PhaseNet for Video Frame Interpolation

Simone Meyer[1,2]   Abdelaziz Djelouah[2]   Brian McWilliams[2]   Alexander Sorkine-Hornung[2*]
Markus Gross[1,2]   Christopher Schroers[2]

[1]Department of Computer Science, ETH Zurich   [2]Disney Research

simone.meyer@inf.ethz.ch   aziz.djelouah@disneyresearch.com

Abstract

*Most approaches for video frame interpolation require accurate dense correspondences to synthesize an in-between frame. Therefore, they do not perform well in challenging scenarios with e.g. lighting changes or motion blur. Recent deep learning approaches that rely on kernels to represent motion can only alleviate these problems to some extent. In those cases, methods that use a per-pixel phase-based motion representation have been shown to work well. However, they are only applicable for a limited amount of motion. We propose a new approach, **PhaseNet**, that is designed to robustly handle challenging scenarios while also coping with larger motion. Our approach consists of a neural network decoder that directly estimates the phase decomposition of the intermediate frame. We show that this is superior to the hand-crafted heuristics previously used in phase-based methods and also compares favorably to recent deep learning based approaches for video frame interpolation on challenging datasets.*

1. Introduction

Video frame interpolation is a classic problem in video processing and has many applications ranging from frame rate conversion to slow motion effects. Traditionally this problem is formulated as finding correspondences between consecutive frames which are then used to synthesize the in-between frames through warping. These methods [5, 36, 40] usually suffer from the inherent ambiguities in estimating the correspondences and are particularly sensitive to occlusions/dis-occlusion and changes in color or lighting.

To overcome the limitations of traditional methods two main directions have been explored. The first [8, 28] relies on phased-based decomposition of the input images, but methods in this category are limited in the range of motion they can handle. The second direction is based on recent ad-

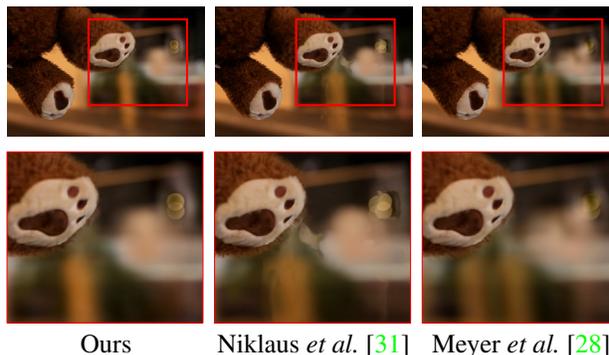Ours   Niklaus *et al.* [31]   Meyer *et al.* [28]

Figure 1: **Video frame interpolation.** Compared to recent kernel based method [31], our approach is able to handle complex scenarios containing motion blur or light changes. It also improves over existing phase-based interpolation methods [28] relying on heuristics, which are limited in their motion range. (Image source: [21])

vances in deep learning [31]. These methods have largely improved over optical flow based methods, but are still not able to handle challenging scenes containing light changing and motion blur.

In this work we propose a novel neural network architecture, PhaseNet, which combines the phase-based approach with a learning framework. PhaseNet mirrors the hierarchical structure of the phase decomposition which it takes as input. It then predicts the phase and amplitude values of the in-between frame level by level. The final image is reconstructed from these predictions at different levels. Therefore, PhaseNet is able to handle a larger range of motion than existing phase-based methods [28] (which use hand-tuned parameters) while addressing the issues of optical flow and kernel based methods [31].

PhaseNet processes channels of the input images independently and shares weights across channels and pyramid levels and as such requires a relatively small number of parameters.

Furthermore, we introduce a *phase loss*, which is based on the phase difference between the prediction and the

---

ground truth and encodes motion relevant information. To improve training efficiency and stability, PhaseNet is trained hierarchically starting from the coarsest scale and proceeding incrementally to the next finest scale. Altogether, we show that this allows us to outperform existing state-of-the-art methods for video frame interpolation in challenging scenarios.

2. Related Work

Intermediate frames of a video sequence are commonly obtained by interpolating an optical flow field [5] representing a dense correspondence field between images. Therefore the final interpolation result is heavily dependent on the accuracy of the computed flow. However, finding a pixel-accurate mapping is an inherently ill-posed problem. Existing approaches usually require computationally expensive regularization and optimization, see [40] for a thorough analysis. Furthermore, they often rely on the brightness constancy constraint and therefore have difficulties handling scenes with large changes in brightness, although small changes can be handled by working in the gradient domain [25]. Alternatevly, Fleet *et al.* [11] suggest to use a phase constancy constraint to compute the optical flow and recently, a pure phase-based interpolation method was proposed [28]. By using per-pixel modifications and not computing explicit correspondences, such an approach is more stable to lighting changes. Its main drawback is the limit in the range of motion and the heuristics it introduces. Phase-based motion representations have also been used for various other applications, such as motion magnification [44, 10, 47], light-fields [48], image editing [27] and image animation [35]. Approaches to extend the motion range have been proposed, e.g., by combining it with optical flow [10] or by computing a disparity map [48]. In this work we increase the robustness by combining it with a neural network.

Neural networks have enjoyed a recent resurgence in popularity due to the huge growth in data and computational resources which has allowed models to be trained successfully [20, 6]. They have achieved state-of-the-art performance in a variety of applications domain such as large-scale image and video classification, detection, localization and recognition (e.g. [19, 39, 41, 15]). Most models for these tasks are trained in a supervised manner, requiring large amounts of labeled data. Supervised methods [9, 16, 42] have also been suggested for optical flow estimation. However, this requires a large volume of ground-truth optical flow data. To estimate optical flow without ground-truth data Long *et al.* [23] synthesize interpolated frames as an intermediate result.

Neural networks have been applied for image synthesis in various contexts [26, 12, 18]. Directly predicting images often produce blurry results [14, 43, 46]. Instead of pre-
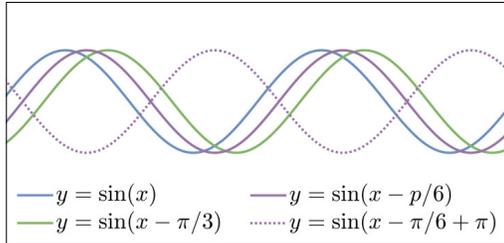

Figure 2: **Interpolation as phase shift.** The translation of a simple sinusoidal function (blue to green) can be expressed by the phase difference. To estimate the middle signal, phase-based interpolation needs to determine the correct phase value among the two possible solutions in purple.

dicting pixel value, Zhou *et al.* [49] predict an *appearance* flow and use it to warp pixels and synthesize novel viewpoints. In the same spirit, Liu *et al.* [22] propose to train a convolution neural network to synthesize an intermediate frame by flowing and blending pixel values from the existing input frames according to the predicted voxel flow. Niklaus *et al.* [31, 30] combine motion estimation and image synthesis into a single convolution step. These methods generally result in sharp images and already better handle challenging situations—such as brightness changes—than traditional optical flow methods. However, in these scenarios, we show that our phase-based approach performs better.

3. Motion Representation

Similar to previous works, we base our method on the intuition that motion of certain signals can be represented by the change of their phase [28, 44]. Our goal is to directly estimate the phase value of the intermediate image. To illustrate our motivation, we adapt the example used in [28] followed by a similar review of the phase-based image decomposition for completeness.

Motivation. We first introduce the concept and challenges of phase-based motion representation. To illustrate them we use one dimensional sinusoidal functions $y = A\sin(\omega x - \phi)$, where A is the amplitude, ω the angular frequency and ϕ the phase. Assuming we have two functions, which are defined as $y = sin(x)$ and $y = sin(x - \pi/3)$, for example. Graphically they represent the same sinusoidal function but one is translated by $\pi/3$, see Figure 2. The translation, i.e. the motion, can be represented by the phase difference of $\pi/3$. This demonstrates the general idea of representing motion as a phase difference. In terms of frame interpolation, these two curves (blue and green) would correspond to the input images. An in-between curve would then represent the interpolated intermediate image. But due to the 2π-ambiguity of phase values (i.e. $y = sin(x - \pi/3) = sin(x - \pi/3 + 2\pi)$) there exists two valid solutions, namely $y = sin(x - \pi/6)$ (purple)

and $y = sin(x - \pi/6 + \pi)$ (purple dotted). The difficulty of phase-based frame interpolation is to determine, which is the correct solution. While [28] describes a heuristic on how to correct the phase difference to correspond to the actual spatial motion. In this work we propose to learn to directly predict the phase value of the desired intermediate result.

Image decomposition. More complex one dimensional functions can be represented in the Fourier domain as a sum of complex sinusoids over all frequencies ω:

$$f(x) = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\phi_\omega} . \tag{1}$$

Images can be seen as two dimensional functions which can be represented in the Fourier domain as a sum of sinusoids over not only different frequencies but also over different spatial orientations. This decomposition of the image can be obtained by using e.g. the complex-valued steerable pyramid [34, 37, 38]. By applying the steerable pyramid filters $\Psi_{\omega,\theta}$, consisting of quadrature pairs, we can decompose an image into a set of scale and orientation depended complex-valued subbands $R_{\omega,\theta}(x,y)$:

$$R_{\omega,\theta}(x,y) = (I * \Psi_{\omega,\theta})(x,y) \tag{2}$$
$$= C_{\omega,\theta}(x,y) + i\,S_{\omega,\theta}(x,y) \tag{3}$$
$$= A_{\omega,\theta}(x,y)\,e^{i\phi_{\omega,\theta}(x,y)} , \tag{4}$$

where $C_{\omega,\theta}(x,y)$ is the cosine part and $S_{\omega,\theta}(x,y)$ the sine part. Because they represent the even-symmetric and odd-symmetric filter response, respectively, it is possible to compute for each subband the amplitude

$$A_{\omega,\theta}(x,y) = |R_{\omega,\theta}(x,y)| \tag{5}$$

and the phase values

$$\phi_{\omega,\theta}(x,y) = \text{Im}(\log(R_{\omega,\theta}(x,y))) , \tag{6}$$

where *Im* represents the imaginary part of the term. The frequencies which can not be captured in the pyramid levels will be summarized in real valued high- and low-pass residuals r_h and r_l, respectively. This decomposition of the image will be used as input to our network.

Phase prediction. The goal of our network is to predict the phase values of the intermediate frame, based on the steerable pyramid decomposition of the input frames. Each level of the multi-scale pyramid represents a band of spatial frequencies. The phase computation according to Equation (6) yields phase values between $[-\pi, \pi]$ for every pixel at each resolution.

We have seen earlier that there exists two solutions for the middle frame. Furthermore, the assumption that motion is encoded in the phase difference is only accurate for small motion, i.e. the lower levels of the pyramid. Due to the frequency banded filter design the response value is based on a locally limited spatial area. On the higher levels the motion could be larger than the receptive field of the filters. As a consequence, the phase values of a pixel at two different time steps are not comparable anymore. By assuming that large motion is already visible and captured correctly by the phase on a lower level, this information can be used to improve the prediction on the higher levels. Instead of using heuristics [28] to propagate the information upwards in the pyramid, we propose using a convolutional network to learn how to combine the available phase information.

4. Method

The aim of the network is to synthesize an intermediate image given its two neighboring images as input. Instead of directly predicting the color pixel values, our network predicts the values of the steerable pyramid decomposition.

4.1. Learning Phase-based Interpolation

The color input frames I_1 and I_2 are decomposed using the steerable pyramid (Eq. (2)). We denote the obtained decomposition as R_1 and R_2, respectively:

$$R_i = \Psi(I_i) = \{\{(\phi_{\omega,\theta}^i, A_{\omega,\theta}^i)|\omega,\theta\}, r_l^i, r_h^i\} . \tag{7}$$

These decomposition responses R_1 and R_2 are the inputs to our network. Using these values, the objective is to predict $\hat{R}$, the decomposition of the interpolated frame. The prediction function, $\mathcal{F}$ is a CNN with parameters Λ. The interpolated frame $\hat{I}$ is given by

$$\hat{I} = \Psi^{-1}(\hat{R}) = \Psi^{-1}(\mathcal{F}(R_1, R_2; \Lambda)) , \tag{8}$$

where Ψ^{-1} the reconstruction function.

The network is trained to minimize the objective function $\mathcal{L}$ over the dataset $\mathcal{D}$ consisting of triplets of input images (I_1, I_2) and the corresponding ground truth interpolation frame, I:

$$\Lambda^* = \arg\min_\Lambda \mathbb{E}_{I_1,I_2,I\sim\mathcal{D}}[\mathcal{L}(\mathcal{F}(R_1, R_2; \Lambda), I)] . \tag{9}$$

Our objective is to predict response values $\hat{R}$ that lead to a reconstructed image similar to I. We also penalize the deviation from the ground truth decomposition R. This is reflected in our loss function that consists of two terms: an image loss and a phase loss.

Image loss. For the image loss we use the ℓ_1-norm of pixel differences which has been shown to lead to sharper results than ℓ_2 [24, 26, 31]:
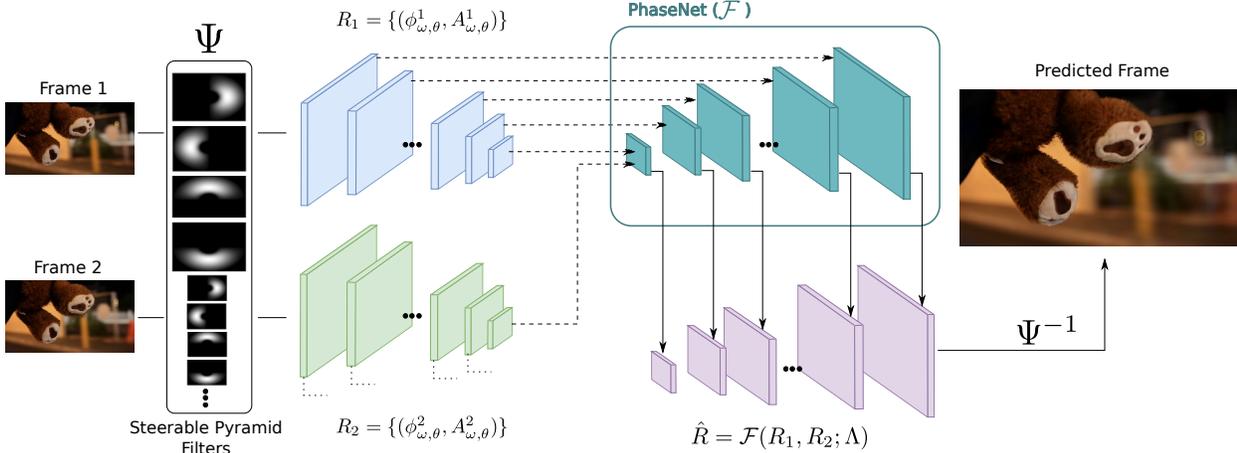
$$\mathcal{L}_1 = ||I - \hat{I}||_1 . \tag{10}$$

Figure 3: **PhaseNet architecture.** Given two consecutive frames, their decomposition can be obtained by applying the steerable pyramid filters (Ψ). The decomposition of these two input frames (denoted as R_1 and R_2) are the inputs to our network: PhaseNet, which has a decoder only architecture. The number of layers and their dimensions mirror the input frame decompositions. We only display the blocks of each level (the details of the blocks are discussed later). Each block takes as input the decomposition values from the corresponding level. We only display the links from the decomposition of the first frame to avoid cluttering the image. The predicted filter responses ($\hat{R}$) are then used to reconstruct the middle frame.

Phase loss. The predicted decomposition $\hat{R}$ of the interpolated frame consists of amplitude and phase values for each level and orientation present in the steerable pyramid decomposition. To improve the quality of the reconstructed images we add a loss term which captures the deviations $\Delta\phi$ of the predicted phase $\hat{\phi}$ from the ground truth phase ϕ. The phase loss is then defined as the ℓ_1 loss of the phase difference values over all levels (ω) and orientations (θ):

$$\mathcal{L}_{\text{phase}} = \sum_{\omega,\theta} ||\Delta\phi_{\omega,\theta}||_1 \ , \qquad (11)$$

where $\Delta\phi$ is defined as

$$\Delta\phi = \text{atan2}(\sin(\phi - \hat{\phi}), \cos(\phi - \hat{\phi})) \ . \qquad (12)$$

We use atan2, the four-quadrant inverse tangent, which returns the smaller angular difference between ϕ and $\hat{\phi}$.

We could also define a similar loss on the predicted amplitude values $\hat{A}_{\omega,\theta}$ but we found that it did not improve over the combination of phase and image loss in practice. As motion is primarily encoded in the phase shift, it is more important to enforce correct phase prediction.

We define our final loss as a weighted sum of the image loss and the phase loss:

$$\mathcal{L} = \mathcal{L}_1 + \nu\mathcal{L}_{\text{phase}} \ . \qquad (13)$$

In our experiments the weighting factor ν is chosen such that the phase loss is one order of magnitude larger than $\mathcal{L}_1$, i.e. $\nu = 0.1$.

4.2. Network Architecture

The architecture of PhaseNet is visualized in Figure 3. The design is inspired by the steerable pyramid decomposition. For each resolution level it predicts the values of the corresponding level of the pyramid decomposition of the intermediate frame. It is structured as a decoder-only network increasing resolution level by level. At each level we incorporate the corresponding decomposition information from the input images. Besides the lowest level, due to the steerable pyramid decomposition, all other levels are structurally identical. At each level we also incorporate the information from the previous level. This follows the assumption that motion will be captured at different scales and the phase values do not differ arbitrarily from level to level.

As input to the network we use the response values from the steerable pyramid decomposition of the two input frames consisting of the phase $\phi_{\omega,\theta}$ and amplitude $A_{\omega,\theta}$ values for each pixel at each level ω and orientation θ, as well as the low pass residual. Before passing them through the network we normalize the phase values by dividing by π. The residual and amplitude values are normalized by dividing by the maximum value of the corresponding level.

Each resolution level consist of a PhaseNet block (Figure 4) which takes as input the decomposition values from the input images, the resized feature maps from the previous level as well as the resized predicted values from the previous level. This information is passed through two convolution layers each followed by batch normalization [17] and ReLU nonlinearity [29], which have shown to help training. Each convolution layer produces 64 feature maps by either
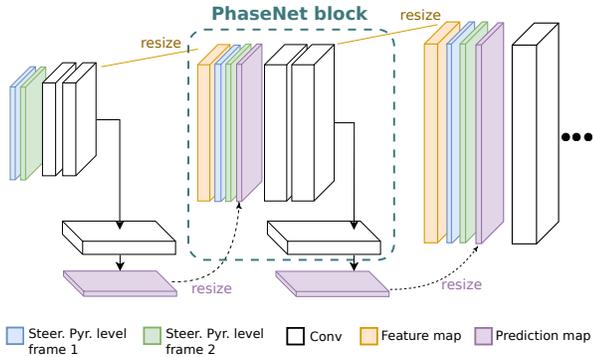
Figure 4: **PhaseNet block.** Each block of the PhaseNet takes as input the decompositions of the input frames at current level (shown in blue and green). Each level performs two successive convolutions with batch normalization and ReLU. From the intermediate features map, each block predicts the response (amplitude and phase) at current level with one convolution layer followed by the hyperbolic tangent function. Feature map and predicted values are reused in the next block after resizing.

using 1×1 or 3×3 convolution filters (see supplementary material for details). In general, we observe, that smaller kernels are preferable for lower resolution. Between levels the resolution is increased by the scaling factor λ, which has been used to produce the steerable pyramid. Resizing is done by bilinear interpolation. On the lowest level, the first PhaseNet block receives as input only the concatenation of the two low level residuals of the two input frames.

After each PhaseNet block we predict the values of the in-between frame decomposition by passing the output feature maps of the PhaseNet block through one convolution layer with filter size 1×1 followed by the hyperbolic tangent function to predict output values within the range of $[-1, 1]$. From these we can compute the decomposition values $\hat{R}$ of the intermediate image and reconstruct it, see Section 4.3. The number of output channels depends on the number of predicted values for each pixel, i.e. d for the lowest level, and $2bd$ for the intermediate levels, where we predict phase and amplitude for each dimension d and orientation b.

In our case, the network is built for a single color dimension (i.e. $d = 1$) and trained for color images by reusing the weights across the color channels. This allows to significantly reduce the weights while producing comparable results. To process higher resolutions at testing time we share the weights of the highest three levels. We describe this in Section 5.

4.3. Image Reconstruction

In general we can reconstruct an image from the steerable pyramid decomposition by integrating over all pyramid levels according to Equation 1 and adding the low and high pass residual. Due to the normalization of the steerable pyramid values before passing them through PhaseNet and by predicting values between $[-1, 1]$ we need to remap the predicted values before we can reconstruct the image. The following remapping is applied to each pixel (x, y) at each level ω and orientation θ.

To compute the phase values $\hat{\phi}$ of $\hat{R}$ we scale the predicted values by multiplying them with π. To approximate the low level residuals and the amplitudes of the intermediate frame [28] propose to average the values. This works well for lower levels where these values correspond mainly to global luminance changes. For higher frequency bands, averaging the amplitude values can lead to artifacts. For more flexibility, instead of exactly averaging, we allow the network to learn the mixing factors.

The low level residual, $\hat{r}_l$ as well as the amplitude values $\hat{A}$ of $\hat{R}$ are computed using the predicted values as a linear scaling factor between the values of the input decompositions R_1 and R_2:

$$\hat{r}_l = \alpha * r_l^1 * (1 - \alpha) * r_l^2 \,, \tag{14}$$

$$\hat{A} = \beta * A^1 + (1 - \beta) * A^2 \,, \tag{15}$$

where α and β are the learned mixing weights mapped to $[0, 1]$. We observe that the high pass residual can be ignored as the introduced blur is often very subtle.

4.4. Training and Implementation Details

Each pixel in the synthesized image is influenced by the predicted phase and amplitude values from all scales. For stability, we adopt a hierarchical training procedure where the layers at lowest levels are trained first. When training the first m levels, we still need to reconstruct the interpolated frame to compute the loss. In this case we use ground truth response values for levels $m + 1, \ldots, n$ as illustrated in Figure 5.

This training procedure can be seen as a form of curriculum learning [7] that aims at improving training by gradually increasing the difficulty of the learning task. This type of learning strategy is often used in sequence prediction tasks and in sequential decision making problems where large speedups in training time and improvements in generalization performance can be obtained.

Our training procedure is related to the filtered scheme adopted in [13] where ground truth masks are first blurred then smoothly sharpened over time. In our case, by using a steerable pyramid decomposition we have already a coarse to fine representation of the image which is well suited for such a hierarchical training procedure. It also matches the assumption that the motion and therefore pyramid values of higher, finer levels are related to the previous, lower levels.

For training we use triplets of frames from the DAVIS video dataset [33, 32], randomly selecting patches of $256 \times$
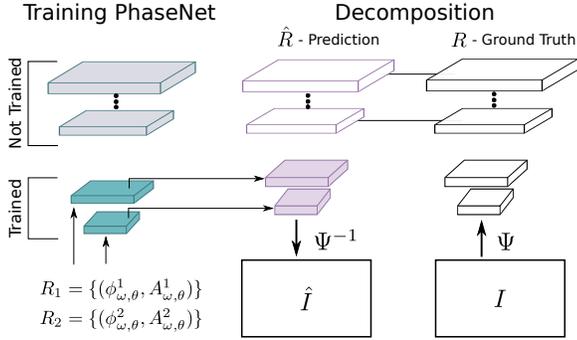
Figure 5: **Hierarchical training.** On the left, PhaseNet takes as input the decompositions R_1 and R_2 of the input frames. In this example the two lowest levels are being trained ($m = 2$). Corresponding blocks are displayed in green. The other blocks (in gray) will be added at the next iteration. On the right, we have the ground truth frame decomposition R. To reconstruct the predicted image, we use ground truth values for the layers not being trained yet.

256 pixels. To build the pyramid decomposition we use a scale factor of $\lambda = \sqrt{2}$ leading to a pyramid of 10 levels. More details on the training procedure can be found in the supplementary material.

Computation Time. PhaseNet is implemented in Tensorflow and takes advantage of efficient spectral decomposition layers. With one Nvidia Titan X (Pascal), training our model ($\sim 460k$ parameters) takes approximately 20h in total for 9 hierarchical training stages. Computation time for decomposition, interpolation and image reconstruction is $0.5s$ for 256×256 patches (training) and $1.5s$ for 2048×1024 images (testing).

5. Results

We compare our method with a representative selection of state-of-the-art methods by evaluating them quantitatively and qualitatively on various images. As a representative of optical flow we chose MDP-Flow2 [45], which currently performs best on the Middlebury benchmark for interpolation. To synthesize the interpolated frames from the computed optical flow field, we use the same algorithm as used in the benchmark [5]. According to Middlebury, MDP-Flow2 is followed closely by [31], a neural network based method learning seperable convolution filters for frame interpolation (SepConv). In terms of phase-based representation methods for frame interpolation we compare to [28] (Phase). The image sequences used are from the footage of [21], Blender Foundation [1], Vision Research [2] and YouTube [3, 4]. To produce the results of these methods, we use the code and trained models provided by the original authors.
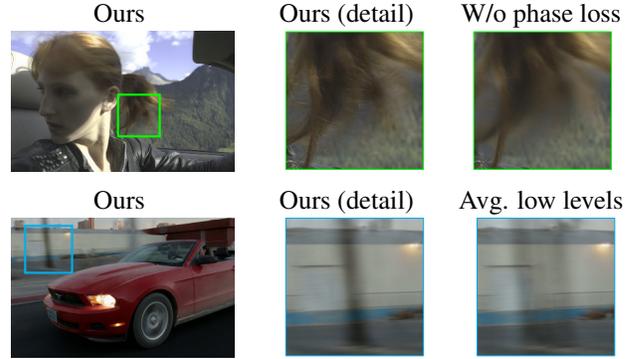

Figure 6: **Design choices.** The first row shows the benefit of using the phase loss giving sharper results compared to only using the image loss (best viewed on screen). For images larger than the training patches, the second row shows the benefit of reusing last layers weights over averaging the lowest levels of the decomposition. (Image source: [3, 21])

Loss function. For training our network we use the combination of the two loss functions: the image loss ($\mathcal{L}_1$) and the phase loss ($\mathcal{L}_{phase}$). Training only with the image loss already produces reasonable interpolation results. Because the phase loss is computed at each resolution level and encodes motion relevant information, it is necessary to achieve sharp results, see Figure 6 (top). Furthermore, we observe that optimizing for the phase loss additionally to the image loss stabilizes the training procedure and helps to reduce training time. For our final results we use a linearly weighted combination of both terms, see Eq. (13). We did not notice any particular sensitivity of the results regarding the weighting factor ($\nu \in [0.1, 1]$). Using only the phase loss is however not sufficient.

High resolution data. Because we are using a fully convolutional network, we are able to handle larger images at testing time. Our network is trained on patches of 256×256 leading to a pyramid of 10 levels. To produce higher resolution images during testing, we need to extend the pyramid. We test our algorithm on images of 1280×720. For stability of the used Fast Fourier transform and the pyramid decomposition we symmetrically pad the images to 2048×1024 leading to 14 pyramid levels. A naive approach would be to consider averaging the phase values at the lower levels and use our model only on the 10 highest levels. However, this implicitly limits the range of motion we can interpolate, see Figure 6 (bottom right). A better approach is to reuse the weights of the trained higher levels for the following, additional layers. Because they shared their weights during training over several levels, this approach generalizes well to further levels, see Figure 6 (bottom middle).

Qualitative comparisons. We evaluate our method on a set of challenging image pairs including motion blur and
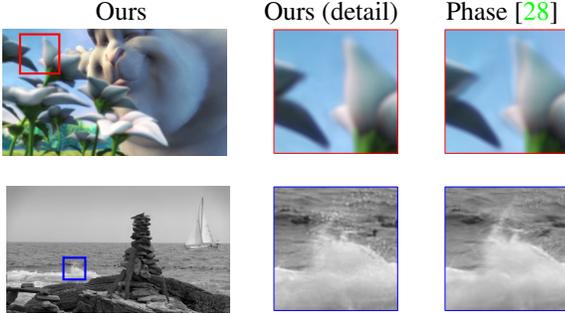
Figure 7: **Advantage of a data driven approach.** Using heuristics [28] for phase-based frame interpolation reaches its limits in these two examples. Our data driven approach is able to better handle large motion and obtains sharper results. (© Blender Foundation [1], © Vision Research [2])

extreme light changes, see Figure 10. Because optical flow based methods, such as MDP-Flow2, compute explicit pixel correspondences it produces visible artifacts once the used brightness constancy assumption is violated. The pure phase-based method as well our phase-based-network combined approach, on the other hand, are robust against such lighting changes and produce smooth and plausible results. In the case of the explosion scene in the second row, our result is even preferable over the pure phase-based approach.

The last two rows show some examples with motion blur. The pure phase-based approach is limited in the amount of motion it can handle. This is visible in the last row, where the pole in the background moves too far to be correctly captured by the method resulting in ghosting artifacts. In this example SepConv is unable to correctly interpolate the car due to the motion blur. Our method improves on both of them. However, the frequency banded filters influence some area around each in pixel in the spatial domain. As a result, reduced accuracy in the phase prediction can lead to some minor ringing and color artifacts during reconstruction. These are noticeable around high frequency edges. Although both phase-based methods have this issue in common, the main improvement of PhaseNet over the pure phase-based methods is visible in the case of interpolating large motion and high frequencies, as shown in Figure 7.

Quantitative comparisons. We use the same set of sequences as in [28], consisting of representative scenes with many moving parts and challenging lighting conditions as well as one synthetic example (*Roto*) containing many high frequencies. For quantitative evaluation, we compare several methods on a number of sequences using the leave-one-out method, where we compare synthesized frames to the original ones. In Figure 8 we report the error measurements using the structural similarity (SSIM) measure. In general, the optical flow method and SepConv achieve a better error
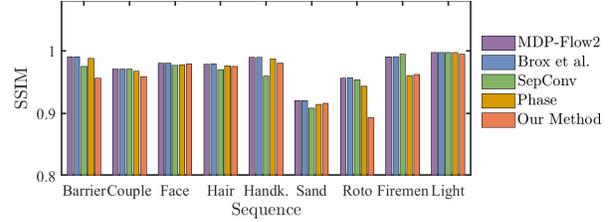

Figure 8: **Error measurements** of different methods for different sequences by computing the structural similarity measuremnt (SSIM) averaged over several frames. Example images of the evaluated sequences are shown in the supplementary material.
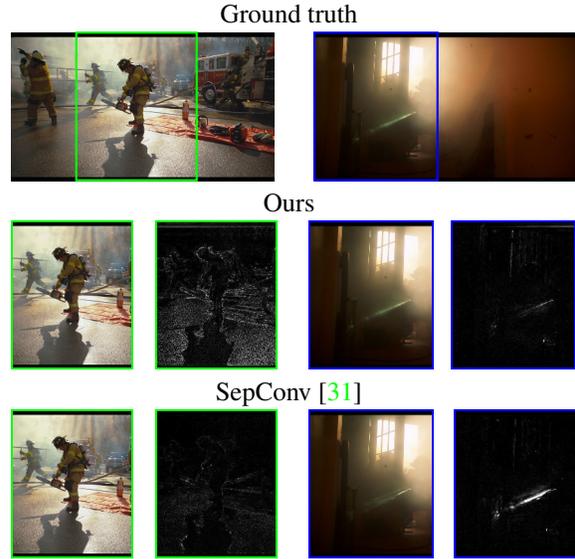

Figure 9: **Comparison** of interpolation results with our method and separable convolution filters to the ground truth including a difference map using absolute differences. Best viewed on screen. (© Vision Research [2])

measure, mainly due to the fact they introduce less blur. Especially for the sequences with high frequencies (*barrier*, *fireman*, *sand* and *roto*) we perform worse. The strength of our method lies in handling challenging scenarios with motion blur and brightness changes (e.g. *light* and *handkerchief*). Although the measure is perceptually motivated it does not always reflect the visual comparison, as illustrated in Figure 9. For the *light* sequence (right column), our approach produces noticeably better results. For the *fireman* sequence (left column), although the difference map shows a global degeneration for high frequency content for our method, there is no perceptual difference between the different methods.

Discussion and limitations. Our method significantly improves over previous phase-based methods, both in terms of motion range and high frequencies. It is well suited for scenes with motion blur and difficult light changes. We
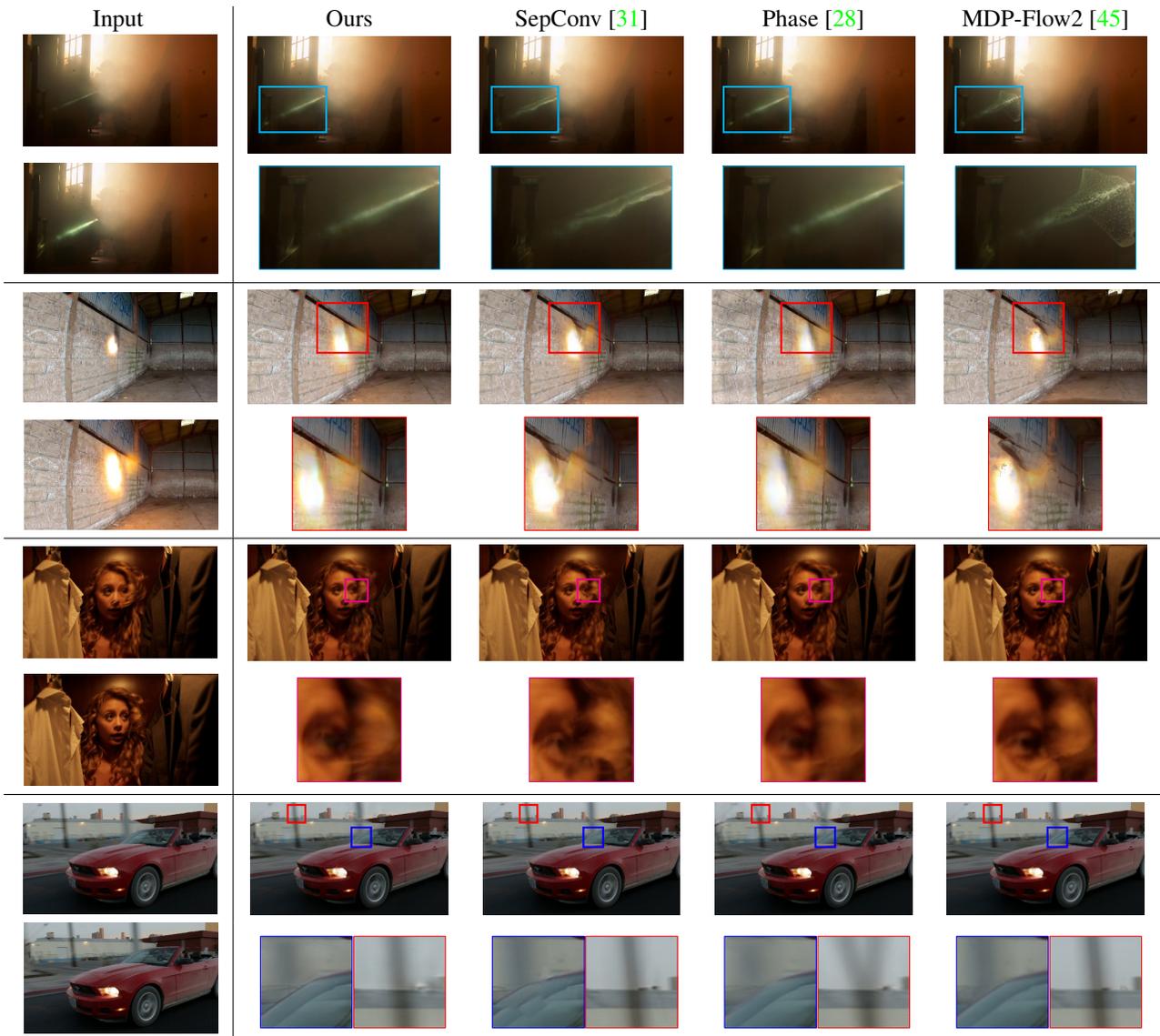
Figure 10: **Visual comparison** with frame interpolation methods on challenging scenarios. See text for details and discussion. (Image source: [2, 4, 21])

still however do not reach the same level of detail as methods which explicitly match and warp pixels. On the other hand these methods may produce more disturbing artifacts whereas our model creates less noticeable effects.

6. Conclusions

We have presented a method which combines the advantage of phase-based and data driven methods for frame interpolation. We propose a neural network architecture that synthesizes an interpolated frame from its predicted phase-based representation. By combining both a phase loss and standard ℓ_1-norm over the reconstructed image we are able to produce visually preferable results over optical flow for challenging scenarios containing motion blur and brightness changes.

References

[1] www.bigbuckbunny.org. 6, 7

[2] www.visionresearch.com/Gallery. 6, 7, 8

[3] https://www.youtube.com/watch?v=3zfV0Y7rwoQ. 6

[4] https://youtu.be/AshgeY5hlec?t=12. 6, 8

[5] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for opti-

cal flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 1, 2, 6

[6] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. 2

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009. 5

[8] 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

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *International Conference on Computer Vision*, pages 2758–2766, 2015. 2

[10] M. A. Elgharib, M. Hefeeda, F. Durand, and W. T. Freeman. Video magnification in presence of large motions. In *Computer Vision and Pattern Recognition*, pages 4119–4127, 2015. 2

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

[12] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Computer Vision and Pattern Recognition*, pages 5515–5524, 2016. 2

[13] M. Gardner, K. Sunkavalli, E. Yumer, X. Shen, E. Gambaretto, C. Gagné, and J. Lalonde. Learning to predict indoor illumination from a single image. *ACM Trans. Graph.*, 36(6):176:1–176:14, 2017. 5

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances In Neural Information Processing Systems*, pages 2672–2680, 2014. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[16] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Computer Vision and Pattern Recognition*, pages 1647–1655. IEEE Computer Society, 2017. 2

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 4

[18] N. K. Kalantari, T. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 35(6):193:1–193:10, 2016. 2

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances In Neural Information Processing Systems*, pages 1106–1114, 2012. 2

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 2

[21] W. Li, F. Viola, J. Starck, G. J. Brostow, and N. D. Campbell. Roto++: Accelerating professional rotoscoping using shape manifolds. *ACM Trans. Graph.*, 35(4), 2016. 1, 6, 8

[22] Z. Liu, R. Yeh, X. Tang, Y. Liu, , and A. Agarwala. Video frame synthesis using deep voxel flow. In *International Conference on Computer Vision*, 2017. 2

[23] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450, 2016. 2

[24] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450, 2016. 3

[25] 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):42:1–42:11, 2009. 2

[26] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 2, 3

[27] S. Meyer, A. Sorkine-Hornung, and M. H. Gross. Phase-based modification transfer for video. In *European Conference on Computer Vision*, pages 633–648, 2016. 2

[28] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *Computer Vision and Pattern Recognition*, pages 1410–1418, 2015. 1, 2, 3, 5, 6, 7, 8

[29] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *International Conference on Machine Learning*, pages 807–814. Omnipress, 2010. 4

[30] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Computer Vision and Pattern Recognition*, pages 2270–2279, 2017. 2

[31] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *International Conference on Computer Vision*, 2017. 1, 2, 3, 6, 7, 8

[32] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016. 5

[33] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 5

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

[35] E. Prashnani, M. Noorkami, D. Vaquero, and P. Sen. A phase-based approach for animating images using video examples. *Comput. Graph. Forum*, 36(6):303–311, 2017. 2

[36] E. Shechtman, A. Rav-Acha, M. Irani, and S. M. Seitz. Regenerative morphing. In *Computer Vision and Pattern Recognition*, pages 615–622, 2010. 1

[37] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative com-

putation. In *Proceedings 1995 International Conference on Image Processing*, pages 444–447, 1995. 3

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

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[40] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. 1, 2

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, pages 1–9, 2015. 2

[42] D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428, 2016. 2

[43] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016. 2

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

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

[46] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2016. 2

[47] Y. Zhang, S. L. Pintea, and J. C. van Gemert. Video acceleration magnification. In *Computer Vision and Pattern Recognition*, 2017. 2

[48] Z. Zhang, Y. Liu, and Q. Dai. Light field from micro-baseline image pair. In *Computer Vision and Pattern Recognition*, pages 3800–3809, 2015. 2

[49] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301, 2016. 2