion, and using the simple sequence model in equation 1 leads to the simple weighted temporal averaging operation proposed as early as 1989 [31]. Hence the interpolated pixel at

$\hat{I}_{t+\Delta}(\mathbf{x})$ is as follows.

$$\hat{I}_{t+\Delta}(\mathbf{x}) = w_1 I(\mathbf{x} + \hat{\mathbf{d}}_{t+\Delta,t}(\mathbf{x})) + w_2 I(\mathbf{x} + \hat{\mathbf{d}}_{t+\Delta,t+1}(\mathbf{x})) \quad (8)$$

where $w_1 = (1-\Delta), w_2 = \Delta$.

For robustness to erroneous motion estimates, or occlusion and uncovering, the two main approaches taken were either i) generate a number of candidate image interpolants and filter them with a non-linear filter [13, 20, 31] or ii) use OBMC (overlapped block motion compensation) [21, 22]. Candidate interpolated pixels included typically the non-motion compensated frame average, the estimate as shown previously and pixels in a support neighbourhood in the previous and next frames suitably compensated for motion. Overlapped block motion compensation is a robust scheme for implicitly dealing for many kinds of motion compensation errors by combining blocks in the previous and next frames based on some kind of motion reliability weight. For example, using four possible motion candidates (perturbed from the ML estimate above) we can measure the block similarities between I_t and I_{t+1} (typically using SSIM or MSE). The similarity scores can be normalised and then used as filter weights together with spatial window functions for generating the interpolated block in the new frame. In a sense, the work by Wang et al [16, 17, 32] falls into this category because they employ bilateral filters in space and time (trilateral filtering).

Thoma and Bierling [10] were perhaps the first to explicitly deal with occlusion and uncovering in a frame interpolation context. In their work however, they were not strictly dealing with creating a new interpolated frame but instead they were predicting existing frames in a sequence for the purposes of video compression. Expressing their idea in the context of this paper, we can introduce an occlusion state variable $S = \{00, 01, 10\}$ to indicate no-occlusion and occlusion in the next and previous frames respectively. See Figure 3. Their interpolation process can then be described as follows

$$\hat{I}_{t+\Delta}(\mathbf{x}) = \begin{cases} [(1-\Delta) I_t(\mathbf{x} + \hat{\mathbf{d}}_{t+\Delta,t}(\mathbf{x})) + \\ \quad \Delta I_{t+1}(\mathbf{x} + \hat{\mathbf{d}}_{t+\Delta,t+1}(\mathbf{x}))] & : s = 00 \\ I_{t-1}(\mathbf{x} + \mathbf{d}_{t+\Delta,t-1}(\mathbf{x})) & : s = 01 \\ I_{t+1}(\mathbf{x} + \mathbf{d}_{t+\Delta,t+1}(\mathbf{x})) & : s = 10 \end{cases} \quad (9)$$

where $\Delta = 0.5$ in their work. Because they were predicting existing frames, they were able to use motion compensated frame differences to configure s by thresholding the forward and backward motion compensated frame differences. Since then several authors have established Bayesian approaches for estimating these occlusion indicators [33–36] in the context of motion estimation and missing data interpolation.

Phase based interpolation deserves special mention [37]. That work is based on the idea of expressing the model in Equation 1 in the Fourier frequency space by taking the Fourier transform of both sides as follows.

$$\mathcal{F}_{I_n}(\omega_1, \omega_2) = \exp -j\Big[d_1\omega_1 + d_2\omega_2\Big] \mathcal{F}_{I_{n-1}}(\omega_1, \omega_2) + \mathcal{F}_{\epsilon}(\omega_1, \omega_2) \quad (10)$$

Since the early 1990's this was used to express the motion estimation problem in the phase space [12, 38, 39]. As can be seen the problem of motion estimation now becomes that of estimating the phase shift $j\Big[d_1\omega_1 + d_2\omega_2\Big]$ between the Fourier transform of image patches in different frames. Meyer et al [37] recognised that for small motion magnitudes between frames, the problem of frame interpolation using motion compensation can be reposed as the interpolation of the Fourier phase between the frames. This leads to elegant and robust frame interpolation results when motion is relatively small. We use this as a baseline algorithm in comparative results at the end of this paper.

3.3 Neural Network approaches

The most exciting development in recent times has been the deployment of Deep Learning for frame interpolation [19, 40–44]. Using a database of high frame rate sequences, a Deep Neural Network (DNN) is configured to correctly interpolate the high frame rate sequences from different subsampled sequences. The DNN acts as a powerful non-linear adaptive filter in this case. Much of this work builds on the use of DNNs for motion or optical flow estimation [45–47]. In all approaches using DNNs thus far, only two frames are used at $t, t+1$ to interpolate the intermediate frames, and the architectures have to be retrained for use with interpolated frames at different Δ.

Two broad trends for the image synthesis have emerged. The first trend is exemplified by Niklaus et al [42, 43] who design their DNN to estimate two convolution kernels $H_f(\mathbf{x}), H_b(\mathbf{x})$ that change from site to site for the estimation of the intermediate frame as follows.

$$\hat{I}_{t+\Delta} = H_b(\mathbf{x}) \circledast I_t + H_f(\mathbf{x}) \circledast I_{t+1} \quad (11)$$

This is a generalisation of the motion compensated weighted frame averaging filter in equation 8 and in many ways related to the use of spatio-temporal linear predictors [23, 34] for frame synthesis. Occlusion is implied because the kernel changes at every site and for forward and backward frames.

The other trend is to follow the existing motion based pipeline in the sense that first motion is interpolated using a DNN and then the frames are synthesised with that motion [19, 40, 44]. Zhang et al [40] actually use bilateral motion estimation and weighted frame averaging to generate a first estimate for the intermediate frame. Their DNN acts as a post-processor to clean up artifacts. Jiang et al [19] directly use a DNN to estimate w_1, w_2 in equation 8 as well as motion in a two step refinement process. In the first step a DNN is used to estimate motion between the frames at $t, t+1$ in both directions. That motion information is copied into the intermediate frame location (without motion compensation) to generate an initial interpolated motion field $\hat{\mathbf{d}}_{t+\Delta,t}, \hat{\mathbf{d}}_{t+\Delta,t-1}$, and associated interpolated image $\hat{I}_{t+\Delta}$ using weighted frame averaging. This initial estimate is input to a DNN to generate refinements to the motion field as well as visibility maps w_1, w_2. These maps act as soft occlusion indicators. This updated data is used in equation 8 to generate the final interpolated frame.

The reader is directed to the papers above for a detailed description of the network architectures. It is sufficient to note here that in all cases, the standard U-Net convolutional architecture is deployed. In addition, both L1 loss and perceptual loss functions have been investigated for training and authors report that the perceptual loss functions appear to yield networks that generate sharper results. In addition, Jiang et al [19] introduce smoothness loss functions on the motion interpolated and a temporal image smoothness loss between the motion compensated frames I_t, I_{t+1}. The results reported so far show the motion refinement strategy outperforming the kernel estimation strategy. However the kernel estimation strategy is certainly more general even though the size of the kernels limit the amount of motion it can handle.

4 The proposed algorithm

Recall that we require to first interpolate motion and then perform reconstruction. In our proposed process we wish to explicitly incorporate temporal smoothness of motion and image data as well as occlusion. This interpolation process is a *pull* process rather than *push*, since given the motion at $t+\Delta$ we can *pull* pixels from $t, t+1$ to create the image $I_{t+\Delta}$. We assume that some existing motion estimation process has been used to estimate motion between the four existing frames shown in figure 3.

Following from equation 2, we proceed in a Bayesian fashion by manipulating the posterior probability distribution, $p(\mathbf{d}_{t+\Delta,t+1},\ \mathbf{d}_{t+\Delta,t}|\mathbf{D},\ \mathbf{i})$ (where $\mathbf{D},\ \mathbf{i}$ contain all the existing

(known) motion estimates and image data) as follows.

$$\begin{aligned} p(\mathbf{d}_{t+\Delta,t+1}(\mathbf{x}),\ \mathbf{d}_{t+\Delta,t}(\mathbf{x}),\ s(\mathbf{x})|\mathbf{D},\ \mathbf{i}) \\ = p(\mathbf{i}|\mathbf{d}_{t+\Delta,t+1}(\mathbf{x}),\ \mathbf{d}_{t+\Delta,t}(\mathbf{x}),\ s(\mathbf{x}))\times \\ p(\mathbf{d}_{t+\Delta,t+1}(\mathbf{x}),\ \mathbf{d}_{t+\Delta,t}(\mathbf{x})|\mathbf{D},\ s(\mathbf{x}))\times \\ p(\mathbf{d}_{t+\Delta,t}|\mathbf{d}_{-(\mathbf{x})},\ s(\mathbf{x}))\times \\ p(\mathbf{d}_{t+\Delta,t+1}|\mathbf{d}_{-(\mathbf{x})},\ s(\mathbf{x}))\times p(s(\mathbf{x})) \end{aligned} \quad (12)$$

where $\mathbf{d}_{-(\mathbf{x})}$ collects the motion in the interpolated frame in the neighbourhood of the current site. The estimate for $\mathbf{d}_{t+\Delta}$, used as the interpolated motion, is therefore that which maximises the posterior in expression above. To continue, we need to define the likelihood and priors in that expression.

4.1 Image Likelihood

This expression is derived directly from the image model given in equation 1. We define $e_I(\mathbf{x}) = I_t(\mathbf{x}+\mathbf{d}_{t+\Delta,t}) - I_{t+1}(\mathbf{x}+\mathbf{d}_{t+\Delta,t+1})$, which is the motion compensated pixel difference between the pixel in the next frame and the pixel in the previous. For color images e_I is a vector of 3 differences corresponding to the three color planes. If the interpolated motion is correct this difference should be small, unless occlusion occurs. It is more robust to use motion to explicitly incorporate $s(\cdot)$ rather than the image data, since the data at $t+\Delta$ is clearly not known a-priori. We still incorporate $s(\cdot)$ explicitly in this likelihood as follows.

$$p(\mathbf{i}|\mathbf{d}_{t+\Delta,t+1}(\mathbf{x}),\ \mathbf{d}_{t+\Delta,t}(\mathbf{x})) \propto \begin{cases} \exp -\frac{e_I^2(\mathbf{x})}{2\sigma_I^2} & : s(\mathbf{x}) = 00 \\ \exp -k_I & : \text{Otherwise} \end{cases} \quad (13)$$

where $k_I = 10 \times 2.7^2$ to allow for a strong bias away from occlusion in the image data. In color images e_I^2 is the scaled vector magnitude i.e. the average of the square of the three difference components. σ_I^2 can be measured from the pixel data but we set it to 1.0.

4.2 Motion Likelihood

The true interpolated motion should provide smooth trajectories when compared with the motion already estimated between the existing frames. We enforce this constraint by encouraging various motion compensated *motion* differences to be small. We define these motion compensated motion differences as follows (dropping the $\mathbf{x}$ argument in $\mathbf{d}_{t+\Delta}$ for clarity).

$$e_f = |\mathbf{d}_{t+\Delta,t+1} - (1-\Delta)\mathbf{d}_{t+1,t+2}(\mathbf{x}+\mathbf{d}_{t+\Delta,t+1})| \quad (14)$$

$$e_b = |\mathbf{d}_{t+\Delta,t} - \Delta\mathbf{d}_{t,t-1}(\mathbf{x}+\mathbf{d}_{t+\Delta,t})| \quad (15)$$

$$e_{fb} = |\mathbf{d}_{t+\Delta,t+1} + (1-\Delta)\mathbf{d}_{t+1,t}(\mathbf{x}+\mathbf{d}_{t+\Delta,t+1})| \quad (16)$$

$$e_{bf} = |\mathbf{d}_{t+\Delta,t} + \Delta\mathbf{d}_{t,t+1}(\mathbf{x}+\mathbf{d}_{t+\Delta,t})| \quad (17)$$

$$e_d = \left|\frac{\mathbf{d}_{t+\Delta,t}}{1-\Delta} + \frac{\mathbf{d}_{t,t+1}}{\Delta}\right| \quad (18)$$

In these expressions e_{bf} and e_{fb} are energies that penalise deviation from the mirror constraint for motion. e_d is a direct manifestation of the constraint used in symmetric or bilateral motion [22, 30]. Finally e_f, e_b constrain the estimated forward and backward motion to provide low acceleration into the future and past frames at $t+2, t-1$ respectively.

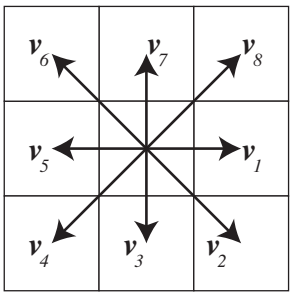

Fig. 4: $\mathbf{v}_k$ *defines the eight nearest neighbouring sites around the current site* $\mathbf{x}$ *in the middle of the* 3×3 *site grid.*

Incorporating $s(\cdot)$ then yields the following expressions for the motion likelihood.

$$p(\mathbf{d}_{t+\Delta,t+1}(\mathbf{x}),\ \mathbf{d}_{t+\Delta,t}(\mathbf{x})|\mathbf{D}, s(\cdot)) = \begin{cases} \exp -\frac{e_f^2+4\alpha+2e_d^2}{2\sigma_d^2} & : s(\mathbf{x}) = 10 \\ \exp -\frac{e_b^2+4\alpha+2e_d^2}{2\sigma_d^2} & : s(\mathbf{x}) = 01 \\ \exp -\frac{e_{fb}^2+e_{bf}^2+e_f^2+e_b^2+2e_d^2}{2\sigma_d^2} & : s(\mathbf{x}) = 00 \end{cases} \quad (19)$$

where α represents penalty energies needed to balance the loss of temporal continuity when in occluded states $10,\ 01$ and discourage the occurrence of these states. e_d also penalizes motion vector pairs which show any acceleration. Thus the motion likelihood for state $s(\cdot) = 00$ encourages the interpolated motion to align with all existing motion between frames $t, t-1;\ t, t+1;\ t+1, t+2$. In the other states $(01,\ 10)$ temporal smoothness is only encouraged with motion between $t, t-1$ and $t+1, t+1$ respectively.

4.3 Motion Priors

We assume that the motion fields are all Markov Random Fields. The motion prior consists of two factors, $p_d(\cdot)$ which enforce spatial smoothness of the estimated motion field in both forward and backward drections. Hence we employ the usual Gibbs energy prior as follows.

$$p_d(\mathbf{d}_{t+\Delta,t}|\mathbf{d}_{-(\mathbf{x})}) \propto \exp -\Lambda_d \left\{ \sum_{k=0}^{K-1} \lambda_k f(|\mathbf{d}_{t+\Delta,t}(\mathbf{x}) - \mathbf{d}_{t+\Delta,t}(\mathbf{x}+\mathbf{v}_k)|) \right\} \quad (20)$$

and similarly for motion in the opposite direction. In this expression Λ_d controls the strength of the smoothness ($\Lambda_d = 2.0$ typically), and λ_k weights the contribution from each of the clique terms inversely with their distance from $\mathbf{x}$ i.e. $\lambda_k = 1/|\mathbf{v}_k|$. We use $K = 8$, to index the eight nearest neighbors of the current site with $\mathbf{v}_k$. These offset vectors all have unit values in the horizontal and vertical directions. This relationship is shown in Figure 4. In addition, $f(\cdot)$ is a robust function defined as follows.

$$f(a) = \begin{cases} a & : |a| < 10.0 \\ 10.0 & : \text{Otherwise} \end{cases} \quad (21)$$

4.4 Occlusion Priors

Finally, the prior for occlusion $p(s(\cdot))$ encourages spatial smoothness in the estimated states

$$p_s(s|s_{-(\mathbf{x})}) \propto \exp -\Lambda_o \left\{ \sum_{k=0}^{K-1} \lambda_k h(s, s(\mathbf{x}+\mathbf{v}_k)) \right\} \quad (22)$$

where $h(s_1,\ s_2)$ is an energy function that assigns energies according to the state pairs $(s_1,\ s_2)$ as follows

$$h(s_1,\ s_2) = \begin{cases} 0 & : s_1 = s_2 \\ 1 & : s_1 = 00 \\ 1 & : s_2 = 00 \\ 2 & : \text{Otherwise} \end{cases}$$

This heavily discourages states $01,\ 10$ from sharing a boundary, while encouraging the states to be the same in local neighborhoods. We use $\Lambda_o = 10.0$ and λ_k is as defined previously.

5 Optimisation

Using the expressions above we can solve for the unknown motion $\mathbf{d}_{t+\Delta}$ by manipulating equation 12 using optimization techniques such as Graph Cuts, Belief Propagation or any other local update scheme. In order to reduce the computational load of the final system we take a more pragmatic route. We propose local candidates for the interpolated motion using simpler temporal motion prediction techniques, and then employ the expressions above to select the best candidates at each site. In addition we jointly estimate for motion and occlusion rather than estimating for each in turn. This process is iterated until conclusion. The approach can be considered as an application of the well known Iterated Conditional Modes (ICM) optimization algorithm combined with local importance sampling. To facilitate this candidate generation step there are three preamble steps which are described next.

5.1 Motion estimation

Motion fields are pre-computed between frame pairs $t, t-1;\ t, t+1;\ t+1, t;\ t+1, t+2$. Any motion estimation process can be used e.g. block matching or optic flow methods. These pre-computed motion fields are used to initialize $\mathbf{d}_{t,t-1},\ \mathbf{d}_{t,t+1},\ \mathbf{d}_{t+1,t-1},\ \mathbf{d}_{t+1,t+2}$ respectively. These fields are not changed during the interpolation process.

5.2 Temporal Hit List

At each site during the iterative process, temporal and spatial candidates are required. To reduce the computational load, we precompute a list of possible temporal motion candidates. These candidates are estimated based on the observation that we can predict the motion at the interpolated locations by copying the motion between existing frames along their motion directions into the pixel locations at $t + \Delta$. This is motion prediction in time. Hence each motion vector between frames $t,\ t+1$ is used to predict possible vectors (candidates) for the interpolated field $\mathbf{d}_{t+\Delta,\ t+1}$. Similarly $\mathbf{d}_{t+1,\ t}$ is used to predict possible vectors for $\mathbf{d}_{t+1,\ t+\Delta}$. Figure 5 illustrates the mechanism. The process is as follows

1. Scan every vector $\mathbf{d}_{t,\ t+1}(\mathbf{x})$ for all $\mathbf{x}$ in frame t
2. At each site $\mathbf{x} + \Delta\mathbf{d}_{t,\ t+1}(\mathbf{x})$ in frame $t + \Delta$, keep a record of $\mathbf{d}_{t,\ t+1}(\mathbf{x})$ since it has caused a 'hit' at that site. These are *forward* hits.

This process is repeated for the reverse direction, to yield backward hits, starting from frame $t + 1$ using $\mathbf{x} + (1 - \Delta)\mathbf{d}_{t+1,\ t}(\mathbf{x})$ instead. The result is two co-located lists of candidate temporal vectors (pointing in the forward and backward temporal directions) for every site in the interpolated frame at $t + \Delta$. In practice, because of inaccuracies in the pre-computed motion fields and the difficulty in handling occlusion, there will be several sites at which there is more than one hit in each list, or no hits in some lists. A no hit case can be seen in Figure 5.

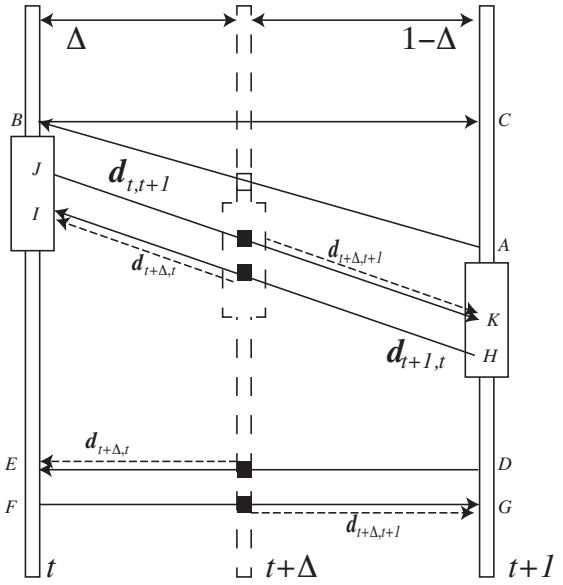

Fig. 5: *Showing how the temporal hit list at $t + \Delta$ is created using motion fields $\mathbf{d}_{t,.}, \mathbf{d}_{t+1,.}$. Sites in $t + \Delta$ at which there are hits from both sides are shown as black squares, and one example of a site showing one hit (in the $t + 1, t$ direction) is shown as a white square. The backward vector from $D \rightarrow E$ yields a temporal candidate for $\mathbf{d}_{t+\Delta,t}$ and $F \rightarrow G$ yields a similar hit in the opposite direction and hence a candidate for $\mathbf{d}_{t+\Delta,t+1}$. Similar situations exist for JK and HI. However the vector AB yields a hit in the backward direction (white square) but there are no vectors BA that would yield a hit on the other side. Instead, all the vectors starting at B would be expected to map to C. This is an indication of an occluded region.*

5.3 Generating an initial estimate

It is possible to initialize the interpolated motion and occlusion information by random assignment. However, given the temporal list above it is preferable to generate a quick initial estimate of the interpolated motion field using the following rules derived from a consideration of the ideal case shown in Figure 5. Here $N_T^b(t, \mathbf{x})$ indicates the number of temporal candidates (hits) in the backward direction and $N_T^f(\mathbf{x})$ for the forward direction.

1. Scan all sites in $t + \Delta$
2. If $(N_T^b(\mathbf{x}) == 1)\&\&(N_T^f(\mathbf{x}) == 1)$ assign the motion in the lists to the interpolated motion and set $s = 00$.
3. If $(N_T^b(\mathbf{x}) \geq 1)\&\&(N_T^f(\mathbf{x}) == 0)$ assign the first motion hit in the backward direction to both directions of interpolated motion and set $s = 10$.
4. If $(N_T^b(\mathbf{x}) == 0)\&\&(N_T^f(\mathbf{x}) \geq 1)$ assign the first motion hit in the forward direction to both directions of interpolated motion and set $s = 01$.
5. Otherwise set the interpolated motion to 0 and $s = 00$.

6 The algorithm

The final algorithm is a local pixel update scheme derived from a MAP estimate of the missing motion and video frame. First the following pre-computation steps are performed.

1. Generate Motion for existing frames. See section 5.1.
2. Generate Temporal Hit List. See section 5.2.
3. Generate initial estimate. See section 5.3.

Then at each pixel site $\mathbf{x}$ the following local updates are executed.

1. Fetch temporal hit candidates in forward and backward directions using the hit list compiled previously. If there are none then this list is empty.
2. Fetch the motion at the 8 nearest neighbors of the current site and use them as 8 motion candidates for forward and backward directions.
3. Include the current motion information at the current site as a final candidate.
4. Reduce the length of the forward and backward motion candidate lists using a primitive vector quantiser : remove vectors which are within ± 0.25 pixels. Denote this new candidate list of vectors as $\mathbf{d}_k^f, \mathbf{d}_k^b$ for the kth forward and backward candidate. Assume that this yields K candidate pairs.
5. For each pair of motion candidates, generate three possible motion/occlusion candidates by augmenting each pair with the three possible states $s = 00, 01, 10$. Denote this new candidate set as $\mathbf{m}_k^1 = [\mathbf{d}_k^f, \mathbf{d}_k^b, s = 00]$, $\mathbf{m}_k^2 = [\mathbf{d}_k^f, \mathbf{d}_k^b, s = 01]$ $\mathbf{m}_k^3 = [\mathbf{d}_k^f, \mathbf{d}_k^b, s = 10]$. There are now $3 \times K$ motion candidates.
6. For each of the $3K$ motion candidates calculate the following energies

$$E_s^b = \Lambda_d \Big\{ \sum_{k=0}^{K-1} \lambda_k f(|\mathbf{d}_k^b(\mathbf{x}) - \mathbf{d}_{t+\Delta,t}(\mathbf{x} + \mathbf{v}_k|) \Big\}$$

$$E_s^f = \Lambda_d \Big\{ \sum_{k=0}^{K-1} \lambda_k f(|\mathbf{d}_k^f(\mathbf{x}) - \mathbf{d}_{t+\Delta,t+1}(\mathbf{x} + \mathbf{v}_k|) \Big\}$$

$$E_i = (I_t(\mathbf{x} + \mathbf{d}_k^b(\mathbf{x})) - I_{t+1}(\mathbf{x} + \mathbf{d}_k^f(\mathbf{x})))/(2\sigma_I^2)$$

$$E_x = E_s^b + E_s^f + (2e_d^2/(2\sigma_d^2))$$

$$E_{00} = E_x + e_{fb}^2 + e_{bf}^2 + e_b^2 + e_f^2 + \Lambda_o \Big\{ \sum_{k=0}^{K-1} \lambda_k h(00, s(\mathbf{x} + \mathbf{v}_k)) \Big\}$$

$$E_{01} = E_x + e_b^2 + 4\alpha + \Lambda_o \Big\{ \sum_{k=0}^{K-1} \lambda_k h(01, s(\mathbf{x} + \mathbf{v}_k)) \Big\}$$

$$E_{10} = E_x + e_f^2 + 4\alpha + \Lambda_o \Big\{ \sum_{k=0}^{K-1} \lambda_k h(10, s(\mathbf{x} + \mathbf{v}_k)) \Big\}$$

where $e_f, e_b, e_{fb}, e_{bf}, e_d$ are defined in equation 18. We use $\Lambda_o = 10.0, \Lambda_d = 2.0$ and λ_k is as defined previously.
7. Assign the motion candidate pair having the lowest total energy to the interpolated motion field, replacing the values currently in that field. For that candidate the state value s is indicated by the energy which is minimum i.e. if E_{00} is minimum, then $s = 00$ etc.
8. Continue to the next site unless all sites have been visited.
9. To reduce the occurrence of impulsive single site artifacts as the algorithm proceeds, detect all occurrences of sites at which $s(\mathbf{x}) \neq s(\mathbf{v}_k + \mathbf{x})$ and $s(\mathbf{v}_k + \mathbf{x})$ are all the same. Then replace $s(\cdot)$ with the value of the neighbors. In addition, replace the motion at the site with the average motion of its neighbors.
10. Terminate iterations when i) there has been no change in any estimated motion or ii) there have been R iterations (e.g. $R = 5$) of passes over the sites.
11. Build the interpolated picture with the estimated motion and occlusion states, using equation 9.

6.1 Multiresolution

The process above alone is not robust to large motion. Hence a multiresolution scheme is sensible. In such a scheme the algorithm as outlined above is applied to a coarse block based motion field instead. Thus each *site* becomes a block of $B \times B$ pixels ($B = 4$ for 1080p images in our case). Site image differences then become the average pixel intensity difference. The interpolated block motion field at the coarse level is then used to initialize iterations at the next finer level. We use 4 levels of decimation in our pyramid. At the highest scale, when the iterations are complete, we use the block based vector field as the final interpolated motion field.

6.2 Post Processing

Due to difficulty in estimating motion when that motion is fast, or the recording was taken in low light, a post-processing step is employed to reduce the appearance of image artifacts. A post-processing step like this is inevitable in a production environment because of the huge variety of material that will have to be dealt with. Artifacts typically appear as holes in the image $I_{t+\Delta}$, or strange warping of the image near large occluded or uncovered regions. The idea is to locate these low confidence image estimates and blend them seamlessly with the average of the future and past frames. In a sense this is similar to the use of visibility maps or soft occlusion maps by Jiang et al [19]. We use the gradient of the motion field (choosing forward or backward direction depending on which is greater) as the measure of confidence in the interpolation. The steps are as follows.

1. Generate a conservative estimate for the interpolated frame using averaging $I^*(\mathbf{x}) = (1 - \Delta)I_t(\mathbf{x}) + \Delta I_{t+1}(\mathbf{x})$
2. For simplicity, define the backward interpolated motion $\mathbf{d}_{t+\Delta,t-1}(\mathbf{x})$ as $[\hat{d}_1^b(h,k),\ \hat{d}_2^b(h,k)]$ and the forward direction similarly: $\mathbf{d}_{t+\Delta,t}(\mathbf{x}) = [\hat{d}_1^f(h,k),\ \hat{d}_2^f(h,k)]$, and where $\mathbf{x} = [h,k]$. Hence measure the motion gradient $g_m(\mathbf{x})$ at each site $\mathbf{x}$ and blending weight $w(\mathbf{x})$ as follows.

$$g_m(\mathbf{x}) = \delta_{hh} + \delta_{hv} + \delta_{vh} + \delta_{vv} \tag{23}$$

$$w(\mathbf{x}) = \begin{cases} 0 & : g_m(\mathbf{x}) > \delta_t \\ 1 & : \text{Otherwise} \end{cases} \tag{24}$$

where $\delta_t = 4$ and δ_{xx} are the differentials of the horizontal and vertical components of motion in the horizontal h and vertical directions v. We use the maximum of the forward and backward gradients at each site.
3. Calculate the final output picture using $\hat{I}(\mathbf{x}) = w(\mathbf{x})I^*(\mathbf{x}) + (1 - w(\mathbf{x}))I_{t+\Delta}(\mathbf{x})$. This is a weighted blend between the non-motion compensated average picture I^* and the output picture from the previous stage $\hat{I}_{t+\Delta}$.

7 Results

We use the Adobe240fps dataset [19] to compare performance across a range of frame interpolation algorithms. It consists of 133 clips of varying durations (155 to 11297 frames), all at 720p resolution. There are 148,815 frames in total. The clips contain consumer derived content from iPhones or GoPros and do not represent the high quality material normally used in post-production for cinema and television. Nevertheless they represent a useful baseline of performance.

In the results that follow the algorithm presented in this paper is labelled ACKMRF. Three very important and industrially popular retimers are also compared : FFMPEG, Twixtor (revisionfx.com) and Kronos/Nuke (thefoundry.com). FFPMEG is open source and used as substrate infrastructure in all the streaming media services available today (Netflix, YouTube, Vimeo, Facebook to name a few). That frame interpolator uses bidirectional motion estimation followed by OBMC [22, 48]. Twixtor and Kronos represent two of the most important post-production software suites in use by professional post-production houses today. We acknowledge that these are commercial implementations and therefore we do not know exactly the algorithms that they implement but we do know that they are both variants of motion based interpolation as discussed here. In fact Kronos is a commercial implementation of the algorithm presented here which includes a more robust initial motion estimate. Taken as a bundle (FFMPEG, Twixtor, Kronos) these would represent the best that industrial strength motion based synthesis can provide now.

Using the open source software provided by Niklaus et al [42] and Jiang et al [19] we were also able to compare against the two most effective DNN based approaches to date. Niklaus et al provide their DNN model at (`github.com/sniklaus/pytorch-sepconv`),

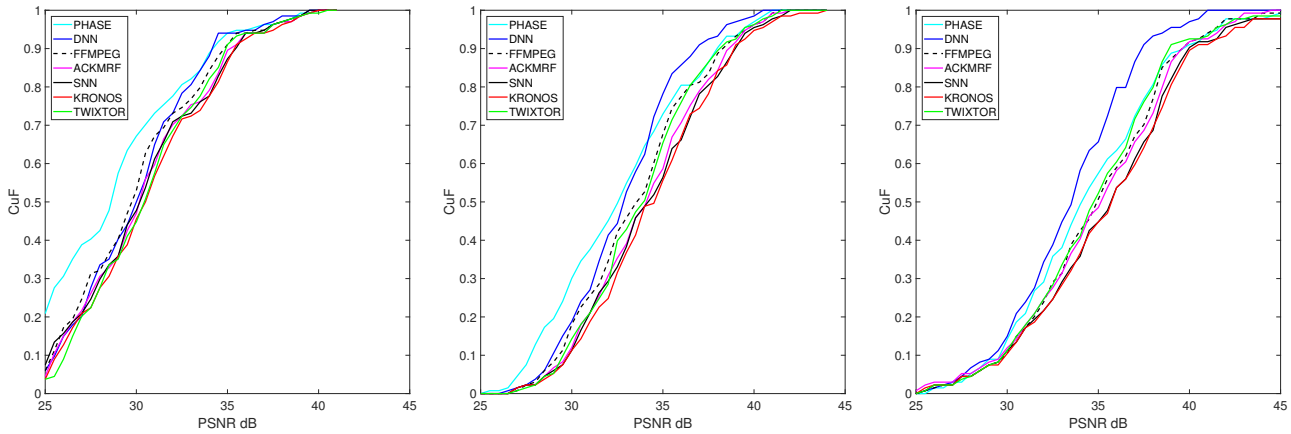

Fig. 6: *Cumulative density wrt PSNR for the Adobe240fps dataset. Left to right* $30 \rightarrow 60$ *fps,* $60 \rightarrow 120$ *fps,* $120 \rightarrow 240$ *fps,*

while the model from Jiang et al was a third party implementation `github.com/avinashpaliwal/Super-SloMo`. In the results that follow these implementations are labelled as `SNN`, `DNN` respectively.

Method	$30 \rightarrow 60$	$60 \rightarrow 120$	$120 \rightarrow 240$
PHASE	28.94	32.89	34.68
DNN	30.18	33.09	33.65
FFMPEG	30.18	33.82	35.13
ACKMRF	30.58	34.31	35.18
SNN	30.56	34.46	35.69
KRONOS	**30.85**	**34.67**	**35.83**
TWIXTOR	30.75	34.02	35.13

The table above shows the mean PSNR for each method used to upsample all 133 sequences by a factor of 2, starting from 30fps, 60fps and 120fps respectively. We create the sequences by downsampling the original 240fps sequences by 2, 4, 8 respectively. As expected, the performance of all methods used to upsample from 30fps is worse than upsamping from 120fps. That is because the sequence becomes significantly temporally aliased at 30fps and motion is larger. KRONOS yields the best result overall, rising from 30.85 dB to 35.83 dB at 120 fps. Tests of significance (t-test) reveals the surprising observation that across all the upsampling experiments none of the differences between ACKMRF, SNN, KRONOS and TWIXTOR are significant at the 95% level across any of the experiments. At $30 \rightarrow 60$ fps DNN and FFMPEG join that group in the sense that the differences between the means are also not significant at the 95% level. The PHASE method is the worst performer except at 120fps when it swaps places with DNN. DNN is otherwise the worst performer.

To try to tease out further what is happening, figure 6 shows the cumulative distribution of PSNRs across all the sequences for every method. The increase in quality is illustrated by a drift of all the curves to the right as the frame rate of the original sequence increases. Note how closely grouped all the methods are except PHASE, DNN, FFMPEG at 30fps. It shows that the temporal aliasing of the sequence is a challenge to all the methods. It also shows that the implicit modeling of motion in SNN with large kernel sizes is almost keeping up with the explicit motion modeling of the other methods. The fact that DNN is the worst at 120fps though indicates something awry with motion handling there since that is an explicit motion handling method.

Figure 7 ranks the performance of each method per sequence and provides a bar chart to summarise that performance. This analysis approach tries to separate the performance of the techniques based on content. For every method we measure how many times it is ranked 1st, 2nd, 3rd ... 7th. Then we histogram that performance. The ideal performer would show zeros for bars 1-6 and 100% for a single red bar indicating that it was always ranked 1st and never any other rank. The worst performer would show a single blue bar at 100% indicating that it was always last. The top chart shows 30fps, middle 60fps, and bottom 120fps. Now we can see that KRONOS is always ranked 1st the most number of times, SNN and ACKMRF is ranked 2nd and 3rd the most number of times, while TWIXTOR has the best performance at 30fps. Overall though KRONOS, TWIXTOR, SNN are the only three that appear to rank 1st for at least 15 % of the sequences. Taken together, these numerical results show that the CNN methods are incorporating information that is complementary to the explicit motion modeling of the other methods.

7.1 Visual Inspection

Figures 8, 9, 10, 11 enable Visual inspection for a few frames to illustrate some subtle observations. This is a useful exercise because the rankings of the methods between sequences is not consistent. Figure 8 shows that large amounts of material appearing and disappearing off the edges of the frame are an issue for the CNN based methods. The PHASE method simply does not perform well when the motion is large but no doubt a more robust motion compensated version of that algorithm should perform well. The explicit motion based methods perform the best especially in this case of global/perspective motion.

Figure 9 shows that many of the differences between the methods disappear when the motion is small i.e. at high original frame rates. This makes sense because at that frame rate the temporal aliasing is substantially reduced. Nevertheless, PHASE still struggles even at high frame rate presumably because of poor motion handling.

Figure 10 shows that all the methods struggle with complicated object interaction. The thin top of the bollard poses a challenge for all the techniques as it is not attached to either the moving background or the stationary foreground with confidence. In the visual effects workflow this kind of problem would be dealt with by segmenting the sequence into different moving layers and then retiming each layer separately.

In Figure 11 SNN outperforms the other techniques (by 1.0dB in this sequence) because it captures the motion of the thin bicycle wheel very well. It is unclear why it can perform in this way in this scenario but fails in others. One possibility is the behaviour of the background which in this case is static and uniform.

8 Conclusion

Retiming tools are an important part of cinema production and television applications. Most of the genres of techniques in the literature currently all use explicit motion information in some way to perform interpolation. We have presented a Bayesian approach for the interpolation of video frames using motion. Unlike previous work we have incorporated temporal motion smoothness constraints across more frames and strengthened the link between motion and occlusion in interpolation. A fast scheme has been presented to initialise

the final motion flow field before optimisation. Extensive tests are performed over more than 14000 frames of video using high frame rate (240fps) to provide ground truth for interpolation. The CNN methods recently developed are shown to perform in the same range as the motion based technique presented here but does not perform as well as industrial strength motion based retimers used in production now. While we do not expect any recent research to outperform existing production tools that have been tried and tested in the wild, the relatively small advantage of current CNN methods shows that hybrid schemes are the future. These kinds of schemes may perhaps be derived by considering a CNN as a post-processor for an explicit motion based technique.

9 Acknowledgments

This work was partially funded by donations from Google/YouTube. Many thanks go to Damien Kelly, Hugh Denman, Andrew Crawford and Francis Kelly for important discussions about motion estimation and motion interpolation over the years at GreenParrotPictures.

10 References

1 Brooks, T., Barron, J.T. 'Learning to synthesize motion blur'. In: IEEE Conference on Vision and Pattern Recognition (CVPR). (, 2019.
2 Couteur, G.M.L.: 'Field store standards conversion: A technical description of the c06/508 converter', *BBC Research and Development Report*, 1972, **BBC RD 1972/22**
3 de Haan, G.: 'Video processing for multimedia systems'. (ISBN 90-9014015-8, 2000)
4 Mahajan, D., Huang, F.C., Matusik, W., Ramamoorthi, R., Belhumeur, P.: 'Moving gradients: a path-based method for plausible image interpolation', *ACM Trans Graph*, 2009, **28**, (3), pp. 42:1–42:11
5 Werlberger, M., Pock, T., Unger, M., Bischof, H. 'Optical flow guided tv-l1 video interpolation and restoration'. In: Proceedings of the 8th international conference on Energy minimization methods in computer vision and pattern recognition. EMMCVPR'11. (Springer-Verlag, 2011. pp. 273–286
6 Linz, C., Lipski, C., Magnor, M.A. 'Multi-image interpolation based on graph-cuts and symmetric optical flow'. In: ACM SIGGRAPH 2010 Posters. SIGGRAPH '10. (ACM, 2010. pp. 129:1–129:1
7 Liu, H., Xiong, R., Zhao, D., Ma, S., Gao, W.: 'Multiple hypotheses bayesian frame rate up-conversion by adaptive fusion of motion-compensated interpolations', *IEEE Transactions on Circuits and Systems for Video Technology*, 2012, **22**, (8), pp. 1188–1198
8 Wang, Y., , Ma, S., Gao, W.: 'Frame rate up conversion via bayesian motion estimation', *Proceedings of SPIE - The International Society for Optical Engineering*, 2010, **7744**
9 Tekalp, M.: 'Digital Video Processing'. (Prentice Hall, 1995)
10 Thoma, R., Bierling, M.: 'Motion compensating interpolation considering covered and uncovered background', *Signal Processing: Image Communication*, 1989, **1**, (2), pp. 191 – 212. 64 kbit/s Coding of Moving Video. Available from: http://www.sciencedirect.com/science/article/pii/092359658990009X
11 Kim, J.H., Kyung, C.M.: 'Fast frame-to-frame interpolation technique for scenes containing moving objects', *Electronics Letters*, 1991, **27**, (20), pp. 1788–1790
12 Thomas, G.A.: 'A comparison of motion compensated interlace to progressive conversion methods', *BBC Research and Development Report*, 1996, **BBC RD 1996/9**
13 Ojo, O.A., de Haan, G.: 'Robust motion-compensated video upconversion', *IEEE Transactions on Consumer Electronics*, 1997, **43**, (4), pp. 1045–1056
14 Chen Yen-Kuang, Vetro, A., Huifang, S., Kung, S.Y. 'Frame-rate up-conversion using transmitted true motion vectors'. In: IEEE Second Workshop on Multimedia Signal Processing. 12. (, 1998. pp. 622–627
15 Nieweglowski, J., Moisala, T., Haavisto, P. 'Motion compensated video sequence interpolation using digital image warping'. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 5. (, 1994. pp. 205–208
16 Hoangvan, X., Ascenso, J., Pereira, F.: 'Improved matching criterion for frame rate upconversion with trilateral filtering', *Electronics Letters*, 2013, **49**, (2), pp. 106 –107
17 Hoang Van, X.: 'Statistical search range adaptation solution for effective frame rate up-conversion', *IET Image Processing*, 2018, **12**, (1), pp. 113–120
18 Dane, G., Nguyen, T.Q.: 'Optimal temporal interpolation filter for motion-compensated frame rate up conversion', *IEEE Transactions on Image Processing*, 2006, **15**, (4), pp. 978–991
19 Jiang, H., Sun, D., Jampani, V., Yang, M., Learned-Miller, E., Kautz, J. 'Super slomo: High quality estimation of multiple intermediate frames for video interpolation'. In: IEEE Conference on Computer Vision and Pattern Recognition. (, 2018. pp. 9000–9008
20 Lee, S., Yang, S.: 'Adaptive motion-compensated frame rate up-conversion', *Electronics Letters*, 2002, **38**, (10), pp. 451–452
21 Kang, S., Yoo, S., Kim, Y.H.: 'Dual motion estimation for frame rate up-conversion', *IEEE Transactions on Circuits and Systems for Video Technology*, 2010, **20**, (12), pp. 1909–1914
22 Choi, B., Han, J., Kim, C., Ko, S.: 'Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation', *IEEE Transactions on Circuits and Systems for Video Technology*, 2007, **17**, (4), pp. 407–416
23 Zhang, Y., Zhao, D., Ji, X., Wang, R., Chen, X. 'A spatio-temporal autoregressive frame rate up conversion scheme'. In: 2007 IEEE International Conference on Image Processing. vol. 1. (, 2007. pp. I – 441–I – 444
24 Shih-Hsuan Yang, Chia-Chi Yang. 'Fast frame rate up-conversion based on multiple frames'. In: 2011 IEEE International Conference on Multimedia and Expo. (, 2011. pp. 1–4
25 Hung, K.W., Siu, W.C. 'Fast video interpolation/upsampling using linear motion model.'. (, 2011. p. 1341
26 Bai, W., Liu, J., Ren, J., Guo, Z. 'Visual-weighted motion compensation frame interpolation with motion vector refinement'. In: 2012 IEEE International Symposium on Circuits and Systems (ISCAS). (, 2012. pp. 500–503
27 Dao, M., Suo, Y., Chin, S., Tran, T. 'Video frame interpolation via weighted robust principal component analysis'. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. (, 2013. pp. 1404–1408
28 Lee, K., Jeong, J.: 'Bilateral frame rate up-conversion algorithm based on the comparison of texture complexity', *Electronics Letters*, 2016, **52**, (5), pp. 354–355
29 Guo, D., Lu, Z.: 'Motion-compensated frame interpolation with weighted motion estimation and hierarchical vector refinement', *Neurocomputing*, 2016, **181**, pp. 76 – 85. big Data Driven Intelligent Transportation Systems
30 Castagno, R., Haavisto, P., Ramponi, G.: 'A method for motion adaptive frame rate up-conversion', *IEEE Transactions on circuits and Systems for Video Technology*, 1996, **6**
31 Haavisto, P., Juhola, J., Neuvo, Y.: 'Fractional frame rate up-conversion using weighted median filters', *IEEE Transactions on Consumer Electronics*, 1989, **35**, (3), pp. 272–278
32 Wang, C., Zhang, L., He, Y., Tan, Y.: 'Frame rate up-conversion using trilateral filtering', *IEEE Transactions on Circuits and Systems for Video Technology*, 2010, **20**, (6), pp. 886–893
33 Kokaram, A. 'Joint detection, reconstruction of severely degraded image sequences'. In: Image Analysis and Processing, 9th International Conference on Image Analysis and Processing (ICIAP '97). (, 1997. pp. 773–780
34 Kokaram, A.: 'Motion Picture Restoration: Digital algorithms for artefact suppression in archived images and video'. (Springer Verlag ISBN 3-540-76040-7, 1998)
35 Konrad, J., Dubois, E.: 'Bayesian estimation of motion vector fields', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, **14**, (9), pp. 910–927
36 Yalcin, H., Black, M.J., Fablet, R. 'The dense estimation of motion and appearance in layers'. In: 2004 Conference on Computer Vision and Pattern Recognition Workshop. (, 2004. pp. 165–165
37 Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A. 'Phase-based frame interpolation for video'. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (, 2015. pp. 1410–1418
38 Dabner, S.C.: 'Real time motion measurement hardware: Phase correlation unit', *BBC Research and Development Report*, 1990, **BBC RD 1990/11**
39 Biswas, M., Nguyen, T. 'A novel motion estimation algorithm using phase plane correlation for frame rate conversion'. In: Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers. vol. 1. (, 2002. pp. 492–496
40 Zhang, Y., Chen, L., Yan, C., Qin, P., Ji, X., Dai, Q.: 'Weighted convolutional motion-compensated frame rate up-conversion using deep residual network', *IEEE Transactions on Circuits and Systems for Video Technology*, 2018, **12**, pp. 1–1
41 Kappeler, A., Yoo, S., Dai, Q., Katsaggelos, A.K.: 'Video super-resolution with convolutional neural networks', *IEEE Transactions on Computational Imaging*, 2016, **2**, (2), pp. 109–122
42 Niklaus, S., Mai, L., Liu, F. 'Video frame interpolation via adaptive separable convolution'. In: 2017 IEEE International Conference on Computer Vision (ICCV). (, 2017. pp. 261–270
43 Niklaus, S., Mai, L., Liu, F. 'Video frame interpolation via adaptive convolution'. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (, 2017. pp. 2270–2279
44 Lu, C., Hirsch, M., Schölkopf, B. 'Flexible spatio-temporal networks for video prediction'. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 7. (, 2017. pp. 2137–2145
45 Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T. 'Flownet 2.0: Evolution of optical flow estimation with deep networks'. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (, 2017. pp. 1647–1655
46 Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., et al. 'Flownet: Learning optical flow with convolutional networks'. In: IEEE International Conference on Computer Vision (ICCV). (, 2015. pp. 2758–2766
47 Ranjan, A., Black, M.J. 'Optical flow estimation using a spatial pyramid network'. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (, 2017. pp. 2720–2729
48 Jiefu Zhai, Keman Yu, Jiang Li, Shipeng Li. 'A low complexity motion compensated frame interpolation method'. In: 2005 IEEE International Symposium on Circuits and Systems (ISCAS). vol. 5. (, 2005. pp. 4927–4930

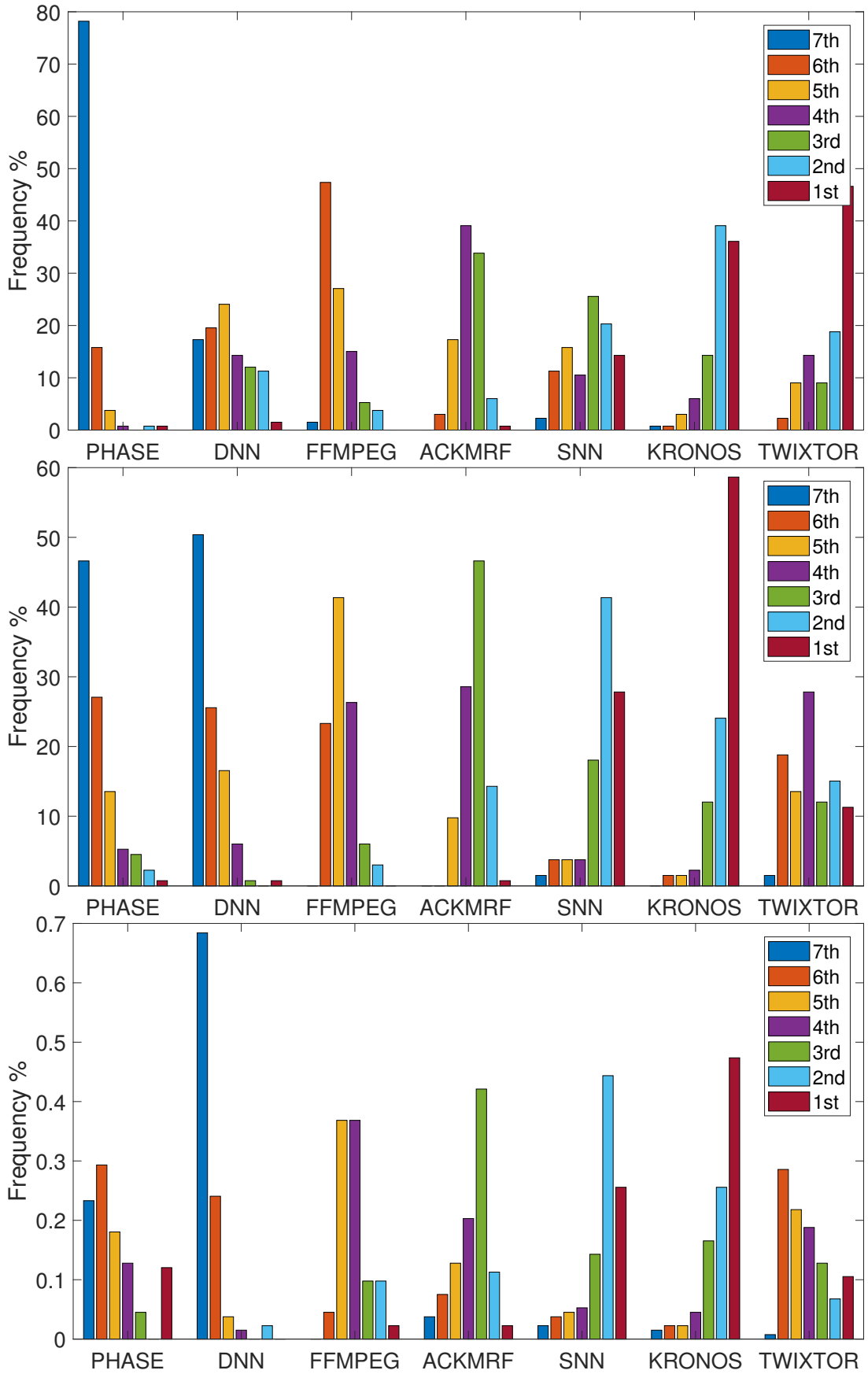

Fig. 7: *Ranking the performance of each method over all 133 clips. Top to bottom :* $30 \rightarrow 60$*fps,* $60 \rightarrow 120$*fps,* $120 \rightarrow 240$*fps. There are 8 groups of 8 bars each. Each group of bars indicates the number of times each method is ranked 7th, 6th, 5th etc from left to right in each group (as a fraction out of 133). Kronos ranks first the most number of times, followed by SNN (2nd th emost number of times) and ACKMRF (3rd the most number of times). In general performance improves as the original frame rate is higher as expected.*

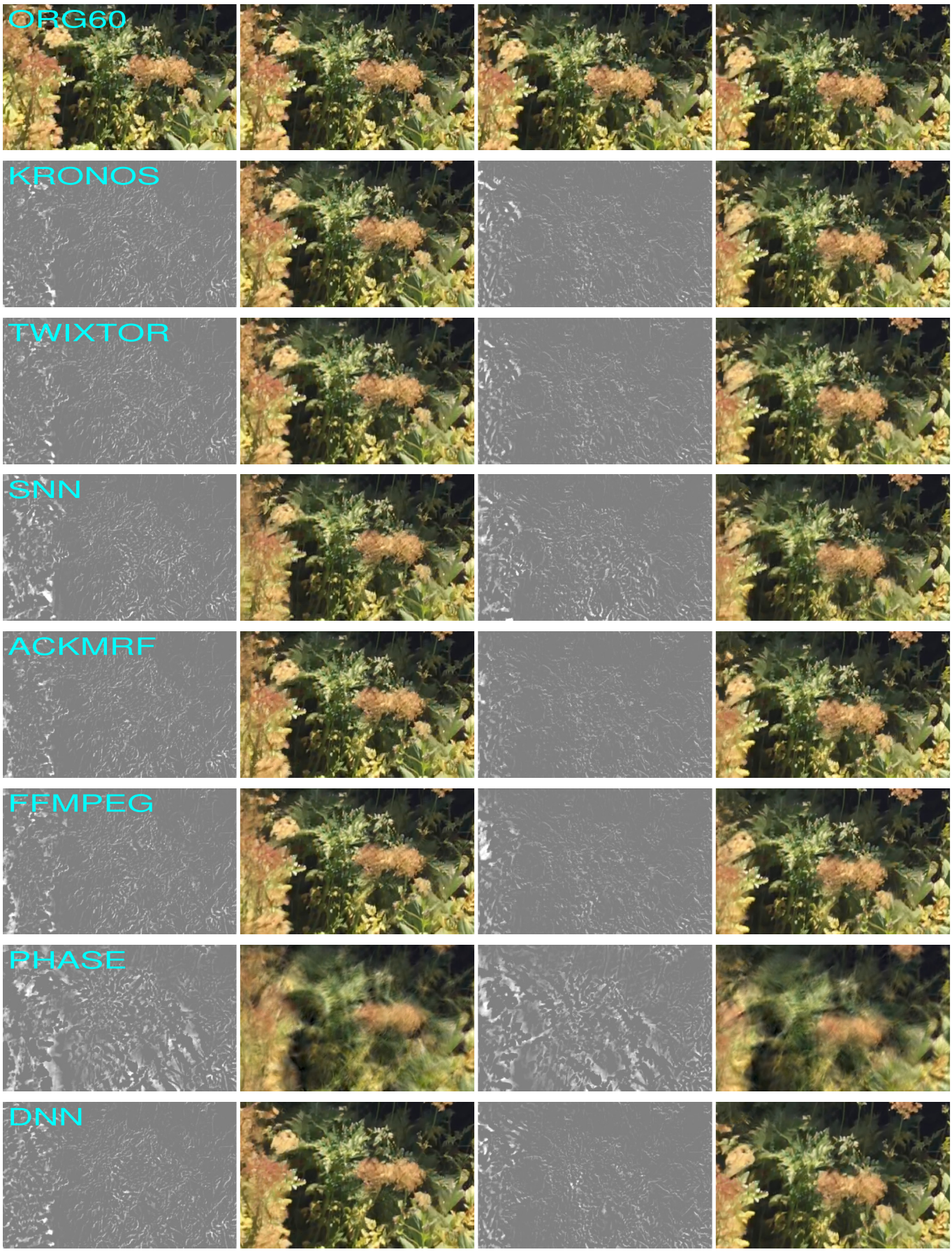

Fig. 8: *Performance of all methods on one sequence with high texture and large camera motion (IMG0188 in the Adobe Dataset). Top row : Original 60fps sequence, frames 17-20. The other rows show the performance of each method in reconstructing frames 18, 20. Left to right in each row : Difference between interpolated frame 18 and actual frame 18, Interpolated frame 18, Difference between interpolated frame 20 and actual frame 20, Interpolated frame 20. Note that both SNN and DNN struggle with the large amount of picture material disappearing off the left hand edge of the frame. Video illustrates tihs better than these still. This issue is worse for all methods in frame 20 than in frame 18. However the motion based techniques still cope better overall.*

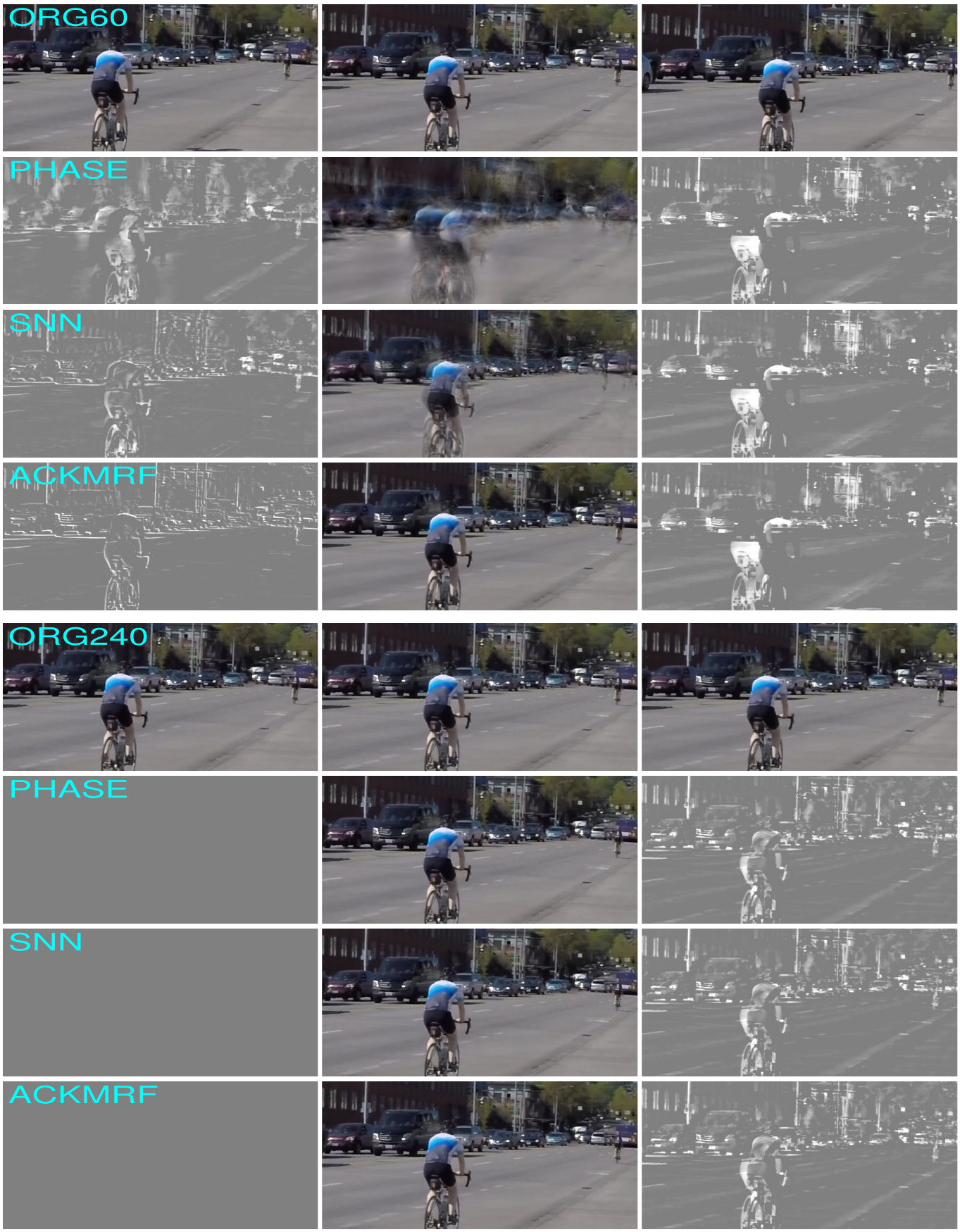

Fig. 9: *Performance improves as temporal aliasing reduces. Top 4 rows :* $30 \rightarrow 60$ *fps. Bottom :* $120 \rightarrow 240$ *fps. Top row shows frame 77,78,79 from GOPRO9637b in the Adobe dataset at 60fps. Subsequent rows show error between interpolated frame and the actual , the interpolated frame 78 (upsampled using 77,79) and the non-motion compensated frame difference between frames 79 and 78. The second set of 4 rows shows the same situation but at 240fps, frames 308, 309, 310. The time instant at frame 309 at 240fps corresponds to frame 78 at 30fps. At 30fps the motion based techniques (ACKMRF is only used here as an example) perform much better than PHASE or SNN in this example.*

Fig. 10: *All methods struggle with complicated occlusion. Top row shows the original sequence at 60fps (IMG014 from the Adobe dataset). Subsequent rows show the upsampled sequence* $30 \rightarrow 60$ *fps using various methods. The 1st and 3rd panes in each of these rows show the frame difference between the reconstructed frames in the 2nd and 4th panes with the original sequence. As the car passes behind the orange bollard, the top part is distorted in all the reconstructions except the PHASE method. The PHASE method performs well here because the bollard is stationary.*

Fig. 11: *One sequence where SNN outperforms the other methods. Top row shows the original sequence at 60fps (frames 193-195 of IMG018 from Adobe dataset). Subsequent rows show the reconstructed middle frame* $30 \rightarrow 60$ *fps using various methods. The 1st and 3rd panes in each of these rows show the frame difference between the reconstructed frames with the original sequence and the frame difference between the next and previous frame respectively. That last pane shows how much motion there is between the two frames used for interpolation and it is the same difference for each row. The differences in the first pane of each row show how well the SNN method deals with the bicycle wheel at the back of the vehicle while all the other methods struggle with this thin dark object.*