# Phase-based Modification Transfer for Video

Simone Meyer[1,2] Alexander Sorkine-Hornung[2] Markus Gross[1,2]

[1]Department of Computer Science, ETH Zurich [2]Disney Research
simone.meyer@inf.ethz.ch alex@disneyresearch.com

**Abstract.** We present a novel phase-based method for propagating modifications of one video frame to an entire sequence. Instead of computing accurate pixel correspondences between frames, e.g. extracting sparse features or optical flow, we use the assumption that small motion can be represented as the phase shift of individual pixels. In order to successfully apply this idea to transferring image edits, we propose a correction algorithm, which adapts the phase shift as well as the amplitude of the modified images. As our algorithm avoids expensive global optimization and all computational steps are performed per-pixel, it allows for a simple and efficient implementation. We evaluate the flexibility of the approach by applying it to various types of image modifications, ranging from compositing and colorization to image filters.

**Keywords:** Phase-based method, video processing, edit propagation.

## 1 Introduction

Many applications in video processing, e.g., frame interpolation or edit propagation, require some form of explicit correspondence mapping between pixels in consecutive frames. Common approaches are based on matching sparse feature points, or dense optical flow estimation. However, finding a pixel-accurate mapping is an inherently ill-posed problem, and existing dense approaches usually require computationally expensive regularization and optimization.

Recently, a number of novel phase-based video processing techniques have been proposed that are able to solve certain types of problems *without* the need for explicit correspondences. Examples include motion magnification [20], view synthesis for autostereoscopic displays [5], or frame interpolation for video [13]. The interesting advantage of such techniques over explicit methods is that they are based on efficient, local per-pixel operations, which do not require knowledge about the actual image-space motion of pixels between frames, and hence avoid the need for solving the above mentioned optimization problems. On the other hand, the price is that phase-based methods are limited to much smaller motions between frames than, e.g., methods for sparse feature point matching. However, given today's steady increase in video resolution and frame rate, there is also an increasing need for computationally simple and efficient methods.

In this paper, we extend the range of possible applications for phase-based techniques. We introduce a method to propagate various types of image modifications over a sequence of video frames, without the need for explicit tracking

or correspondences. As previous phase-based approaches, we decompose each frame of a video sequence using a complex-valued steerable pyramid into local phase and amplitude information. The key question we then address is how to adjust both phase and amplitude in this decomposition on subsequent frames in order to transfer edits made on the first frame of a sequence to all other frames. A particular feature of our method is that it works on textureless or homogeneous image regions, where explicit tracking approaches often struggle or require strong regularization. We present various applications of our algorithm, from adding novel image elements like a logo on a surface and video colorization to propagation of general image filters.

## 2 Related Work

**Correspondence-based methods.** Most methods for transferring modifications from one image to others require some form of explicit correspondences between the pixels. General approaches for such correspondence estimation techniques range from dense optical flow [1] to tracking of sparse features like SIFT [11]. Some early work using optical flow for propagation edits is proposed by Levin et al. [9]. Such methods often require expensive global optimization which is difficult to implement and parallelize.

Tracking particular image elements is a long-studied problem as well [12,17]. It has been used for propagating image edits in unordered image collections [7,23] as well as for video [16]. While methods for image collections can handle large displacements well, they are lacking temporal coherence to avoid artifacts such as flickering when applied to video. A further limitation of tracking-based approaches is that they usually require sufficiently well textured surfaces. More robust are template-based methods [14] for video editing, as they also work with minimal texture and deformable surfaces. However, the template image and the restshape needs to be known in advance, which is usually not the case for general videos.

Another area related to our work is appearance editing like color manipulation and colorization. In these methods, sparse user edits get propagated spatially and temporally throughout a video, usually by solving optimization problems proportional to resolution and number of frames. Recent works therefore focused in particular on reducing high computational costs and memory consumption [2, 10, 21]. Yatagawa et al. [22] propose a method independent of the total length of the video as it only processes two frames at the time. A general approach to ensure temporal consistency for various applications including optical flow and colorization has been proposed by Lang et al. [8]. Instead of optimizing directly for temporal consistency, Bonneel et al. [3] propose a method to restore temporal consistency after a filter operation has been applied to each frame of a video independently. This method also uses optical flow as guidance and assumes that the filter does not generate new content uncorrelated to its input. In contrast to such approaches, the advantage of our method is that it

allows the propagation of global modifications without the knowledge of the particular used image filter.

Our correspondence-free, phase-based method is designed such that it can handle appearance editing, e.g. local recolorization and global changes by an image filter, as well as detailed edits such as adding novel image elements on a surface, given that image motion between video frames is small.

**Phase-based methods.** Recent works have shown that it is possible to use a phase representation of the motion between frames for various applications [5,20] that usually require explicit correspondences. Such a phase-based representation allows for efficient computation as only per-pixel modifications are required. As a drawback, they are limited to small motions between the frames. Effort has been put into extending the motion range, e.g., by combining it again with tracking or optical flow [6] or by computing a disparity map [24]. A purely phase-based approach to extend the range of admissible motion for video frame interpolation has been proposed by Meyer et al. [13].

All these methods have been used for applications that modify or interpolate the unmodified input frames. Complementary, we extend the set of phase-based applications by a method to propagate edits of modified frames, which can significantly differ from the input frames on which the phase information has been originally computed.

## 3 Motion as Phase Shift

Phase-based methods use the intuition that the motion of certain signals or functions can be represented as a shift of their phase. In this section we first explain the basic mathematical justification for the 1D case as well as derive the consequences and challenges when using it for propagating a *modified* signal.

**1D case.** The Fourier Shift Theorem motivates the assumption that some small displacement motion can be encoded using phase differences. In the one dimensional case, a function $f(x)$ can be represented in the Fourier domain as a sum of complex sinusoids over all frequencies $\omega$:

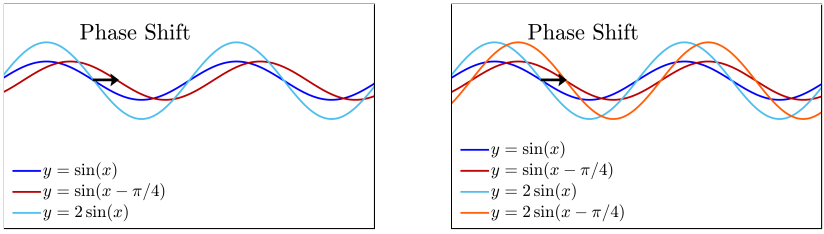
$$f(x) = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\omega x} = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\phi_\omega} \ , \tag{1}$$

where $A_\omega$ and $\phi_\omega$ represent the amplitude and the phase, respectively. The shifted version of $f(x)$ by a displacement function $\delta(t)$ is then defined as:

$$f(x - \delta(t)) = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\omega(x-\delta(t))} \ . \tag{2}$$

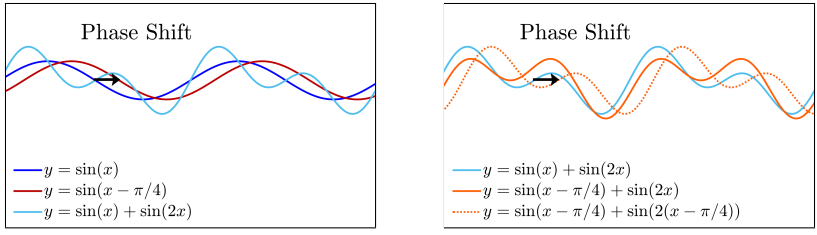
The phase difference between the original and the shifted function

$$\phi^\omega_{diff} = \omega x - \omega(x - \delta(t)) = \omega\delta(t) \tag{3}$$




(a) (b)

Fig. 1: Left: Given a simple sinusoidal input function (blue) and its translated version (red), a modification of the input (cyan) can be translated using the phase difference of the unmodified functions (Equation 4) (orange, right).




(a) (b)

Fig. 2: For less trivial modifications of the input function, e.g., adding an additional frequency (left), transferring the modification using only the known phase difference (orange solid) does not correspond to the actually required, but generally unknown frequency dependent phase shift (orange dotted).

encodes the frequency-dependent version of the spatial displacement $\delta(t)$. In the context of phase-based methods $\delta(t)$ is also referred to as phase shift $\phi_{shift}$.

For propagating modifications we are interested in using this phase difference to translate a modified input function $\hat{f}(x)$. Below we describe the challenges that arise from the fact that the modified function does not have the same frequency decomposition anymore as the original input function.

**Challenges.** Consider the example in Figure 1a, where as an input we are given a function in the form of $f(x) = A\sin(\omega x - \phi)$ (blue). In our targeted application scenario this would correspond to a reference video frame. We also have a modification (cyan), and a translated version of the function (red), which corresponds to the following video frame that we want to propagate the modification to. In this simple example the translation is described by subtracting $\pi/4$ from the phase and the modification consists of replacing the old amplitude with a new amplitude $\hat{A} = 2$. We can compute the translation of the modified function (Figure 1b, orange) by subtracting the phase difference:

$$\hat{f}(x) = \hat{A}\sin(\omega x - \phi_{diff}) = \hat{A}\sin(\omega(x - \phi_{shift}))\ . \tag{4}$$

However, for handling less trivial modifications, e.g., adding new frequencies, we have to decompose the function according to the frequencies and estimate the necessary phase difference for each frequency separately. In our example in Figure 2a this corresponds to the fact that we know the phase difference for the input frequency $\omega = 1$ but not for the added function with $\omega = 2$. This leads us to the main challenge of using phase differences to transfer modifications:

How can we transfer novel frequency content of a modified function that has not been present in the two unmodified input functions?

To solve this problem, we need an algorithm that detects which frequencies have been added in the modified function, and which frequencies in the original input functions represent the relevant motion. Besides addressing this central question, we also resolve some additional, less obvious issues that arise when performing phase-based modification transfer on video sequences.

## 4 Our Algorithm

### 4.1 Decomposition

For images rather than 1D functions, we first need to generalize the idea to two dimensions, where we can separate the sinusoids according to frequency $\omega$ and spatial orientation $\theta$ using, e.g., the complex-valued steerable pyramid [15,18,19]. This step is essentially identical to previous phase-based approaches [5, 13, 20] and we describe it only briefly for completeness. The steerable pyramid filters $\Psi_{\omega,\theta}$ resemble Gabor wavelets and decompose an input image $I$ into oriented frequency bands $R_{\omega,\theta}$:

$$R_{\omega,\theta}(x,y) = (I * \Psi_{\omega,\theta})(x,y) \tag{5}$$

$$= C_{\omega,\theta}(x,y) + i\, S_{\omega,\theta}(x,y) \tag{6}$$

$$= A_{\omega,\theta}(x,y)\, e^{i\phi_{\omega,\theta}(x,y)}\ , \tag{7}$$

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. By using such quadrature filter pairs we can compute the amplitude

$$A_{\omega,\theta}(x,y) = \sqrt{C_{\omega,\theta}(x,y)^2 + S_{\omega,\theta}(x,y)^2}\ , \tag{8}$$

and the phase values

$$\phi_{\omega,\theta}(x,y) = \arctan(S_{\omega,\theta}(x,y)/C_{\omega,\theta}(x,y))\ . \tag{9}$$

For our following algorithm it is important to note that this provides a decomposition with filter responses that are defined in the spatial domain and have local support, providing per-(multi-resolution)-pixel oriented phase and amplitude values. Please see [13, 20] for a more detailed derivation. In general, this decomposition allows the computation of phase differences and amplitudes at various scales and orientations, which is the key element of phase-based methods. Establishing and appropriately using the relationships between different decompositions and across levels is the key element of our algorithm.

### 4.2 Overview

Given the above decomposition for two input images, $I_{t-1}$ and $I_t$, as well as for a modified image $\hat{I}_{t-1}$, our algorithm allows us to recover the unknown, translated version of the modified input $\hat{I}_t$.

To reconstruct $\hat{I}_t$, we need to approximate its filter responses $\hat{A}^t_{\omega,\theta}$ and $\hat{\phi}^t_{\omega,\theta}$ based on the available information. The resulting image is then obtained by integrating the modified responses according to Equation 1. Where clear from the context we omit the indices $\omega$ and $\theta$ in the equations for improved readability. In general, all computations are done for each pixel at each level and orientation.

Using again the assumption that small motion is encoded in the phase shift, we can use the phase difference $\phi_{diff}$ between the phases of the unmodified images $I_{t-1}$ and $I_t$, i.e.,

$$\phi^{t-1,t}_{diff} = \text{atan2}(\sin(\phi_{t-1} - \phi_t), \cos(\phi_{t-1} - \phi_t)) \;, \tag{10}$$

as an initial approximation of the motion. Due to the circular property of the phase values we use the four-quadrant inverse tangent atan2 to get angular phase values between $[-\pi, \pi]$.

The phase of the modified input image can then be translated by subtracting the phase difference from its own phase. Assuming that the motion is small enough such that only the phase is affected, one could try to simply copy the amplitude values, i.e:

$$\hat{\phi}_t = \hat{\phi}_{t-1} - \phi^{t-1,t}_{diff} \;, \tag{11}$$

$$\hat{A}_t = \hat{A}_{t-1} \;. \tag{12}$$

However, as explained in Section 3, and illustrated in Figure 1 and 2, this only works when the modifications do not change the frequency content. Using this initial solution can lead to incomplete or even wrong propagation of some frequency levels. This is the case at locations where the amplitude of the modified image ($\hat{A}_{t-1}$) is large but not the amplitudes of the unmodified images ($A_{t-1}, A_t$). The smaller the amplitude (as a result of weak filter responses in smooth areas), the more noisy are usually the phase values. Artifacts arise in particular when noisy phase values are used for trying to propagate modifications from one image another, and get magnified due to larger corresponding amplitudes of the modified image. Locations with large amplitude values correspond to strong filter responses which have more influence on the final pixel value. Therefore it is important that the corresponding phase values are computed carefully.

We propose an extension to this initial solution in order to handle general modifications, which may alter the decomposition significantly. Furthermore, we are not only interested in propagating the modification to one additional frame but to a whole image sequence. Figure 3 provides an overview of our general procedure. Algorithm 1 provides a summary of the core steps of our algorithm. In order to solve the challenges stated above we have to solve two main tasks: First, determine the pixels per frequency band which contain information

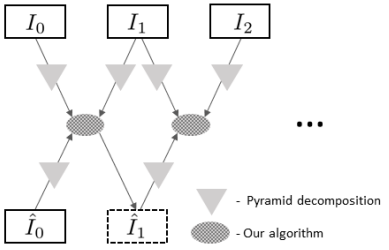



Fig. 3: Illustration of the general procedure. The pyramid decomposition of the input, two unmodified images, $I_0$ and $I_1$, as well as a modified image $\hat{I}_0$, are processed by our algorithm to generate the translated modified image $\hat{I}_1$. By repeating this to the next set of images, the whole sequence can be processed.

**Algorithm 1** The inputs are two unmodified images $I_0$ and $I_1$ and a modified image $\hat{I}_0$. The output is the modified image $\hat{I}_1$. $P_i$ are the steerable pyramid decompositions consisting of $A_i$ and $\phi_i$.

$(P_0, P_1, \hat{P}_0) \leftarrow \text{decompose}(I_0, I_1, \hat{I}_0)$ ▷ [15]

$\phi_{diff} \leftarrow \text{phaseDifference}(\phi_0, \phi_1)$ ▷ Eq. 10
$(A_0, A_1, \hat{A}_0) \leftarrow \text{normalize}(A_0, A_1, \hat{A}_0)$
$\varphi_1 \leftarrow \text{significantMod}(A_0, \hat{A}_0)$ ▷ Eq. 13
$\varrho_1 \leftarrow \text{relevantMotion}(A_0, A_1)$ ▷ Eq. 17
**for all** $l = L - 1 : 1$ **do**
  $\hat{\phi}_1^l \leftarrow \text{compute}(\hat{\phi}_0, \phi_{diff}, \varphi_1, \varrho_1)$ ▷ Eq. 20
**end for**

$\hat{I}_1 \leftarrow \text{reconstruct}(\hat{A}_0, \hat{\phi}_1)$
$\hat{P}_1 \leftarrow \text{decompose}(\hat{I}_1)$
$\hat{A}_1 \leftarrow \text{correctAmpl}(\hat{A}_0, \hat{A}_1, \hat{\phi}_1)$ ▷ Sec. 4.5

$\hat{I}_1 \leftarrow \text{reconstruct}(\hat{P}_1)$ ▷ See [15]

about the motion and those which have been changed due to the modification. Secondly, using this information to approximate the missing information in order to propagate the modifications to succeeding frames. As a guide we can use the amplitude information as larger amplitudes are the result of strong and reliable filter responses. In the following we explain the core algorithmic steps.

### 4.3 Detecting Missing Phase Information

Detecting the locations where we have new frequency content with unknown motion is the first central step in our algorithm. In principle we of course know exactly which pixels in the input image have been modified. However, it is important to consider which modifications result in actual frequency and phase changes, and on which decomposition level these changes happen.

Therefore we perform the detection process in two steps: We first detect pixels with significant modifications, and then decide whether the corresponding phase difference between the unmodified signals represent the motion. To guide this detection process we employ the available amplitude information, which indicates how strong the response at a specific pixel location is.

**Amplitude normalization.** Before we can use the amplitude as a guide we need to normalize the values across the levels such that they are scale-independent and comparable. Because we are downsampling the image during the pyramid decomposition the amplitudes have to be rescaled by the scaling

factor of the pyramid decomposition $\lambda$, i.e., $A(l,x,y) \leftarrow \frac{A(l,x,y)}{\lambda^{l-1}}$, with $l = 1$ being the topmost, i.e. finest, level of the pyramid.

**Identification of significant modifications.** In general, not all modifications result in a significant change on a specific frequency level. Significant means in our case that the modification results in large amplitude values compared to the amplitude values of the unmodified image. Without postprocessing (see next paragraph) possibly noisy phase values will be used. As a first idea to identify significant modifications one could use the difference between the amplitudes of the unmodified and modified image, i.e. $|\hat{A}_{t-1}(x,y) - A_{t-1}(x,y)|$ as a measurement on how much a pixel has changed. In order to get a relative measurement between all pixels and a definition of significance, we need to normalize the differences. Using the absolute difference and a fixed threshold, i.e. $|\hat{A}_{t-1}(x,y) - A_{t-1}(x,y)| > \vartheta$ would be feasible for a single image pair, but as we are interested in propagating the edits over a whole image sequence we need a more robust measurement.

Propagating phase information over several images results in diminishing response and therefore inevitably leads to smaller amplitudes and loss of information for the modified image. This reduction of the amplitude is shown for one time step in Figure 4c (left). As a general solution we therefore propose to standardize the distribution of amplitude differences:

$$\varphi_t(A_{t-1}(x,y), \hat{A}_{t-1}(x,y)) = \frac{|\hat{A}_{t-1}(x,y) - A_{t-1}(x,y)| - \mu_t}{\sigma_t} , \tag{13}$$

where $\mu_t$ represents the sample mean over all pixels, orientations and scales

$$\mu_t = \frac{1}{N} \sum_{x,y} |\hat{A}_{t-1}(x,y) - A_{t-1}(x,y)| \tag{14}$$

and $\sigma_t$ the sample standard deviation

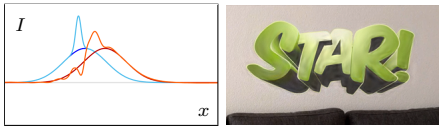
$$\sigma_t = \sqrt{\frac{1}{N-1} \sum_{x,y} (|\hat{A}_{t-1}(x,y) - A_{t-1}(x,y)| - \mu_t)^2} . \tag{15}$$

$N$ is the number of all pixels over all orientations and scales. Although the amplitude differences are technically not normal distributed (only positive values, with a peak close to 0) experiments have shown that the concluded criterion together with a threshold $\tau_\varphi$ independent of $t$

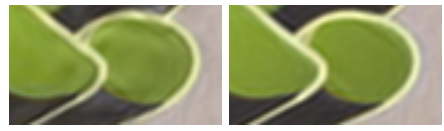
$$\varphi_t(A_{t-1}(x,y), \hat{A}_{t-1}(x,y)) > \tau_\varphi \tag{16}$$

allows for a robust identification of significant modifications.

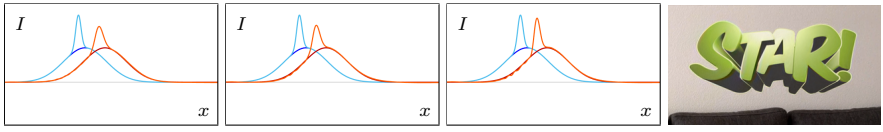
**Estimation of relevant motion.** Until now we have only compared the unmodified image $I_{t-1}$ with the modified one $\hat{I}_{t-1}$. In order to estimate how to use existing phase differences for edit propagation, we have to identify useful motion information between the two unmodified images $I_{t-1}$ and $I_t$.




(a) Propagation of edits without phase and amplitude correction introduces high frequency artifacts and decreases quality.



(b) Closeup left without, right with our proposed correction algorithm. Note the reduced low and high frequency artifacts.




(c) The one dimensional signals illustrate the three intermediate steps of our algorithm improving the result (orange) from left to right: Correcting phase difference at all locations where significant modifications has been detected. Applying correction only where necessary, i.e. no relevant motion information is available. Postprocessing the amplitude to recover details. Our final result.

Fig. 4: Improvement of our algorithm for applications which changes the frequency content. The blue and red curve correspond to the unmodified input signals at two consecutive time steps. The cyan curve corresponds to the modification on the blue input curve, and the orange curve represents our result.

As a consequence of the downsampling, any modification affects the amplitude on the lower levels. On the other hand, the low frequency levels correspond to the general, more global image motion that we are interested in. We therefore distinguish pixels with significant modifications into two cases: either the corresponding phase difference already captures the relevant motion or not. Only in the second case we need to adjust the phase differences for better propagation.

For reliable motion (i.e., phase difference) estimation we require reliable phase information in both unmodified input images, which, in turn, depends on the relative strength of the respective amplitudes compared to other pyramid levels. We therefore measure pixels with relevant phase information using:

$$\varrho_t(A_{t-1}(x,y), A_t(x,y)) = \frac{min(A_{t-1}(x,y), A_t(x,y)) - \mu_t}{\sigma_t} \,, \tag{17}$$

where $\mu_t$ and $\sigma_t$ are the mean, respectively the standard deviation, of the $min(A_{t-1}(x,y), A_t(x,y))$ samples. Pixels with $\varrho_t$ larger than some threshold $\tau_\varrho$

$$\varrho_t(A_{t-1}(x,y), A_t(x,y)) > \tau_\varrho \tag{18}$$

are defined to have a relevant motion.

The combination of these two criteria, Equation 16 and 18, define where we are missing relevant information, i.e., where we have significant change in amplitude information and no reliable motion information. This allows us to

define an indicator function in which areas an adaption of the phase information is required in order to achieve phase-based edit propagation:

$$\mathbb{I}_A(x,y) = (\varphi_t > \tau_\varphi) \ \wedge \ (\varrho_t < \tau_\varrho) \ . \tag{19}$$

The thresholds are independent of $t$ and can be fixed for an image sequence.

### 4.4 Correction of Phase Differences

After having detected the locations where we are missing necessary phase difference information, we need to fill them in with values representing the required motion. Due to the change of frequency content this corresponds to inferring phase differences for frequencies which do not already exist in the input data. To approximate them we use the available information given by the pyramid decomposition. Due to the fact that the complex steerable pyramid is translation-invariant, we can assume that the frequency bands move in a similar way. In addition, we already know the relevant phase differences (Equation 18), i.e. the relevant motion of the unmodified image pair. Therefore, in our correction algorithm, we substitute missing phase differences by a reliable phase difference from the closest lower level, denoted as $k$. To propagate the chosen phase difference $\phi^k_{diff}$ to the current level, we need to multiply it with the scale factor of the pyramid $\lambda$ accordingly. At all other locations we can use the computed phase difference to translate the phase of the modified input image:

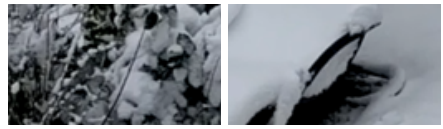
$$\hat{\phi}^l_t(x,y) = \begin{cases} \hat{\phi}^l_{t-1} - \lambda^{k-l}\phi^k_{diff} & \text{if } \mathbb{I}_A(x,y) = 1, \\ \hat{\phi}^l_{t-1} - \phi^l_{diff} & \text{otherwise.} \end{cases} \tag{20}$$

### 4.5 Correction of Amplitudes

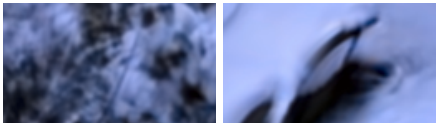
Although the above algorithm improves the results there is still the problem of the diminishing response in the amplitude, see Figure 4c, which manifests in images as increasing blur. One reason is the propagation of phase information from pyramid levels with lower resolution, which can result in a loss of sharpness of details. Secondly we assume that the motion is only captured in the phase, and the amplitude remains the same. The resulting artifacts such as ringing and blurriness are mainly visible at high frequency details such as edges. As we want to avoid the computation of any explicit correspondences which would allow to move the amplitude, we propose the following algorithm to recover some of the details by using only per pixel modifications.

By comparing the decomposition of the newly synthesized modified image $\hat{I}_t$ with the unmodified image $I_t$ at the same time step we can detect how much the amplitudes have changed due to the modification. The idea is to increase the amplitude of the modified image where necessary using a specific transfer function. At locations where the new amplitude is large, it is probably not as large as it should be. Because a linear transfer function unnecessarily enhances small amplitudes, we propose a sigmoid function. To get an estimation on how

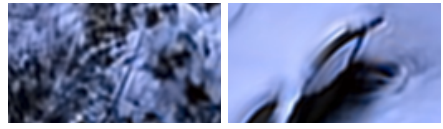
(a) Input and modification, $t = 0$



(b) Input detail, $t = 20$ (unmodified)



(c) Blur without amplitude correction



(d) With correction of amplitudes

Fig. 5: Processing the amplitude helps to recover some of the details and reduces blur, but can also magnify artifacts such as ringing. In order to increase the visibility of the effect, the modification has been propagated over $t = 20$ frames.

much energy in terms of the amplitudes has been lost, we use the amplitude of the previous modified image, i.e. $\max(\hat{A}_{t-1})/\max(\hat{A}_t)$. The proposed transfer function $\eta(\hat{A}_t(x,y))$ maps the input range of $[0, \max(\hat{A}_t)]$ to the range of the magnification factor $\left[1, \frac{\max(\hat{A}_{t-1})}{\max(\hat{A}_t)}\right]$:

$$\eta(\hat{A}_t(x,y)) = \frac{\max\left(\frac{\max(\hat{A}_{t-1})}{\max(\hat{A}_t)} - 1, 0\right)}{1 + \left(\frac{\hat{A}_t(x,y)}{\alpha \max(\hat{A}_t) + \epsilon}\right)^{-\beta}} + 1\,, \tag{21}$$

where $\beta$ defines the steepness of the curve, $\alpha \max(\hat{A}_t)$ the midpoint of the transition and $\epsilon = 10^{-4}$ is used to avoid division by 0. The maximal amplitude values are computed for each oriented frequency band separately. The advantages of this approach are demonstrated in Figure 5.

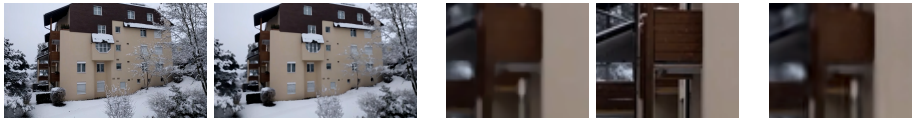
## 5 Results

We demonstrate the flexibility of our method by applying it to various kinds of video editing operations, ranging from appearance changes to detailed edits including adding new image content. More detailed results can be found in the supplementary video on our project webpage.

**Edit image appearance.** The appearance of an image can be modified either locally or globally by changing its color or applying a filter. Modifications which only change the color of an image are easier to propagate as they do not change the frequency content significantly. In these cases we do not have to correct the phase difference to adapt for changing frequencies. But our proposed correction of the amplitude is still a necessary step for the quality of the results as shown in Figure 5. Figure 6 shows the result of a local recolorization, while Figure 7 shows the propagation result of applying an artistic filter operation.

(a) Input and modification, $t = 0$ (b) Input detail, $t = 0$ (modified) and $t = 10$ (unmodified) (c) Our result, $t = 10$

Fig. 6: Propagation result for local recolorization. A longer sequence of this example can be found in the supplementary video.



(a) Input and modification, $t = 0$ (b) Input detail, $t = 0$ (modified) and $t = 5$ (unmodified) (c) Our result, $t = 5$

Fig. 7: Propagation result for applying a filter to the first image. In this case an artistic filter has been used, which adds an artistic blur to the image.

**Adding image content.** Due to our detection and correction algorithm we can also propagate image edits which significantly change the frequency content, see Figure 8. Furthermore, our method is especially suitable to handle edits on homogeneous, textureless surfaces, see Figure 9.

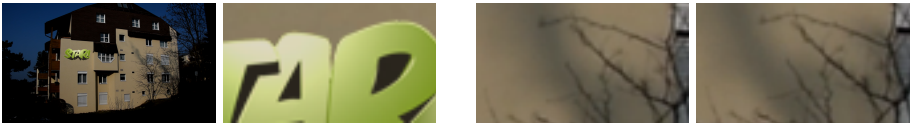
As the filter responses used in our method have local support, the method can also be applied to scenes with additional moving objects. Figure 9 shows such an example where a moving action has been captured while the hand-held camera was subject to small motion. Because in this case the motion between any frame and the first frame is sufficient small, we can apply our algorithm to process the current frame relatively to the first one. This avoids potential artifacts due to incremental propagation.

Furthermore, as the modifications only happen locally in a more or less static area, they can be localized easily, e.g. by using an approximate bounding box to the difference image of the unmodified/modified image pair. The propagation then only has to be applied to this area while the rest of the image can be substituted by the corresponding original frame of the input video sequence. This avoids potential artifacts in unmodified areas.

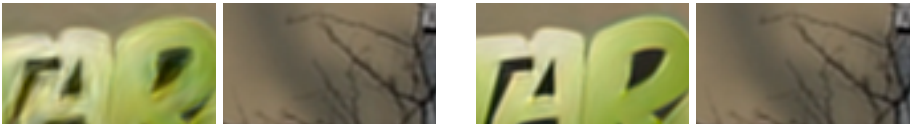
**Qualitative comparisons.** The advantage of our method is that it is applicable to general modifications independent on whether they contain local or global modifications of the input image. The methods mentioned in the related work section are optimized for specific use cases. In order to compare our results visually to correspondence based methods we use a general approach consisting of computing the optical flow field [4] and using it to warp the modified image. Figure 8 shows that we obtain visually similar results. Because optical

(a) Input, $t = 0$ (b) Optical flow [4] (c) Our result

Fig. 8: Propagation result ($t = 10$) for adding image content on a wall and comparison to using optical flow based propagation.



(a) Input, $t = 0$ (b) Moving shadow, $t = 0$ and $t = 8$

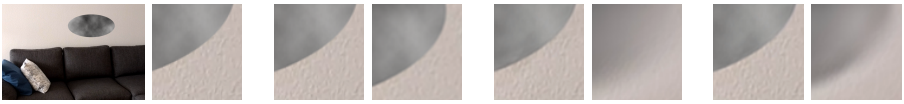


(c) Without our correction, $t = 8$ (d) Our result, $t = 8$

Fig. 9: Propagation result for a video with small camera motion while there is an additional motion happening, in this case a moving shadow on the wall. Due to the locality of the filter responses the camera motion and the additional motion can be processed separately.

flow based approaches use explicit matching they introduce less blur and can naturally handle longer propagation sequences better, see Figure 10.

**Implementation details.** Our proposed phase-based approach has a few parameters. One set for controlling the pyramid decomposition, the other for the described phase and amplitude correction algorithm. The parameters for the pyramid decomposition are a tradeoff between separability and localization. Smaller frequency bands are better for separation but have a larger spatial support. Regarding the correction algorithm, experiments have shown that we obtain favorable results for a wide set of parameter choices. For the results in this paper we have used a fixed set of values: For constructing the pyramid we used $\#_\theta = 8$ number of orientations, a scale factor $\lambda = 1.2$, and the number of levels is determined such that the coarsest level has a minimal dimension of 10 pixels. For the correction of the phase difference we have used $\tau_\varphi = \tau_\varrho = 3$. The function for correcting the amplitude has been defined with $\beta = 8$ and $\alpha = 0.1$. Similar to previous phase-based methods, frequency content which has not been captured in the pyramid levels and is summarized in real valued high- and low-pass residual needs to be treated specially. As we have no motion information available for these two residuals, we just use as an approximation the low-pass residual of the modified image $\hat{I}_{t-1}$ and ignore the high-pass residual. As the high-pass residual



(a) Input, $t = 0$ (b) Optical flow [4] (c) Naive approach (d) Our result

Fig. 10: The input image has been modified by adding a patch of Perlin noise to a flat, textureless wall. Our correction algorithm improves the propagation results, but propagation over several frames still lead to increased blur, see comparison with optical flow based propagation. Results are shown at $t = 1$ and $t = 10$.

contains high frequency details, adding it without considering motion would potentially add artifacts. We share this open research question with previous works on phase-based methods.

**Discussion and limitations.** While our method provides a novel and efficient alternative to traditional edit propagation algorithms using optical flow and tracking, it has some limitations. The difficulties lie in propagating high frequencies. As the phase-based encoding of the motion is only an approximation we lose sharpness in each propagation step. Additionally, the blurring gets increased by our multi-scale approach, as we are using the motion of lower levels which do not contain the same level of details as the higher levels to which the information gets propagated. Furthermore, sharp edges can cause ringing artifacts. Our current transfer function, used for correcting the amplitude and recovering details, can not distinguish between correct details and these artifacts. As a result, these get incorrectly amplified as well, see Figure 5. In general, artifacts such as ringing and blurriness become more visible the further the edits get propagated resulting in a degeneration of quality. But our algorithm still enables an improved phase-based propagation, see Figure 10.

## 6 Conclusions

We presented a novel approach for correspondence-free modification transfer for video, extending the existing range of video processing operations possible using purely phase-based approaches. We believe that, in particular in the context of the steady increase in video frame rate and resolution, phase-based approaches provide an interesting and efficient alternative to traditional approaches that require explicit frame-to-frame correspondences.

The recently regained interest in phase-based methods has opened up a number of surprising applications that were believed impossible before. We think that such methods bear potential for many more interesting research and applications, and hope that our work provides a new step in such a direction. For more immediate directions of future improvements and research the discussed limitations of our method provide various opportunities.

## References


1. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. IJCV 92(1), 1–31 (2011)
2. Bie, X., Huang, H., Wang, W.: Real time edit propagation by efficient sampling. Comput. Graph. Forum 30(7), 2041–2048 (2011)
3. Bonneel, N., Tompkin, J., Sunkavalli, K., Sun, D., Paris, S., Pfister, H.: Blind video temporal consistency. ACM Trans. Graph. 34(6), 196 (2015)
4. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: ECCV. pp. 25–36 (2004)
5. Didyk, P., Sitthi-amorn, P., Freeman, W.T., Durand, F., Matusik, W.: Joint view expansion and filtering for automultiscopic 3D displays. ACM Trans. Graph. 32(6), 221 (2013)
6. Elgharib, M.A., Hefeeda, M., Durand, F., Freeman, W.T.: Video magnification in presence of large motions. In: CVPR. pp. 4119–4127 (2015)
7. Hasinoff, S.W., Jwiak, M., Durand, F., Freeman, W.T.: Search-and-replace editing for personal photo collections. In: ICCP. pp. 1–8 (2010)
8. Lang, M., Wang, O., Aydin, T.O., Smolic, A., Gross, M.H.: Practical temporal consistency for image-based graphics applications. ACM Trans. Graph. (2012)
9. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM Trans. Graph. 23(3), 689–694 (2004)
10. Li, Y., Ju, T., Hu, S.: Instant propagation of sparse edits on images and videos. Comput. Graph. Forum 29(7), 2049–2054 (2010)
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (2004)
12. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: IJCAI. pp. 674–679 (1981)
13. Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: CVPR. pp. 1410–1418 (2015)
14. Ngo, D.T., Park, S., Jorstad, A., Crivellaro, A., Yoo, C.D., Fua, P.: Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In: ICCV. pp. 2273–2281 (2015)
15. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. IJCV 40(1), 49–70 (2000)
16. Rav-acha, A., Kohli, P., Rother, C., Fitzgibbon, A.: Unwrap mosaics: A new representation for video editing. ACM Trans. Graph. 27(3), 17 (2008)
17. Shi, J., Tomasi, C.: Good features to track. In: CVPR. pp. 593–600 (1994)
18. Simoncelli, E.P., Freeman, W.T.: The steerable pyramid: a flexible architecture for multi-scale derivative computation. In: ICIP. pp. 444–447 (1995)
19. Simoncelli, E.P., Freeman, W.T., Adelson, E.H., Heeger, D.J.: Shiftable multiscale transforms. IEEE Trans. Information Theory 38(2), 587–607 (1992)
20. Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. ACM Trans. Graph. 32(4), 80 (2013)
21. Xu, K., Li, Y., Ju, T., Hu, S., Liu, T.: Efficient affinity-based edit propagation using K-D tree. ACM Trans. Graph. 28(5), 118 (2009)
22. Yatagawa, T., Yamaguchi, Y.: Temporally coherent video editing using an edit propagation matrix. Computers & Graphics 43, 1–10 (2014)
23. Yücer, K., Jacobson, A., Hornung, A., Sorkine, O.: Transfusive image manipulation. ACM Trans. Graph. 31(6), 176 (2012)
24. Zhang, Z., Liu, Y., Dai, Q.: Light field from micro-baseline image pair. In: CVPR. pp. 3800–3809 (2015)