Phase-Based Frame Interpolation for Video

Simone Meyer[1] Oliver Wang[2] Henning Zimmer[2] Max Grosse[2] Alexander Sorkine-Hornung[2]

[1]ETH Zurich [2]Disney Research Zurich

Abstract

Standard approaches to computing interpolated (in-between) frames in a video sequence require accurate pixel correspondences between images e.g. using optical flow. We present an efficient alternative by leveraging recent developments in phase-based methods that represent motion in the phase shift of individual pixels. This concept allows in-between images to be generated by simple per-pixel phase modification, without *the need for any form of explicit correspondence estimation. Up until now, such methods have been limited in the range of motion that can be interpolated, which fundamentally restricts their usefulness. In order to reduce these limitations, we introduce a novel, bounded phase shift correction method that combines phase information across the levels of a multi-scale pyramid. Additionally, we propose extensions for phase-based image synthesis that yield smoother transitions between the interpolated images. Our approach avoids expensive global optimization typical of optical flow methods, and is both simple to implement and easy to parallelize. This allows us to interpolate frames at a fraction of the computational cost of traditional optical flow-based solutions, while achieving similar quality and in some cases even superior results. Our method fails gracefully in difficult interpolation settings, e.g., significant appearance changes, where flow-based methods often introduce serious visual artifacts. Due to its efficiency, our method is especially well suited for frame interpolation and retiming of high resolution, high frame rate video.*

1. Introduction

Computing interpolated, in-between images is a classic problem in image and video processing, and is a necessary step in numerous applications such as frame rate conversion (e.g. between broadcast standards), temporal upsampling for generating slow motion video, image morphing, as well as virtual view synthesis. Traditional solutions to image interpolation first compute correspondences (mostly leveraging optical flow or stereo methods), followed by correspondence-based image warping. Due to inherent ambiguities in computing such correspondences, most methods are heavily dependent on computationally expensive global optimization and require considerable parameter tuning [24]. With today's trend in the movie and broadcasting industry to higher resolution, higher frame rate video (e.g. current cameras support 4k resolution at 120 frames per second and beyond), there is the need for interpolation techniques that can deal efficiently with this considerably larger data volume. Standard optical flow techniques based on global optimization often become inefficient for interpolating this type of large scale, densely sampled input.

Recently, phase-based methods have shown promising results in applications such as motion and view extrapolation [10, 27]. These methods rely on the assumption that small motions can be encoded in the phase shift of an individual pixel's color. Currently, however, the spatial displacement which can be encoded in the phase information with these methods is highly limited, which narrows their practical usability.

Contributions. To overcome this fundamental issue, we propose a method that propagates phase information across oriented multi-scale pyramid levels using a novel bounded shift correction strategy. Our algorithm estimates and adjusts the phase shift information using a coarse-to-fine approach, assuming that high frequency content moves in a similar way to lower frequency content. We propose an adaptive upper bound on the phase shift that effectively avoids artifacts for large motions, and an extension to phase-based image synthesis that leads to smoother transitions between interpolated images. In combination, these extensions considerably increase the amount of displacement that can be successfully represented and interpolated, rendering phase-based methods practical to be used for general motion interpolation tasks, e.g., for high resolution, high frame rate video.

Based on these extensions, we describe an efficient framework to synthesize in-between images, which is simple to implement and parallelize. Our implementation allows us to interpolate a HD frame in a few seconds on the CPU and in 1 second on a GPU, while scaling favorably for larger image sizes. Thus our method only requires a

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

fraction of the time compared to typical flow-based methods, while having comparable memory requirements and obtaining visually similar results. In the presence of severe brightness variations, our approach can even give better results than optical flow. Furthermore, our method only has a small set of parameters, most of which were fixed for all experiments. Evaluations on ground truth data against state-of-the art techniques using optical flow show similar quality and favorable performance in terms of efficiency.

2. Related Work

Image interpolating techniques can roughly be classified as either Lagrangian or Eulerian. While Lagrangian methods model motion as spatial pixel displacement, Eulerian methods consider the change of color per pixel over time.

Lagrangian methods. The most popular methods for finding pixel correspondences across images are based on optical flow; see [4, 24] for extensive reviews. Seminal works include the first global variational formation by Horn and Schunck [14] and the local alternative by Lucas and Kanade [16]. We compare to modern global methods that perform well when used for interpolation, including the method of Brox et al. [6], and the currently best-ranked method for interpolation on the Middlebury benchmark [29]. As most optical flow methods rely on global optimization to resolve matching ambiguities they are difficult to implement and parallelize, and slow in convergence. There has been some parallel flow implementations on GPUs which allow to significantly reduce running times [28, 30]. However, they still do not scale well w.r.t. image size and are restricted by GPU memory, limiting their applicability for high resolution data. Tao et al. [26] proposed a flow method that replaces global optimization by edge-aware filtering. They achieve a favorable scaling in practice, but at the cost of sacrificing accuracy. Flow methods are in general sensitive to parameter settings and tend to produce visual artifacts for strong brightness changes. As an alternative to optical flow, Mahajan et al. [17] compute paths in the source images and copy pixel gradients along them to the in-between images, that are then obtained by Poisson reconstruction. While alleviating some problems of optical flow, this method still requires expensive global optimization and careful parameter tuning. In contrast to above approaches, our method is simple and local, allowing for an easy and efficient parallelization. It scales well w.r.t. image size, does not require careful parameter tuning, and fails gracefully in challenging cases.

Some earlier work [11, 12] used phase information directly in optical flow computations by replacing image brightness with phase in the data term of standard optical flow formulations. The main benefit is that the phase is more robust than the amplitude w.r.t smooth lighting variations. More recent methods include reliability measurements and GPU implementations [13, 19]. These methods use phase information only to match pixels in a standard optical flow formulation, whereas our method interpolates the phase information directly.

Eulerian methods. There has been a recent interest in using Eulerian methods for enhancing subtle motions in a simple and efficient way [9, 10, 27]. We aim at extend these ideas in order to use them for frame interpolation, which mainly requires to model larger amounts of motion.

Arguably the simplest Eulerian method for interpolating images is weighted averaging. As no motion is interpolated, ghosting of high frequency content is unavoidable. To hide ghosting artifacts, multiband blending [7] or gradient domain blending [20] can be used. Further improvements can be obtained by matching the visual appearance of the images before blending, e.g. by manipulating the scales of a pyramid decomposition [25]. However, all these methods are designed to create composites of images where the blending is done along seams or within a mask. Thus non-trivial modifications would be needed to use them to interpolate the motion between two images. Image melding [8] allows blending between images using patch based similarity, but this approach is not guaranteed to yield smooth and temporally consistent results.

Most related to our work is the method of Didyk et al. [10] that proposes a phase-based approach to extrapolate a pair of stereo images to multi-view content for automultiscopic 3D displays. As we show in our comparisons, this method is restricted to very small displacements, and cannot directly be applied for more general frame interpolation for video. To deal with larger displacements as well as fractional interpolation of views, we present several improvements including a novel, bounded shift correction algorithm and modifications required for a smooth interpolation between frames.

3. Method

3.1. Phase-based Motion Interpolation

Phase-based approaches build on the insight that the motion of certain signals can be represented as phase-shift. We first explain the basic concepts before introducing our generalization and modifications for image interpolation.

1D Case. Consider a one dimensional sinusoidal function shown in Figure 1 which is defined as $y = A\sin(\omega x - \phi)$, where A is the amplitude, ω the angular frequency and ϕ the phase. A translation of this function can be described by modifying the phase, e.g. by subtracting $\pi/4$ in our example. The phase shift ϕ_{shift}, which corresponds to the actual spatial displacement between two translated functions, is defined as the phase difference ϕ_{diff} between the two

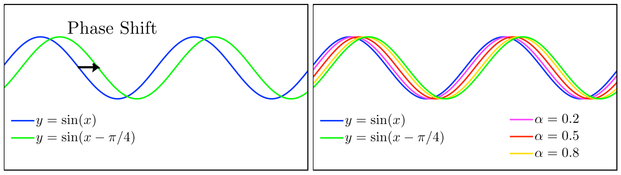


Figure 1: The translation of two sinusoidal functions (left) can be interpolated according to Equation (2) (right).

phases of the curves scaled by ω:

$$\phi_{shift} = \frac{\phi_{diff}}{\omega} \ . \tag{1}$$

Let us now modify the phase difference according to a weight $\alpha \in (0,1)$ that describes an intermediate position between the functions:

$$y = A \sin(\omega x - \alpha\, \phi_{diff}) = A \sin(\omega(x - \alpha\, \phi_{shift})) \ . \tag{2}$$

The resulting functions then correspond to intermediate sinusoids representing the translational motion, see Figure 1.

This idea can be extended to general functions $f(x)$ translated by a displacement function $\delta(x)$ [27]. The shifted function $f(x+\delta(t))$ can be represented in the Fourier domain as a sum of complex sinusoids over all frequencies ω:

$$f(x + \delta(t)) = \sum_{\omega=-\infty}^{\omega=-\infty} R_\omega(x,t) \ , \tag{3}$$

where each sinusoid represents one band $R_\omega(x,t) = A_\omega\, e^{i\omega(x+\delta(t))}$. The corresponding phase $\phi_\omega = \omega(x+\delta(t))$ can be directly modified w.r.t. α, leading to modified bands

$$\hat{R}_\omega(x,t) = A_\omega\, e^{i\omega(x+\alpha\delta(t))} \ . \tag{4}$$

The in-between functions are then obtained by integrating the modified bands in accordance to (3).

2D Generalization. For two dimensional functions one can separate the sinusoids into bands not only according to the frequency ω, but also according to spatial orientation θ, using e.g. the complex-valued steerable pyramid [21, 22, 23]. The steerable pyramid filters resemble Gabor wavelets and, when applied to the discrete Fourier transform of an image, they decompose the input images into a number of oriented frequency bands $R_{\omega,\theta}$. The remaining frequency content which has not been captured in the pyramid levels is summarized in (real valued) high- and low-pass residuals. An example for such a decomposition can be found in [27].

Phase computation. The complex-valued response $R_{\omega,\theta}$ obtained by applying the steerable filters $\Psi_{\omega,\theta}$ [21] to an image I can be written as:

$$\begin{aligned} R_{\omega,\theta}(x,y) &= (I * \Psi_{\omega,\theta})(x,y) && (5)\\ &= A_{\omega,\theta}(x,y)\, e^{i\phi_{\omega,\theta}(x,y)} && (6)\\ &= C_{\omega,\theta}(x,y) + i\, S_{\omega,\theta}(x,y), && (7) \end{aligned}$$

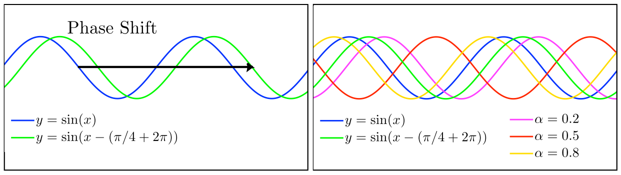


Figure 2: Interpolation of a large phase shift, with the same initial phase difference as in Figure 1. Adding (in this case) 2π to the phase difference enables correct interpolation of larger motion. The challenge is how to reliably estimate such large shifts.

where $C_{\omega,\theta}$ is the cosine part, representing the even-symmetric filter response, and $S_{\omega,\theta}$ is the sine part, representing the odd-symmetric filter response. From this we can compute the amplitude $A_{\omega,\theta}(x,y) = \sqrt{C_{\omega,\theta}(x,y)^2 + S_{\omega,\theta}(x,y)^2}$, and the phase components $\phi_{\omega,\theta}(x,y) = \arctan(S_{\omega,\theta}(x,y)/C_{\omega,\theta}(x,y))$.

Phase difference. Based on the assumption that small motion is encoded in the phase shift, interpolating it requires the computation of the phase difference ϕ_{diff} (see Equation (1)) between the phases of the two input frames as

$$\phi_{diff} = \operatorname{atan2}(\sin(\phi_1 - \phi_2), \cos(\phi_1 - \phi_2)) \ , \tag{8}$$

where atan2 is the four-quadrant inverse tangent. This approach results in angular values between $[-\pi, \pi]$, which correspond to the smaller angular difference between the two input phases. It additionally determines the limit of motion that can be represented, which is bounded by:

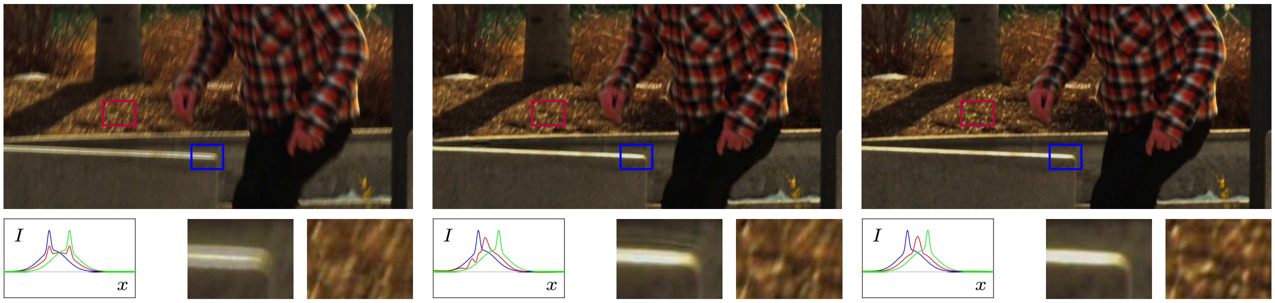
$$|\phi_{shift}| = \frac{|\phi_{diff}|}{\omega} \leq \frac{\pi}{\omega} \ , \tag{9}$$

where $\omega = 2\pi\nu$, and ν being the spatial frequency.

In the multi-scale pyramid, each level represents a particular band of spatial frequencies $\nu \in [\nu_{min}, \nu_{max}]$. Assuming ν_{max} corresponds to the highest representable frequency on that level, then a phase difference of π corresponds exactly to a shift of one pixel. While this is a reasonable shift for low frequency content represented on the coarser pyramid levels, it is too limiting for high frequency content to achieve realistic interpolation results in the presence of larger motions. Our extensions described below overcome this limitation.

3.2. Bounded Shift Correction

Large displacements corresponding to a phase difference of more than π lead to a phase ambiguity. Due to the periodicity of the phase value, the phase difference is only defined between $[-\pi, \pi]$ and corresponds to the smallest angular difference. An example is shown in Figure 2, where the



(a) Linear blending of the input images (b) Shift correction based on absolute values [10] (c) Our shift correction based on phase differences

Figure 3: Shift correction comparison. Simple linear blending (a) results in obvious ghosting. The approach of [10] (b) interpolates, but leads to undesirable oscillations which manifest as ringing and blurring artifacts. Our approach (c) produces a plausible interpolation and resolves these problems. (© Tom Guilmette [1]).

actual spatial displacement is larger than the original computed phase by a multiple of $\pm 2\pi$. Although the input functions look identical to Figure 1, the intermediate sinusoids differ depending on the assumed phase shift. Interpolation only works as long as the shift computed on a particular level mainly captures the frequency content corresponding to the true motion.

Didyk et al. [10] propose to handle this problem by setting the phase difference at pyramid level l to two times its absolute value at the next coarser level $l+1$ whenever the phase difference becomes greater than $\pi/2$. Formally, if $|\phi_{diff}^{l+1}| > \pi/2$, the corrected phase difference is given by $\tilde{\phi}_{diff}^{\,l} = 2\,\phi_{diff}^{l+1}$. This essentially defines a level where the phase difference of a pixel is assumed to correctly estimate the motion, and then this value is simply copied to the remaining finer resolution levels, ignoring their own respective phase values. This leads to a loss of detailed motion for high frequency content, resulting in artifacts such as ringing and blurring of detail (see Figure 3b). Additionally, this approach is limited to pyramids constructed using a scaling factor of two, which is known to be suboptimal for the robustness of image-based multi-scale methods [6].

Confidence based shifting. To overcome the limitations of the above approach, we developed a new confidence estimate for the shift correction which takes all available shift information into account and interpolates the motion of high frequency content more robustly. Our approach is based on the assumption that the phase difference between two resolution levels does not differ arbitrarily, i.e. phase differences between levels can be used as a confidence measure that quantifies whether the computed phase shift is reliable.

More specifically, we try to first resolve the 2π ambiguity (Figure 2) based on the information of the next coarser level. Only if the computed shift on the current level l differs more than a threshold from the coarser level $l+1$ we perform shift correction on level l. To this end, we first add multiples of $\pm 2\pi$ to ϕ_{diff} s.t. the absolute differences between the phase values of consecutive levels are never greater than a given tolerance. We use π as a tolerance distance which modifies the phase values in such a way that the phase difference of a pixel between two levels is never larger than π. Because the original phase differences are truncated to the range $[-\pi, \pi]$, this step allows the extension of the range in a meaningful way.

The actual shift correction depends on the difference between two levels, which serves as our confidence estimate

$$\varphi = \operatorname{atan2}\Big(\sin(\phi_{diff}^{l} - \lambda\,\phi_{diff}^{l+1}), \cos(\phi_{diff}^{l} - \lambda\,\phi_{diff}^{l+1})\Big)\,, \tag{10}$$

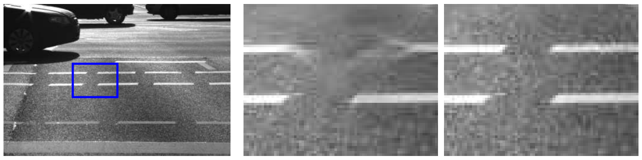
where the phase value of the coarser level is scaled according to an arbitrary pyramid scale factor $\lambda > 1$ to get a scale-independent estimate. If $|\varphi| > \pi/2$, we apply shift correction and obtain the corrected phase difference as $\tilde{\phi}_{diff}^{\,l} = \lambda\,\phi_{diff}^{l+1}$. As shown in Figure 3c and in the supplementary video [2], this approach produces considerably higher quality interpolation results compared to the simpler correction based on absolute values [10].

Bounded phase shift. While our shift correction allows larger motions to be modeled, there is still a limit to the motion that can be represented without introducing blurring artifacts. We therefore propose an additional enhancement, which limits the admissible phase shifts to well representable motions.

To this end we limit the phase difference by a constant ϕ_{limit}. If the phase value is above this limit, i.e. $|\phi_{diff}^{l}| > \phi_{limit}$, the phase value from the next coarser level is used as the corrected phase difference: $\tilde{\phi}_{diff}^{\,l} = \lambda\,\phi_{diff}^{l+1}$.

We define ϕ_{limit} depending on the current level l, the total number of levels L, and the scale factor λ as

$$\phi_{limit} = \tau\,\pi\,\lambda^{L-l}\,, \tag{11}$$



(a) Input image [18] (b) Unbounded (c) Bounded ($\tau = .1$)

Figure 4: Bounding in the shift correction allows to retain sharpness when interpolating large motion.

where the parameter $\tau \in (0, 1)$ determines the percentage of limitation. On the coarsest level we set the corrected phase difference to zero if its magnitude exceeds ϕ_{limit}.

The effect of this limiting is shown in Figure 4 where blurring that extends beyond the moving car is avoided as can be seen in the retained sharpness of the white stripe.

As an additional advantage over prior work [10], our formulation is generalized to arbitrary pyramid scale factors λ. Similar to findings for optical flow based techniques [6], we found this to be important for high quality results as with more pyramid levels it becomes possible to represent sharper transitions, meaning that the phase differences for higher frequencies are estimated more reliably and accurately. Decreasing the scale factor naturally comes at the cost of higher memory consumption and increased computation time. We found a good balance of result quality and computation time, which we discuss in Section 4.

3.3. Phase Interpolation

For the rest of this section we omit the superscript l denoting the pyramid level to improve readability.

Matching phase differences. We now explain how to compute a smooth interpolation between phases ϕ_1 and ϕ_2. Due to the shift correction, $\phi_1 + \tilde{\phi}_{diff}$ is no longer guaranteed to match ϕ_2, or any equivalent multiple $\phi_2 \pm \gamma 2\pi$, where $\gamma \in \mathbb{N}_0$. In order for the resulting images to be smoothly interpolated, we must preserve the original computed phases ϕ_1 and ϕ_2, up to the 2π ambiguity, while still respecting the shift corrected phase difference $\tilde{\phi}_{diff}$. We do this by searching for a phase difference $\hat{\phi}_{diff}$ that is $\pm\gamma 2\pi$ the original phase difference ϕ_{diff} from Equation (8) while being as close as possible to $\tilde{\phi}_{diff}$, i.e.,

$$\hat{\phi}_{diff} = \phi_{diff} + \gamma^* \, 2\pi \, , \tag{12}$$

where γ^* is determined as

$$\gamma^* = \underset{\gamma}{\operatorname{argmin}} \left\{ \left(\tilde{\phi}_{diff} - (\phi_{diff} + \gamma \, 2\pi) \right)^2 \right\} \, . \tag{13}$$

Due to the adjustment we can now compute the phase ϕ_α of the interpolated images based on the phase of one input frame and a fraction of the final phase differences $\hat{\phi}_{diff}$ as

$$\phi_\alpha = \phi_1 + \alpha \, \hat{\phi}_{diff} \, . \tag{14}$$

Subtracting $\alpha \, \hat{\phi}_{diff}$ from ϕ_2 would give the same result.

Blending amplitudes. To reconstruct the interpolated images we not only need to interpolate the phase, but also the low-pass residual and the amplitude. For the amplitude we found that using the approach of Didyk et al. [10], which extrapolates using the amplitude value of the closer input frame, leads to undesirable popping artifacts for interpolation. To obtain a smoother transition, we propose to linearly blend the amplitude as well as the low frequency residual. On coarse resolutions these quantities mainly correspond to the global luminance difference between the input frames, hence linear blending does not create any visible artifacts.

In combination, both extensions proposed in this section allow for smooth interpolation between the two images, as demonstrated in the supplementary video [2]. Additionally we add back the high-pass residual of the closer input frame to retain as much high frequency content as possible.

3.4. Algorithm Summary

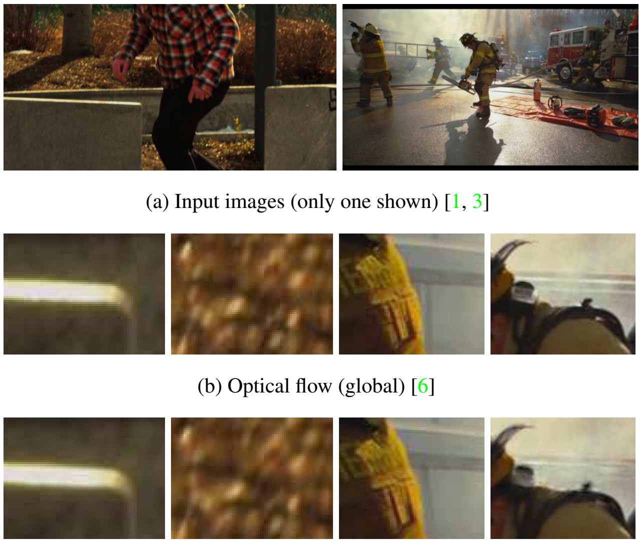
Algorithm 1 provides a summary of all the key steps of our method. The handling of the residuals as described above is omitted for reasons of brevity.

Algorithm 1 The inputs are two images I_1 and I_2 and interpolation parameter α. The output is the interpolated image I_α. P_1 and P_2 are the steerable pyramid decompositions.

```
$(P_1, P_2) \leftarrow$ decompose$(I_1, I_2)$                          ▷ See [21]
$(\phi_1, \phi_2) \leftarrow$ phase$(P_1, P_2)$
$(A_1, A_2) \leftarrow$ amplitude$(I_1, I_2)$

$\phi_{diff} \leftarrow$ phaseDifference$(\phi_1, \phi_2)$                 ▷ See Eq. 8
for all $l = L - 1 : 1$ do
    $\tilde{\phi}^l_{diff} \leftarrow$ shiftCorrection$(\tilde{\phi}^{l+1}_{diff})$        ▷ See Sec. 3.2
end for
$\hat{\phi}_{diff} \leftarrow$ adjustPhase$(\phi_{diff}, \tilde{\phi}_{diff})$             ▷ See Eq. 12

$\phi_\alpha \leftarrow$ interpolate$(\phi_1, \hat{\phi}_{diff}, \alpha)$                 ▷ See Eq. 14
$A_\alpha \leftarrow$ blend$(A_1, A_2, \alpha)$                        ▷ See Sec. 3.3

$P_\alpha \leftarrow$ recombine$(\phi_\alpha, A_\alpha)$                          ▷ See [21]
$I_\alpha \leftarrow$ reconstruct$(P_\alpha)$
```

4. Results

We evaluate our approach using images from the Middlebury dataset [4], from the challenging dataset for stereo and optical flow [18], as well as ground truth data obtained with high-speed cameras [1, 3]. We compare our



(a) Input images (only one shown) [1, 3]

(b) Optical flow (global) [6]

(c) Our phase-based method

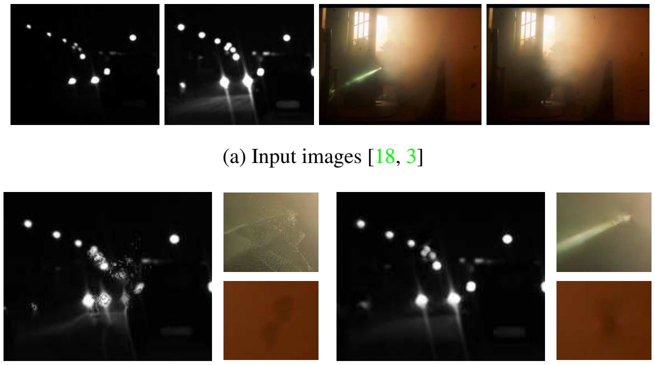
Figure 5: Interpolated frames using optical flow (b) and our approach (c). Our approach produces similar quality results while being faster and simpler. (© Tom Guilmette [1], © Vision Research [3])

results to interpolation methods using state-of-the art optical flow, including MDP-Flow [29] which currently ranks first on the Middlebury interpolation benchmark, as well the method of Brox et al. [6] and a GPU flow implementation (flowlib) [28]. We also include comparisons to the SimpleFlow method [26] that avoids global optimization and to the local optical flow method of Lucas-Kanade [16] using a multi-scale pyramid and iterative refinement [5] implemented in Matlab (and therefore removed from the timings). For synthesizing the in-between images given the computed flow fields, we apply the interpolation algorithm used in the Middlebury interpolation benchmark [4]. We also compare to Eulerian methods, including simple linear blending, and the work of Didyk et al. [10].

Qualitative comparisons. Figure 5 visually compares our method to a representative flow-based method and shows that we obtain visually similar results, which becomes even more obvious in the supplementary video [2].

Besides being computationally more expensive, optical flow-based approaches also fail when brightness constancy is violated. In these cases, our approach reduces to simple blending, which is preferable to the artifacts caused by optical flow. See Figure 6 and the supplementary video [2] for examples with strong lighting changes.

The robustness of our phase-based method in dealing with brightness changes can also be used to interpolate scenes with both motion and color change, as color changes are often encoded in the lower frequencies, which our phase-based method handles well. An example is shown in Figure 7 around the border of the sun.

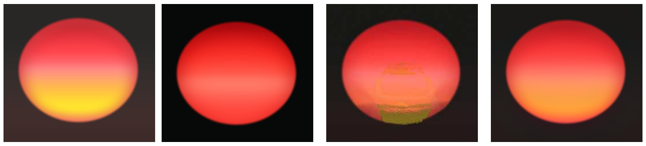


(a) Input images [18, 3]

(b) Optical flow (global) [6]

(c) Our phase-based method

Figure 6: Some failure cases for optical flow due to strong lighting changes that are gracefully handled by our method. (© Vision Research [3])



(a) Input images

(b) Flow [6]

(c) Ours

Figure 7: Interpolating between an images featuring both scene motion and color change.

Ground truth comparisons. We conducted a number of ground truth comparisons using the leave-some-out method, i.e. we synthesize intermediate frames and compare these to the original ones. We use one synthetic example (*Roto*) that demonstrates typical artifacts. All other sequences are extracted from real world footage taken from the Middlebury database [4], or from high speed videos recorded at 120+fps [3]. We focus on challenging scenes with many moving parts as well as changing lighting conditions.

The plot in Figure 8 (left) visualizes the decay in quality when leaving out an increasing number of intermediate images (causing larger displacements). We computed error measurements for different Lagrangian, i.e. flow-based methods, as well as Eulerian methods. As error measurements we use the sum of squared distances (SSD). The supplementary material also reports results for the perceptually motivated structural similarity (SSIM) measure. The plot in Figure 8 (right) visualizes the SSD averaged over several frames for various sequences when skipping a single frame.

Both plots show that our phase-based method performs better than other Eulerian approaches, and is on par with optical flow methods. This observation is also confirmed by the SSIM error measures shown in the supplementary ma-

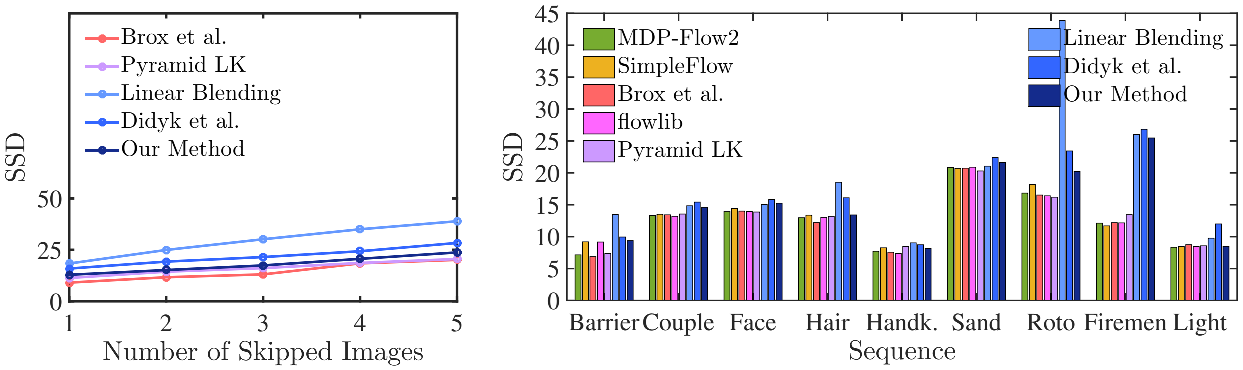


Figure 8: Error measurements (SSD) for different methods. On the RubberWhale sequence [4] showing quality degradation with increased motion between frames (left). Averaged over several frames of different sequences, skipping one frame (right). Example images from these datasets are shown in the supplementary material.

Method	Processing	Synthesizing	Total Time
MDP-Flow2 [29]	952.1	0.7	952.8
SimpleFlow [26]	*70.2+3.1	0.7	74.0
Brox et al. [6]	25.2	0.7	25.9
Our method (CPU)	3.3	3.3	6.6
flowlib (GPU) [28]	3.1	0.7	3.8
Our method (GPU)	0.9	0.3	1.2

Table 1: Runtimes (in seconds) for different methods measured when interpolating a 1280x720 (HD) frame. *For SimpleFlow we report the sum of the preprocessing time (done in Matlab) and the final flow computation.

terial. In general, optical flow-based methods give slightly better error measurements, largely because they introduce less blur, see Figure 10 for examples and the end of this section for a detailed discussion of the limitations. In many cases however, the numerical differences do not correspond to differences in visual quality, which is our main goal, and which is best visible in the supplementary video [2].

Concerning parameters, for Pyramid LK we optimized the smoothness parameter defined by the window size and for the method of Brox et al. [6] we used the parameters proposed by the author of the implementation [15]. Our method has more intuitive parameters that can mostly be fixed. We discuss parameters in more detail below.

Running times. In Table 1 we report running times for interpolating between a pair of 1280x720 (HD) images. We use such a rather small resolution to be able to run all competing methods, as the highly accurate MDP-Flow2 method [29] runs into memory limitations on larger images. All measurements were performed on a standard desktop PC (Intel Core i7 3.4GHz, 32GB memory). The GPU measurements were run on a GeForce GTX 770 with 2GB memory. We report running times for processing the images and synthesizing one output image. For the phase-based method, processing includes the pyramid decomposition as well as our bounded shift correction. Synthesizing an intermediate image includes the smooth interpolation computations as well as the collapsing of the pyramid. For optical flow these two steps correspond to computing the optical flow field and the warping procedure, respectively.

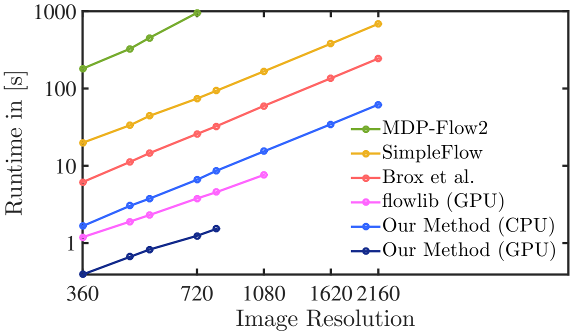
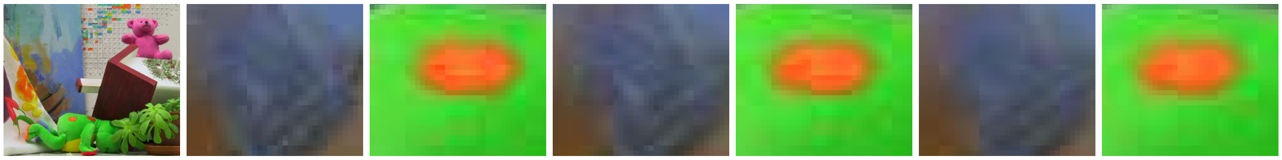


Figure 9: Log-log plot of running time versus (vertical) image resolution when interpolating one image. Our method scales favorably compared to flow-based methods.

For the flow-based methods we use available code; for Brox et al. [6] we use the C++ implementation of [15], whereas for MDP-Flow2 [29], SimpleFlow [26] and flowflib [28] we use the C / CUDA code provided by the authors. The image warping using the flow fields to interpolate the intermediate images is implemented in C++. Our method is implemented and parallelized in C++ on the CPU and using CUDA on the GPU.

Comparing the times in Table 1 clearly shows that our method is significantly and consistently faster than any of the flow-based methods, both for CPU as well as GPU implementations. Additionally, the log-log plot in Figure 9



(a) Input Images [4]
(b) Optical flow [29]
(c) Our phase-based method

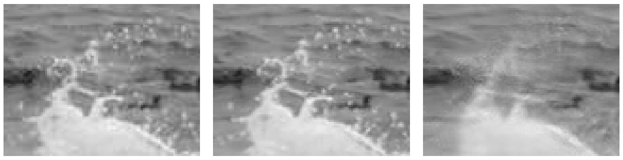
Figure 10: Introduced blur of different methods. (a) shows the input images, compared to interpolated images generated by optical flow (b), and our method (c), which are slightly more blurry.

shows that our method scales favorably w.r.t. the number of pixels compared to flow-based methods, which is an important criterion for processing high resolution data.

Implementation details. Our phase-based approach has very few, and intuitive parameters. The main parameters control the number of orientations and levels corresponding to the different scales of the pyramid. More levels and orientations allow for a better separation of the motion at the expense of increased memory and computation time. Unless otherwise specified we used the following parameters: The pyramid was constructed using $\#_\theta = 8$ number of orientations, a scale factor $\lambda = 1.2$, and the number of levels L is determined s.t. the coarsest level has a minimal width of 10 pixels. For the limitation factor we use $\tau = 0.2$. The size of the coarsest level together with this choice of τ leads to a theoretical limit of motion which can be modeled reliably as 2% of the image width. We applied the pyramid decomposition and the phase modifications to each color channel in Lab color space independently.

Discussion and limitations. While our method provides an efficient alternative to optical flow, it has some limitations. Although we retain more sharpness compared to [10] (see e.g. Figure 3), our approach still incurs some blurring even in areas with small motion. This is due to the fact that the high-pass residual contains the highest frequencies whose motion is not represented in any pyramid levels and consequently the motion of this frequency can currently not be interpolated. In general we thus cannot reach the same level of detail as Lagrangian approaches that explicitly match and warp pixels, see Figure 10. Note however that the introduced blur is often very subtle. Reintroducing or preserving high frequencies in phase-based methods is still an interesting and promising area for future work.

Similar observations apply to large motions of high frequency content that cannot be represented by our phase estimation. Here our algorithm degenerates to linear blending, which however often leads to less objectionable artifacts than the entirely wrong motion estimates produced by optical flow in similar situations. An example is shown in Figure 11 where interpolating an image between neighboring frames, i.e., with small motion, the detail of the water remains, whereas when interpolating between images twenty frames apart, the motion of the water is too large and it becomes blurred. On the other hand, the current trend towards higher frame rate video will surely continue in the foreseeable future, and thus the apparent motion between frames will become smaller, and efficiency more important.



(a) Original Frame
(b) Small motion
(c) Large motion

Figure 11: When interpolating between frames with small displacements (b), our approach can reconstruct high frequency details. For a separation of twenty frames (c), the larger motion causes blurring. (© Vision Research [3])

5. Conclusions

We have presented a novel method for frame interpolation using a phase-based technique. We compared our results with state-of-the-art optical flow-based approaches and found similar visual quality over a number of real world datasets. For strong illumination changes we even observed a better performance. Important advantages of our method are fewer, more intuitive parameters that can mostly be fixed, as well as a graceful degradation for challenging scenarios. Most importantly, our method is computationally very efficient and simple to implement and parallelize. All these make our method a practical tool for frame interpolation on high resolution, high frame rate video, which is generally challenging for optical flow based techniques.

We hope that this new perspective on a long standing problem has many interesting areas for future work. Making phase computation more robust using spatial similarity would be an interesting direction. Additionally, one may combine phase-based with traditional Lagrangian approaches to generate higher quality results and handle illumination changes more gracefully.

6. Acknowledgements

We would like to thank Tom Guilmette [1] and Vision Research [3] for providing test footage.

References

[1] http://www.tomguilmette.com/archives/593. 4, 5, 6, 9

[2] http://www.disneyresearch.com/project/phasebased. 4, 5, 6, 7

[3] http://www.visionresearch.com/Gallery. 5, 6, 8, 9

[4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011. 2, 5, 6, 7, 8

[5] J.-Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. *Intel Corporation*, 2000. 6

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36, 2004. 2, 4, 5, 6, 7

[7] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2(4):217–236, 1983. 2

[8] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82, 2012. 2

[9] A. Davis, M. Rubinstein, N. Wadhwa, G. J. Mysore, F. Durand, and W. T. Freeman. The visual microphone: passive recovery of sound from video. *ACM Trans. Graph.*, 33(4):79, 2014. 2

[10] P. Didyk, P. Sitthi-amorn, W. T. Freeman, F. Durand, and W. Matusik. Joint view expansion and filtering for automultiscopic 3D displays. *ACM Trans. Graph.*, 32(6):221, 2013. 1, 2, 4, 5, 6, 8

[11] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *IJCV*, 5(1):77–104, 1990. 2

[12] D. J. Fleet and A. D. Jepson. Stability of phase information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(12):1253–1268, 1993. 2

[13] T. Gautama and M. M. V. Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5):1127–1136, 2002. 2

[14] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artif. Intell.*, 17(1-3):185–203, 1981. 2

[15] C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, 2009. 7

[16] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 2, 6

[17] D. Mahajan, F. Huang, W. Matusik, R. Ramamoorthi, and P. N. Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Trans. Graph.*, 28(3), 2009. 2

[18] S. Meister, B. Jähne, and D. Kondermann. Outdoor stereo camera system for the generation of real-world benchmark data sets. *Optical Engineering*, 51(02):021107, 2012. 5, 6

[19] K. Pauwels and M. Van Hulle. Realtime phase-based optical flow on the GPU. In *CVPR Workshops*, June 2008. 2

[20] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003. 2

[21] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1):49–70, 2000. 3, 5

[22] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing*, pages 444–447. IEEE, 1995. 3

[23] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992. 3

[24] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439. IEEE, 2010. 1, 2

[25] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister. Multi-scale image harmonization. *ACM Trans. Graph.*, 29(4), 2010. 2

[26] M. W. Tao, J. Bai, P. Kohli, and S. Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. *Comput. Graph. Forum*, 31(2):345–353, 2012. 2, 6, 7

[27] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman. Phase-based video motion processing. *ACM Trans. Graph.*, 32(4):80, 2013. 1, 2, 3

[28] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, pages 1–11, 2009. 2, 6, 7

[29] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1744–1757, 2012. 2, 6, 7, 8

[30] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L^1 optical flow. In *DAGM*, pages 214–223, 2007. 2