Video Frame Interpolation and Editing with Implicit Motion Estimation

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Simone Andrea Camilla Schaub-Meyer
MSc ETH in Computer Science, ETH Zurich
born on 08.12.1989
citizen of Reinach BL, Switzerland

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Frédo Durand, co-examiner
Dr. Alexander Sorkine-Hornung, co-examiner

2018
Abstract

The amount of video data captured is steadily increasing not only in terms of quantity but also in quality due to higher spatial and temporal resolutions of cameras. This poses new challenges in processing visual data efficiently. In this thesis we focus on applications for frame interpolation and modification propagation in videos. Traditional approaches usually require some accurate pixel correspondences between the images, which is an ill-posed problem. Thus they suffer from the inherent ambiguities in correspondence estimation and are particularly sensitive to occlusion/disocclusion and changes in color or brightness. In this thesis, we present efficient and novel methods which reduce and even remove the need for computing explicit correspondences. To achieve this, we build upon recent advances in phase-based methods as well as neural networks which estimate the motion between images implicitly.

First, we present a purely phase-based method for edit propagation in videos. We propose a novel algorithm to combine and adapt the phase information of the pixels in order to propagate image edits. We evaluate the flexibility by applying it to various edit applications.

Second, we develop a data driven approach for the application of color propagation in grayscale videos. By combining appearance and semantics we are able to extend the temporal range to which colors can be propagated. The extended comparisons with recent methods show the superiority of our method.

Finally, we propose a method for frame interpolation combining the advantages of a phase-based approach with a data driven strategy. We implement a convolutional neural network which reasons on the phase-based representation of the images. As a consequence, we are able to produce visually preferable results over optical flow for challenging scenarios containing motion blur and brightness changes.

To conclude, we believe that, methods using implicit motion estimation provide an interesting and efficient alternative to traditional approaches and bear potential for many more interesting research and applications. We hope that our work provides an important step in such a direction.
Zusammenfassung


Zunächst präsentieren wir eine rein phasenbasierte Methode zur Bearbeitung von Videos. Wir entwickeln einen neuartigen Algorithmus, welcher die Phaseninformation der Pixel entsprechend der Modifikation richtig kombiniert und anpasst. Wir evaluieren die Flexibilität der Methode für verschiedene Anwendungen.

Zweitens entwickeln wir eine datenbasierte Methode für die Kolorierung von Graustufenvideos. Durch die Kombination von Struktur und Semantik können wir die Farben erfolgreich über einen längeren Zeitraum als bisher existierenden Methoden propagieren.


Abschließend glauben wir, dass Methoden, welche die Bewegung zwischen Bildern implizit abschätzen eine interessante und effiziente Alternative zu den herkömmlichen Ansätzen darstellen und Potenzial für weitere spannende Entwicklungen bieten.
Acknowledgments

First of all, I would like to thank my advisor Prof. Markus Gross who offered me to pursue a Ph.D. at the Computer Graphics Laboratory. Together with the collaboration with Disney Research Zurich he gave me a great opportunity to work in an exciting environment. His support, trust and the freedom to investigate new research fields were invaluable during my Ph.D.

Furthermore, I would like to thank my close collaborators for their support, advice and fruitful discussions. Special thanks goes to Dr. Alexander Sorkine-Hornung, with whom I was lucky to work with at the beginning of my Ph.D. During that time I was able to learn a lot from his experience and guidance and he had a substantial impact on setting the foundation for my continuing research path. I am very grateful for all the support of Dr. Abdelaziz Djelouah, who joined our group during my Ph.D. I am happily looking back to many exciting discussions with him that challenged my ideas and pushed the projects forward. He was always there and offered his help till the last minute of a deadline. I am sincerely thankful to Dr. Brian McWilliams for his enduring support and help in applying machine learning concepts to our projects. I am also very thankful to Prof. Frédéric Durand, who accepted to be a member of my examination committee. It was an honor to have him in the committee, especially as this thesis is inspired by some of his interesting papers. Finally, I would like to thank my additional collaborators, Dr. Christopher Schroers, Victor Cornillère and Cyrill Hedinger, who contributed to the work presented in this thesis.

I would also like to thank all current and former members of Disney and the Institute for Visual Computing at ETH for making it such an inspiring and fun place to work. I will miss our refreshing coffee breaks.

I am deeply thankful to my friends and family, particularly to my parents, who supported me from the beginning. Last, I am most thankful to my husband Christian for his unconditional love, support and understanding over all these years. Without you, this work would not have been possible.

This work was supported by ETH Research Grant ETH-12 17-1.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Zusammenfassung</td>
<td>v</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>vii</td>
</tr>
<tr>
<td>Contents</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.2 Publications</td>
<td>7</td>
</tr>
<tr>
<td>Related Work</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Motion Representation</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Motion Estimation</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1 Explicit Methods</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 Implicit Methods</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Image and Video Synthesis</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1 Neural Networks</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Frame Interpolation</td>
<td>15</td>
</tr>
<tr>
<td>2.3.3 Video Editing</td>
<td>17</td>
</tr>
<tr>
<td>Phase-based Motion Representation</td>
<td>21</td>
</tr>
<tr>
<td>3.1 1D Case</td>
<td>21</td>
</tr>
<tr>
<td>3.2 2D Generalization</td>
<td>22</td>
</tr>
<tr>
<td>Phase-based Modification Propagation</td>
<td>25</td>
</tr>
<tr>
<td>4.1 Challenges</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Algorithm</td>
<td>27</td>
</tr>
<tr>
<td>4.2.1 Overview</td>
<td>27</td>
</tr>
<tr>
<td>4.2.2 Detecting Missing Phase Information</td>
<td>30</td>
</tr>
</tbody>
</table>
## Contents

4.2.3 Correction of Phase Differences ........................................ 32  
4.2.4 Correction of Amplitudes ............................................. 33  
4.3 Results ........................................................................... 35  
4.4 Discussion and Limitations .................................................. 39  

### Neural Network for Color Propagation  
5.1 Overview ........................................................................ 43  
5.2 Approach ........................................................................ 44  
5.2.1 Local Color Propagation .................................................. 45  
5.2.2 Global Color Transfer ..................................................... 46  
5.2.3 Fusion and Refinement Network ....................................... 47  
5.2.4 Training ....................................................................... 48  
5.3 Results ............................................................................ 50  
5.4 Discussion and Limitations .................................................. 56  

### Phase-based Frame Interpolation  
6.1 Challenges ........................................................................ 59  
6.2 PhaseNet .......................................................................... 61  
6.2.1 Learning Phase-based Interpolation .................................... 62  
6.2.2 Network Architecture ..................................................... 64  
6.2.3 Image Reconstruction ..................................................... 67  
6.2.4 Training and Implementation Details ............................... 67  
6.3 Results ............................................................................ 69  
6.4 Discussion and Limitations .................................................. 78  

### Conclusion  
7.1 Review of Principal Contributions ....................................... 81  
7.2 Future Work ..................................................................... 82  

### References  

| 85 |
# List of Figures

1.1 Image colorization and propagation of heritage footage ............................ 2
1.2 Issues of existing methods for video frame interpolation .......................... 3
1.3 Examples for edit propagation .................................................................. 4

2.1 Conceptual difference between Lagrangian and Eulerian motion representation .................................................. 10

3.1 Motion as phase shift - The translation of a simple sinusoidal function can be expressed by the phase difference ................................................................. 22
3.2 Visualization of steerable pyramid decomposition ........................................ 23

4.1 Modification transfer of a simple sinusoidal input function .......................... 26
4.2 Modification transfer of less trivial modifications ....................................... 26
4.3 Illustration of the general procedure .......................................................... 29
4.4 Improvement of our algorithm for applications which change the frequency content ............................................................................................................. 32
4.5 Illustration of the intermediate steps of our algorithm with a one dimensional signal ........................................................................................................ 33
4.6 Effect of processing the amplitude ............................................................... 34
4.7 Propagation result for local recolorization .................................................. 36
4.8 Propagation result for applying a filter ....................................................... 36
4.9 Propagation result for adding image content ............................................. 37
4.10 Propagation result for adding content to a video with two different motion patterns ........................................................................................................ 37
4.11 Limitations of our algorithm ..................................................................... 39

5.1 Phase-based color propagation ................................................................... 42
5.2 Color propagation after 30 frames .............................................................. 42
5.3 Concept of our color propagation method .................................................. 43
5.4 Overview network architecture for color propagation in videos ................. 44
5.5 Local color propagation network .................................................................. 45
5.6 Global color transfer network ..................................................................... 46
5.7 Fusion and refinement network .................................................................... 48
5.8 Ablation study - Effect of local and global color propagation strategy ........ 51
5.9 Comparison with image color propagation methods .................................... 52
List of Figures

5.10 Comparison with video color propagation methods .................. 52
5.11 Comparison with photo-realistic style transfer .................... 53
5.12 Temporal evaluation - Average PSNR error per frame over time . 55
5.13 Temporal evaluation of individual example sequences ............ 55
5.14 Comparison with Video PropNet .................................. 56
5.15 Example images of selected sequences for temporal evaluation. . 56
5.16 Propagation result for other applications .......................... 57
5.17 Yuv-decomposition .................................................. 57
5.18 Matching result for style transfer ................................. 58

6.1 Video frame interpolation .......................................... 60
6.2 Interpolation as phase shift ........................................ 61
6.3 PhaseNet overview and architecture ................................ 62
6.4 Structure of a PhaseNet block ..................................... 66
6.5 Visualization of hierarchical training procedure ................... 68
6.6 Advantage of phase loss ............................................ 70
6.7 Effect of GAN loss .................................................. 70
6.8 Results at test time for high resolutions ........................... 71
6.9 Visual comparisons with frame interpolation methods on challeng- 
ing scenarios with \textit{motion blur} ................................. 72
6.10 Visual comparison with frame interpolation methods on challeng- 
ing scenarios with \textit{brightness change} ......................... 73
6.11 Advantage of a data driven approach ............................. 73
6.12 Error measurements of different methods for different sequences 
by computing SSIM .................................................. 74
6.13 Comparison of interpolation results to the ground truth including 
a difference map .................................................... 74
6.14 Example images from the sequences for the error measurements .. 75
6.15 Processing images patch by patch ............................... 75
6.16 Processing images patch by patch with resizing .................. 76
6.17 Large motion with patches ....................................... 78
List of Tables

5.1 Statistics of the training dataset for video color propagation . . . . . 49
5.2 Quantitative evaluation - Average PSNR error over the first N frames 54

6.1 Details of the PhaseNet architecture . . . . . . . . . . . . . . . . . 65
6.2 Statistics of the training dataset for PhaseNet . . . . . . . . . . . 69
6.3 Quantitative comparison of different variations of PhaseNet . . . . 76
6.4 Quantitative comparison of improved PhaseNet with other inter-
    polation methods. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
List of Tables
CHAPTER 1

Introduction

Digital images and videos are omnipresent in our world, and have become our central means for analysis, documentation, education, and entertainment in virtually all areas of our modern society. The amount of video data captured, distributed and consumed daily is steadily increasing. Thus, there is an enormous interest in efficient techniques to analyze, modify, enhance and distribute captured data. The video data is not only increasing in quantity but also in quality due to higher spatial and temporal resolutions of cameras. This leads to challenges in processing the video frames after they have been captured in terms of memory and computation time. Many applications in video processing include synthesizing new images based on the existing frames. Specific applications include frame interpolation, e.g., for frame rate conversion and generating slow-motion effects, view interpolation and extrapolation, e.g., for light fields and multi-view displays, and video editing, e.g., changing appearance and adding image content.

In this thesis we focus on image synthesis methods for frame interpolation and modification transfer in videos. For example in movie making workflow, color modification for artistic purpose \(^1\) plays an important role. It is also used in the restoration and colorization of heritage footage for more engaging experiences, see Figure 1.1. Propagating the modifications in time automatically would significantly reduce the manual work. Another artistic tool used in videos is the generation of slow motion effects. Instead of using expensive equipment which can capture a scene at high frame rate, digital frame interpolation would allow for a cheaper option. Furthermore, frame interpolation can be use in frame rate conversion to match the various

\(^1\)See e.g. the short documentary - Painting with Pixels: O Brother, Where Art Thou?
**Introduction**

Figure 1.1: Image colorization and propagation of heritage footage for more engaging experiences. Images from the videos America in color.²

standards in television and digital cinema. Finally, the ability to faithfully interpolate frames and propagate colors in videos can have a direct impact on video compression.

Many of these applications require some form of motion estimation between the frames, which can then be used to synthesize the additional frames or edit existing ones. This motion is often obtained from an explicit correspondence mapping of pixels in consecutive frames. Common approaches are based on matching sparse feature points, or dense optical flow field estimation. However, finding a pixel-accurate mapping is an inherently ill-posed problem. Existing dense approaches usually require computationally expensive regularization and optimization. Furthermore they are particularly sensitive to occlusion/disocclusion and changes in color or brightness.

The goal of this thesis is to do basic research and to investigate innovative methods for video processing applications which reduce or even eliminate the computation of explicit correspondences between images. The focus lies on developing efficient methods which can also handle high resolution data and challenging scenarios. To achieve this, we build upon recent advances in phase-based methods as well as neural networks which estimate the motion implicitly.

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. Instead they use the assumption that small motion can be represented as the phase shift of individual pixels. Example applications include motion magnification [Wadhwa et al., 2013], view synthesis for autostereoscopic displays [Didyk et al., 2013], or frame interpolation for video [Meyer et al., 2015]. Despite their limit in motion range, the interesting advantage of such techniques over explicit methods

²www.smithsonianchannel.com/shows/america-in-color/1004516
Figure 1.2: Issues of existing methods for video frame interpolation. Inaccuracies in the computed optical flow field [Xu et al., 2012] due to change in brightness and motion blur can lead to artifacts in the synthesized intermediate frame. Also recent kernel based methods [Niklaus et al., 2017b] still have difficulties in challenging scenarios. Existing phase-based interpolation methods [Meyer et al., 2015] address these issues but are limited in their motion range. (Input image: [Li et al., 2016])

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. Furthermore, by not relying on dense correspondences, phase-based methods work better in challenging scenarios with e.g. lighting changes or motion blur.

Another direction to overcome the limitations of traditional motion estimation methods is based on recent deep learning approaches. While the continuously increasing amount of data poses a challenge in processing it efficiently, it is beneficial for training neural network models successfully [LeCun et al., 2015]. Together with the growth in computational resources, neural networks have enjoyed a recent resurgence in popularity and quickly achieved state-of-the-art performance in various applications, e.g. machine translation [Sutskever et al., 2014] and large-scale image and video classification [Krizhevsky et al., 2012]. Especially convolutional neural networks have shown to be very suitable to process images efficiently. Related to motion they have also been used to learn a mapping from input frames to an optical
flow field [Ilg et al., 2017]. Another related approach predicts kernels which implicitly represent the motion [Niklaus et al., 2017b]. Even though these methods have largely improved over traditional optical flow methods, both are still have difficulties to handle challenging scenes containing change in brightness and motion blur, see Figure 1.2.

Motivated by these recent successes of alternative approaches using implicit motion estimation, the goal of this thesis is to analyze and improve them as well as to apply them to new applications. We believe that, in particular in the context of the steady increase in video frame rate and resolution, implicit methods provide an interesting and efficient alternative to traditional approaches that require explicit frame-to-frame correspondences. Specifically we investigate in this thesis how implicit motion estimation can be used for frame interpolation and modification propagation in videos and evaluate their results.

First, we extend Chapter 4 the range of possible applications for phase-based techniques in by a method to propagate various types of image modifications over a sequence of video frames. Some possible examples of modifications are shown in Figure 1.3. The presented method does not require any 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 over time 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. Even though our results are limited to short video sequences with small displacements between frames, we believe this work represents an important step towards extending the range of applications where phase-based approaches can be used.

Second, we investigate in Chapter 5 how a deep learning method can be used for the specific example of temporal color propagation in grayscale videos. In order to propagate the color of the first frame of a sequences to the subsequent frames, our framework follows recent trends [Xue et al., 2016; Jia et al., 2016; Niklaus et al., 2017b], where motion and warping are expressed as a convolution process. Using the estimated motion the given colors of the first frame are warped frame by frame. As a result this local warping becomes less reliable with increasing distance from the reference frame. To account for that we propose a global strategy to transfer colors of the first frame based on semantics, through deep feature matching, directly to any other frame in the sequence. These two complementary approaches are combined through a fusion and refinement network to synthesize the final image. The network is trained on video sequences and our evaluation shows that our method is able to maintain better colorization results over a longer time interval compared to wide range of methods, including other deep learning based approaches.

Finally, we combine in Chapter 6 how we can leverage the advantages of a phase-based motion representation with a learning based approach. As a result, we present a novel approach, PhaseNet, for frame interpolation. Instead of learning a direct mapping from the input frames to the resulting intermediate frames using pixel colors we use the phase-based motion representation of the images as input to the network. Given this input, PhaseNet predicts the phase representation of the in-between frame. The final image is then reconstructed from these predictions. Previous phase-based methods [Meyer et al., 2015] used a hand-crafted algorithm to estimate the intermediate phase values. By replacing that part with a convolutional neural network, PhaseNet is able to handle a larger range of motion. Furthermore it also addresses the issues of previous methods in challenging scenarios, see Figure 1.2. Additionally we introduce a phase loss, which is based on the phase difference between the prediction and the ground truth and encodes motion relevant information. This is beneficial for the training procedure of the network. Altogether, we show that this allows us to outperform existing state-of-the-art methods for video frame interpolation in challenging scenarios.
Introduction

1.1 Contributions

In the following we list the main contributions of the work presented in this thesis:

- We present a novel, purely phase-based method for propagating modifications of one video frame to an entire sequence. Instead of computing accurate pixel correspondences between frames, we use the assumption that small motion is encoded in the phase shift. 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.

- We propose a deep learning framework for color propagation in videos that combines a local and global color propagation strategy. The local strategy propagates colors frame-by-frame ensuring temporal stability, and the global strategy uses semantics for color propagation within a longer range. We use a two-stage training procedure necessary to fully take advantage of both strategies. Our approach achieves state-of-the-art results as it is able to maintain better colorization results over a longer time interval compared to a wide range of methods.

- We propose a novel approach for frame interpolation, which combines the phase-based approach with a learning framework. Our presented method is designed to robustly handle challenging scenarios. Instead of predicting the color pixels our approach consists of a neural network decoder that estimates directly the phase and amplitude values of the in-between frame. The final image is reconstructed from these predictions. To efficiently train the network we introduce a phase loss, which encodes motion relevant information. We show that our method 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.
1.2 Publications

This thesis is based on the following peer-reviewed conference publications:


This thesis includes the contents of all above papers as well as additional implementation and evaluation details not present in the papers.

During the course of this thesis, the following peer-reviewed paper was published, which is not part of the thesis:

Introduction
CHAPTER 2

Related Work

This chapter reviews influential research in the area of videos synthesis with a focus on frame interpolation and video editing. Because such applications often require some notion of motion between the frames we also include a more general review on motion estimation and representation techniques.

2.1 Motion Representation

Motion representation can roughly be classified as either Lagrangian or Eulerian, illustrated in Figure 2.1. Lagrangian methods model the motion as a spatial displacement, e.g. by an optical flow field. Eulerian methods on the other hand consider the change at a fixed spatial location. This differentiation in terms of the representation helps to understand the performances of the different motion estimation methods following below.

Lagrangian. Traditionally motion is represented by a displacement field describing where the pixel moves to in the next image. This displacement field can for example be obtained by explicitly finding pixel correspondences across the images using optical flow methods, see [Sun et al., 2014] and Section 2.2.1. Some recent alternatives propose to learn to predict the optical flow field using convolutional neural networks, e.g. [Ilg et al., 2017; Sun et al., 2018].
Related Work

![Lagrangian vs. Eulerian](image)

**Figure 2.1: Lagrangian vs. Eulerian.** Motion can either be modeled as a spatial displacement (Lagrangian) or as the change at a fixed spatial location (Eulerian).

**Eulerian.** Eulerian method consider the change of color at the same pixel location over time. The most simple representation would be the difference in color value. More sophisticated algorithms consider the decomposition of the image into frequency bands using the Laplacian pyramid [Burt and Adelson, 1983] or the gradient domain [Pérez et al., 2003]. However these methods are designed to blend images along seams or within a mask. Thus non-trivial modifications would be needed to use them for other applications which require some notion of motion. [Wu et al., 2012] for example present an approach for motion magnification based on the Laplacian pyramid. [Wadhwa et al., 2013] replaces the Laplacian pyramid with a localized Fourier decomposition using complex-valued image pyramids [Simoncelli et al., 1992; Portilla and Simoncelli, 2000]. The coefficients of this decomposition allow the computation of phase values. See Chapter 3 for more details. Representing motion by the phase variations of a pixel over time is an interesting alternative to the tradition Lagrangian methods.

### 2.2 Motion Estimation

Many methods in video processing require some form of motion estimation between the images. To estimate the motion, we distinguish between two approaches: explicit and implicit motion estimation. Traditionally, motion estimation is obtained based on explicitly computed correspondences between the images. Alternatively, it is also possible to infer the motion implicitly using for example neural networks or the phase representation of pixels.
2.2 Motion Estimation

2.2.1 Explicit Methods

General approaches for explicit correspondence estimation techniques range from tracking of sparse features like SIFT [Lowe, 2004] to dense optical flow [Baker et al., 2011; Sun et al., 2014]. Because we use the motion estimation for images synthesis, the quality of the resulting synthesized images is heavily dependent on the accuracy of these correspondences.

Tracking. Tracking particular image elements is a long-studied problem [Shi and Tomasi, 1994; Lucas and Kanade, 1981] in computer vision. Related to this thesis, tracking has been used for propagating image edits in unordered image collections [Yücer et al., 2012; Hasinoff et al., 2010] as well as in videos [Rav-Acha et al., 2008]. 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. Template-based methods [Ngo et al., 2015] are more robust and also work with minimal texture and deformable surfaces. However, a template image of the object to track needs to be known in advance, which is usually not the case for general videos. Being able to track objects is also useful for video segmentation [Xiao and Lee, 2016].

The fundamental problem with sparse correspondences from tracking is the required post-processing to yield dense correspondences in the end [Dolson et al., 2010] necessary for the considered applications.

Optical flow. In the context of optical flow estimation, global energy minimization technique have been proposed that give dense correspondences [Baker et al., 2011]. However, finding a pixel-accurate mapping is an inherently ill-posed problem. Existing approaches usually require computationally expensive regularization and optimization, see [Sun et al., 2014] for a thorough analysis. Furthermore, they often rely on the brightness constancy constraint and therefore have difficulties handling scenes with large changes in brightness. Alternately, some earlier work, [Fleet and Jepson, 1990; Fleet and Jepson, 1993] suggest to use a phase constancy constraint to compute the optical flow, which is more robust with respect to smooth lighting variations. More recent methods add reliability measurements [Gautama and Hulle, 2002] and a GPU implementation [Pauwels and Hulle, 2008]. However, these methods only use phase information to match pixels in a standard optical flow formulation.
2.2.2 Implicit Methods

Convolutional Neural Networks. Instead of deriving an optical flow field from correspondences using expensive optimization techniques, neural networks can be used.

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 [Bengio et al., 2013; LeCun et al., 2015]. 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. [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016]). Most methods for these tasks are trained in a supervised manner, requiring large amounts of labelled data.

Neural networks have also been suggested to infer the motion field from data. Some of the first presented were FlowNet [Dosovitskiy et al., 2015] and FlowNet2 [Ilg et al., 2017] which learn a direct mapping from two input images to an optical flow field. They consist of one or several stacked convolutional neural networks and are trained on a large synthetic dataset. Nevertheless, they generalize well to real data and FlowNet2 approaches similar accuracy than energy minimization methods, while being much faster. However, the network itself is relatively large, consisting of over 160M parameters. SPyNet by [Ranjan and Black, 2017] is much smaller (1.2M parameters) but does not achieve the same accuracy. Another recent optical flow network, LiteFlowNet by [Hui et al., 2018], proposes feature warping at different pyramid levels instead of image warping compared to the previous methods. Concurrent to LiteFlowNet, also PWC-Net [Sun et al., 2018] has been presented, which adds a cost volume layer to its architecture. The cost volume stores the matching costs defined as the correlation between the features of the input images. Incorporating the classical principles of optical flow into the network architecture [Hui et al., 2018] has shown to improve the result while requiring less computation than FlowNet2 [Ilg et al., 2017]. These more recent methods have been presented concurrently to our work and represent a potential alternative to traditional energy-minimization optical flow methods. However, because they learn a direct mapping from the input images to the corresponding flow field, they require a large volume of ground truth optical flow data.

Effort has also been put in unsupervised optical flow methods. For example [Long et al., 2016] synthesize interpolated frames as an intermediate result. [Ren et al., 2017] uses image warping by the predicted flow and the
reconstruction error to guide the training. To improve these results, [Wang et al., 2018] compute the photometric loss only in non-occluded regions.

**Phase-based.** Instead of computing explicit correspondences, small displacements can also be inferred from the phase variations of individual pixels. This approach is motivated by the Fourier shift theorem. 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 expensive optimization problems of optical flow. Example applications using a phase-based approach include motion magnification [Wadhwa et al., 2013; Wadhwa et al., 2014; Zhang et al., 2017b], view synthesis for autostereoscopic displays [Didyk et al., 2013], light-fields [Zhang et al., 2015], image animation [Prashnani et al., 2017], action recognition [Hommos et al., 2018] and frame interpolation [Meyer et al., 2015].

The restriction of phase-based methods, is its limitation to much smaller motions between frames than, e.g., methods for sparse feature point matching. Effort has been put into extending the motion range, e.g., by combining it again with tracking or optical flow [Elgharib et al., 2015], by computing a disparity map [Zhang et al., 2015] or using additionally depth information [Kooij and van Gemert, 2016]. In this thesis we extend the range of motion by combining it with a neural network approach. While we still use the phase information obtained from the response of hand-defined filters, [Oh et al., 2018], another recent learning-based approach, suggest to learn the filters from data instead. Applied to video motion magnification, they show that they can obtain similar result while reducing the ringing artifacts, a common phenomenon of phase-based methods.

### 2.3 Image and Video Synthesis

Synthesizing new images based on existing ones often require some estimation of relation, i.e. motion, between the images. While we have discussed different options of motion estimation above we now focus on how they can be used for various videos synthesis tasks. We start with a general discussion about image and video synthesis using neural network, followed by a detailed review of the main applications of this thesis, frame interpolation and video editing.


Related Work

2.3.1 Neural Networks

Neural networks have been applied for image and video synthesis in various contexts. Related to this thesis, e.g., neural networks have been used to synthesize new views, such as in [Flynn et al., 2016] and [Kalan-tari et al., 2016] or to predict future frames as in [Mathieu et al., 2015; Villegas et al., 2017]. In terms of edit applications, neural networks have been used, e.g., to transfer the artistic style of an input image to a target image [Gatys et al., 2016], inpainting [Yang et al., 2017a] and image-to-image translation [Isola et al., 2017]. While this subsection is more a general overview, a more thorough list and discussion of methods related to the specific applications of the thesis can be found in the following subsections.

Generative models. Most models for image synthesize tasks are trained in a supervised manner, requiring large amounts of labeled data, which is extremely time-consuming and expensive to obtain. Generative models such as variational auto-encoders (VAEs) [Kingma and Welling, 2013] or generative adversarial nets (GANs) [Goodfellow et al., 2014], on the other hand, emerge as an alternative to fully supervised learning as they do not require ground truth data or labels. Instead they learn an approximation to the data generating probability distribution which can be used to sample new images. However, both methods suffer from inherent drawbacks: VAEs typically produce blurry images whereas GANs, although they generate sharper images [Radford et al., 2015], are often difficult to train. [Denton et al., 2015] propose a multi-scale approach to gradually increase the resolution of the generated images. In order to incorporate the benefits of both VAEs and GANs, [Larsen et al., 2016] propose combining both approaches resulting in a generative model which produces sharp results but is comparatively easier to train. Only recently, [Karras et al., 2018; Mescheder et al., 2018] manage to generate some convincing results on high resolution, even though only for some restrictive image classes. Another approach called PixelRNN [van den Oord et al., 2016] generates images by modeling the conditional distribution of every individual pixel given previous pixels. While PixelRNNs have a very stable training process, they are computationally expensive. In general, generative models do not yet reach the same quality as non-probabilistic methods for image synthesis.

Deterministic models. Due to the restrictions of existing generative models in terms of resolution and visual quality, deterministic non-probabilistic convolutional neural networks are still best suited for processing large im-
2.3 Image and Video Synthesis

ages. They are therefore also the approach used in this thesis. Related methods are discussed in the context of the main applications of the thesis in the corresponding subsections below.

Loss function. In general, the results of neural networks for image generation are highly dependent on the loss function. The simplest approach is to use per-pixel reconstruction loss in image space, e.g. [Dosovitskiy and Brox, 2016b]. [Johnson et al., 2016] instead define the reconstruction loss in feature space, which is believed to better represent perceptual differences.

They apply their network to applications such as style transfer as well as image super-resolution. While the perceptual loss by [Johnson et al., 2016] compares the features pixel-by-pixel, [Mechrez et al., 2018] proposes a global contextual loss measured as the similarity between the features independent of their spatial location. However, only using a loss in feature space can lead to high frequency artifacts [Mahendran and Vedaldi, 2015] in the generated images. To reduce them and to encourage spatial smoothness, a natural image prior is necessary. [Mahendran and Vedaldi, 2015] and [Johnson et al., 2016] use a total variation regularizer while [Dosovitskiy and Brox, 2016a] use an adversarial loss similar to the GAN [Goodfellow et al., 2014]. A different approach to capture the perceptual similarity of images is to use the structural similarity metric (SSIM) by [Wang et al., 2004] such as in [Ridgeway et al., 2015].

2.3.2 Frame Interpolation

Intermediate frames of a video sequence are commonly obtained by first computing correspondences (mostly leveraging optical flow or stereo methods), followed by correspondence-based image warping [Baker et al., 2011; Werlberger et al., 2011; Yu et al., 2013]. But these methods usually suffer from the inherent ambiguities in estimating the correspondences and are particularly sensitive to occlusion/disocclusion and changes in color or lighting. Alternatively to optical flow, [Mahajan et al., 2009] propose to move pixel gradients to the in-between time step and reconstruct the intermediate images by solving a Poisson equation. While addressing some problems of optical flow, this method still requires expensive global optimization.

Recently, a pure phase-based interpolation method was proposed by [Meyer et al., 2015]. By using only per-pixel modifications and not computing explicit correspondences, such an approach is more stable to lighting changes. Its main drawback is the limited range of motion it can interpolate. [Meyer et al., 2015] propose a hierarchical coarse to fine correction scheme based
Related Work

on heuristics. Concurrent to our work, [Zhou et al., 2018] propose to not only considering the phase information as in [Meyer et al., 2015], but also to shift the amplitude accordingly to improve the quality of interpolated depth frames.

Neural network have also been applied for image synthesis in various contexts, see also Section 2.3.1. Directly predicting images, however, often produce blurry results [Goodfellow et al., 2014; Vondrick et al., 2016; Xue et al., 2016]. Instead of predicting pixel value, [Zhou et al., 2016] predict an appearance flow and use it to warp pixels and synthesize novel viewpoints. In the same spirit, [Liu et al., 2017] 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., 2017a; Niklaus et al., 2017b] combine motion estimation and image synthesis into a single convolution step. The intermediate image is generated as a convolution between the input image patches and a predicted kernel. As a result, the motion which can be handled is limited by the kernel size. To handle larger displacements, [Reda et al., 2018] propose to combine optical flow and a kernel-based approach for video prediction. These methods generally result in sharp images and already better handle challenging situations—such as brightness changes—than traditional optical flow methods.

Frame interpolation is still an open research area. Concurrent to our work, some other interesting approaches have been presented. One of them is the work by [Niklaus and Liu, 2018]. Instead of blending directly the warped images they use a neural network to synthesis the final intermediate image. Additionally they also feed into the network contextual information consisting of pre-extracted feature maps. This helps to correct for some warping errors due to inaccurate optical flow estimation and improves interpolation results in case of larger motion, occlusions, and motion blur. Also [Xue et al., 2017] observes that using directly the estimated optical flow is not optimal. Instead they propose to learn a task-oriented flow representation, guided by the end application.

Limitation shared by many learning based methods for frame interpolation is their restriction on single-frame interpolation. Several intermediate frames are obtain by applying the models recursively. A method for variable-length mulit-frame interpolation is presented in [Jiang et al., 2018]. They also model the motion interpretation and occlusion reasoning in a single step.
2.3 Image and Video Synthesis

2.3.3 Video Editing

Extending image editing techniques, such as for example colorization [Zhang et al., 2017a] or style transfer [Gatys et al., 2016], to video is non-trivial due to additional requirements in terms of data, memory and computation. One approach to extend image based methods to video is to apply it to each video frame separately and combine it with some methods for temporal stability. Alternative approach to video editing is to only modify the first frame of a sequence and propagate the modifications directly to the remaining frames of the sequences. While we consider here both directions, the methods presented in this thesis address the modification propagation approach.

Temporal consistency. Applying image-based method independently to the frames of a video sequence often leads to temporal flickering. A general approach to ensure temporal consistency for various applications including optical flow and colorization has been proposed by [Lang et al., 2012]. Instead of optimizing directly for temporal consistency, [Bonneel et al., 2015] propose a general method to restore temporal consistency after a filter operation has been applied to each frame of a video independently. While producing temporally stable results, their algorithm still depends on the quality of dense correspondences from e.g. optical flow or PatchMatch [Barnes et al., 2009] and may fail in case of severe occlusion. The extension by [Yao et al., 2017] accounts for occlusion by selecting a set of key-frames resulting in increased computational cost. Besides the dense correspondences, both methods assume that the used filter does not generate new content uncorrelated to its input. Recently, also a learning based method have been proposed [Lai et al., 2018] which can handle various applications including for example stylization. In addition, it does not require explicit dense correspondences between the frames at test time, enabling processing of the frames at real-time.

Image and video colorization. Besides the general approaches presented above there exist also task-specific approaches for video editing. Related to our work is appearance editing like color manipulation and colorization. In these methods, sparse user edits get propagated spatially and temporally in case of videos, usually by solving optimization problems proportional to resolution and number of frames. A traditional approach to image colorization is to propagate colors or transformation parameters from user scribbles to unknown regions. Seminal works in this direction considered low level
affinities based on spatial and intensity distance [Levin et al., 2004]. To re-
duce user interaction, many directions have been considered such as design-
ing better similarities [Luan et al., 2007]. Other approaches to improve edit
propagation include embedding learning [Chen et al., 2012], iterative fea-
ture discrimination [Xu et al., 2013] or dictionary learning [Chen et al., 2014].
Achieving convincing results for automatic image colorization [Cheng et al.,
2015; Iizuka et al., 2016], deep convolutional networks have also been con-
sidered for edit propagation [Endo et al., 2016] and interactive image col-
orization [Zhang et al., 2017a].

To extend edit propagation to videos, computational efficiency is critical and
various strategies have been investigated [An and Pellacini, 2008; Xu et al.,
2009; Yatagawa and Yamaguchi, 2014].

One of the first method considering grayscale video colorization was pro-
posed by [Welsh et al., 2002] as a frame-to-frame color propagation. Later,
image patch comparisons [Sýkora et al., 2004] were used to handle large dis-
placements and rotations. However this method targets cartoon content and
is not directly adaptable to natural videos. [Yatziv and Sapiro, 2006] con-
sider geodesic distance in the 3d spatio-temporal volume to color pixels in
videos and [Sheng et al., 2011] replace spatial distance by a distance based
on Gabor features. The notion of reliability and priority [Heu et al., 2009] for
coloring pixels allow better color propagation. These notions are extended
to entire frames [Xia et al., 2016], considering several of them as sources for
coloring next gray images. For increased robustness, [Pierre et al., 2017] use
a variational model that rely on temporal correspondence maps estimated
through patch matching and optical flow estimation. While these methods
often produce satisfactory result they are inherently slow due to the compu-
tation of the dense correspondences and therefore not suited for long, high
resolution video sequences.

Instead of using pixel correspondences, some recent methods have proposed
alternative approaches to the video colorization problem. [Paul et al., 2017]
uses instead of motion vectors the dominant orientations of a 3D steerable
pyramid decomposition as guidance for the color propagation of user scrib-
bles. [Jampani et al., 2017], on the other hand, use a temporal bilateral net-
work for dense and video adaptive filtering, followed by a spatial network
to refine features.

**Color and style transfer.** Video editing including colorization can also be
seen as transferring the color or style of the first frame to the rest of the im-
egages in the sequence. We only outline the main directions of color transfer
2.3 Image and Video Synthesis

as an extensive review of these methods is available in [Faridul et al., 2016]. Many methods rely on histogram matching [Reinhard et al., 2001] which can achieve surprisingly good results given their relative simplicity but colors could be transferred between incoherent regions. Taking segmentation into account can help to improve this aspect [Tai et al., 2005]. Color transfer between videos is also possible [Bonneel et al., 2013] by segmenting the images using luminance and transferring chrominance. Recently [Arbelot et al., 2016] proposed an edge-aware texture descriptor to guide the colorization. Other works focus on more complex transformations such as changing the time of the day in photographs [Shih et al., 2013], artistic edits [Shih et al., 2014] or season change [Okura et al., 2015].

Since the seminal work of [Gatys et al., 2016], various methods based on neural networks have been proposed for style transfer [Li and Wand, 2016; Johnson et al., 2016; Li et al., 2017]. In order to extend them to video [Huang et al., 2017; Gupta et al., 2017] embed a temporal consistency loss in the training inspired by [Ruder et al., 2016].

Several recent works have targeted photo-realistic style transfer [Mechrez et al., 2017; Luan et al., 2017; Li et al., 2018; He et al., 2017]. [Mechrez et al., 2017] rely on Screened Poisson Equation to maintain the fidelity with the style image while constraining the results to have gradients similar to the content image. In [Luan et al., 2017] photo-realism is maintained by constraining the image transformation to be locally affine in color space. This is achieved by adding a corresponding loss to the original neural style transfer formulation [Gatys et al., 2015]. To avoid the resulting slow optimization process, [He et al., 2017] use patch matching on VGG features to obtain a guidance image. Finally, [Li et al., 2018] proposed a two stage architecture where an initial stylized image, estimated through whitening and coloring transform (WCT) [Li et al., 2017], is refined with a smoothing step.
Phase-based methods use the intuition that the motion of certain signals or functions can be represented as a shift of their phase [Wadhwa et al., 2013]. In this section we first explain the basic mathematical justification for the 1D case as well as the generalization to images.

### 3.1 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 $\omega$:

$$f(x) = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\omega x} = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\phi_\omega}, \quad (3.1)$$

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:

$$f(x - \delta(t)) = \sum_{\omega=-\infty}^{\omega=+\infty} A_\omega e^{i\omega(x - \delta(t))}. \quad (3.2)$$

The phase difference between the original and the shifted function

$$\phi_{\text{diff}}^\omega = \omega x - \omega(x - \delta(t)) = \omega \delta(t) \quad (3.3)$$
Phase-based Motion Representation

![Figure 3.1: Motion as phase shift. The translation of a simple sinusoidal function (blue to green) can be expressed by the phase difference, here $\pi/3$.](image)

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:

$$\delta(t) = \frac{\phi^{\omega}_{\text{diff}}}{\omega}. \quad (3.4)$$

To illustrate this concept we use simple one dimensional sinusoidal functions $y = A \sin(\omega x - \phi)$, where $A$ is the amplitude, $\omega$ the angular frequency and $\phi$ 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 3.1. 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 video applications, these two curves (blue and green) would correspond to the input images.

### 3.2 2D Generalization

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 [Portilla and Simoncelli, 2000; Simoncelli and Freeman, 1995; Simoncelli et al., 1992]. By applying the steerable pyramid filters $\Psi_{\omega,\theta}$ resembling Gabor filters, we can decompose an image into a set of scale and
3.2 2D Generalization

Figure 3.2: Visualization of decomposition. Shows filters implemented in frequency domain for two orientations and two scales as well as the responses obtained in spatial domain by applying them to the discrete Fourier transform of an image from the Middlebury dataset [Baker et al., 2011].

Orientation depended complex-valued subbands $R_{\omega,\theta}(x,y)$:

\[ R_{\omega,\theta}(x,y) = (I \ast \Psi_{\omega,\theta})(x,y) \]  
\[ = C_{\omega,\theta}(x,y) + i S_{\omega,\theta}(x,y) \]  
\[ = A_{\omega,\theta}(x,y) e^{i \phi_{\omega,\theta}(x,y)} \]  

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

\[ A_{\omega,\theta}(x,y) = \sqrt{C_{\omega,\theta}(x,y)^2 + S_{\omega,\theta}(x,y)^2} = |R_{\omega,\theta}(x,y)| \]
and the phase values

\[ \phi_{\omega, \theta}(x, y) = \arctan \left( \frac{S_{\omega, \theta}(x, y)}{C_{\omega, \theta}(x, y)} \right) = \text{Im} \left( \log \left( R_{\omega, \theta}(x, y) \right) \right), \]  

(3.9)

where \text{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. Figure 3.2 visualizes the filters in frequency domain as well the responses in spatial domain for an example image.

To summarize, 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. 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 the algorithms presented in Chapter 4 and Chapter 6.
Chapter 4

Phase-based Modification Propagation

In this chapter, 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.

4.1 Challenges

For propagating modifications we are interested in using the 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 4.1a, where as an input we are given a function in the form of $f(x) = A \sin(\omega x - \phi)$ (blue). In our tar-
Phase-based Modification Propagation

**Figure 4.1:** 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 4.1) (orange, right).

**Figure 4.2:** 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).

generated 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 4.1b, orange) by subtracting the phase difference:

\[
\hat{f}(x) = \hat{A} \sin(\omega x - \phi_{\text{diff}}) = \hat{A} \sin(\omega(x - \phi_{\text{shift}})),
\]

where \( \phi_{\text{diff}} \) represents the frequency-dependent version of the actual spatial displacement \( \phi_{\text{shift}} \), see also Chapter 3 for details.
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 4.2a 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.

### 4.2 Algorithm

#### 4.2.1 Overview

Given the steerable pyramid 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 $\hat{A}_{\omega,\theta}^t$ and $\hat{\phi}_{\omega,\theta}^t$ based on the available information. The resulting image is then obtained by integrating the modified responses according to Equation 3.1. 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 $\phi_{\text{diff}}$ between the phases of the unmodified images $I_{t-1}$ and $I_t$, i.e.,

$$\phi_{\text{diff}}^{t-1,t} = \text{atan2}(\sin(\phi_{t-1} - \phi_t), \cos(\phi_{t-1} - \phi_t)),$$

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

27
Phase-based Modification Propagation

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:

\[
\hat{\phi}_t = \hat{\phi}_{t-1} - \phi_{\text{diff},t},
\]

\[
\hat{A}_t = \hat{A}_{t-1}.
\]

However, as explained in Section 4.1, and illustrated in Figure 4.1 and 4.2, 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 4.3 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 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.
4.2 Algorithm

Figure 4.3: 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 $\phi_i$.

\[
(P_0, P_1, \hat{P}_0) \leftarrow \text{decompose}(I_0, I_1, \hat{I}_0) \quad \triangleright \quad \text{[Portilla and Simoncelli, 2000]}
\]

\[
\phi_{\text{diff}} \leftarrow \text{phaseDifference}(\phi_0, \phi_1) \quad \triangleright \quad \text{Eq. 4.2}
\]

\[
(A_0, A_1, \hat{A}_0) \leftarrow \text{normalize}(A_0, A_1, \hat{A}_0) \quad \triangleright \quad \text{Eq. 4.5}
\]

\[
\phi_1 \leftarrow \text{significantMod}(A_0, \hat{A}_0) \quad \triangleright \quad \text{Eq. 4.5}
\]

\[
\phi_1 \leftarrow \text{relevantMotion}(A_0, A_1) \quad \triangleright \quad \text{Eq. 4.9}
\]

\[\text{for all } l = L - 1 : 1 \text{ do} \]

\[
\hat{\phi}_1^l \leftarrow \text{compute}(\hat{\phi}_0, \phi_{\text{diff}}, \phi_1, \phi_1) \quad \triangleright \quad \text{Eq. 4.12}
\]

\[\text{end for}\]

\[
\hat{I}_1 \leftarrow \text{reconstruct}(\hat{A}_0, \phi_1) \quad \triangleright \quad \text{See [Portilla and Simoncelli, 2000]}
\]

\[
\hat{P}_1 \leftarrow \text{decompose}(\hat{I}_1) \quad \triangleright \quad \text{Sec. 4.2.4}
\]

\[
\hat{\hat{A}}_1 \leftarrow \text{correctAmpl}(\hat{A}_0, \hat{A}_1, \phi_1) \quad \triangleright \quad \text{[Portilla and Simoncelli, 2000]}
\]

\[
\hat{I}_1 \leftarrow \text{reconstruct}(\hat{P}_1)
\]
4.2.2 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 post-processing (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. $|\hat{A}_{t-1}(x, y) - A_{t-1}(x, y)| > \theta$ 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
4.2 Algorithm

for one time step in Figure 4.5 (second plot). As a general solution we therefore propose to standardize the distribution of amplitude differences:

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

(4.5)

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

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

(4.6)

and \( \sigma_t \) the sample standard deviation

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

(4.7)

\( N \) 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 \( t \)

\[ \varphi_t(A_{t-1}(x,y), \hat{A}_{t-1}(x,y)) > \tau_{\varphi} \]  

(4.8)

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 \( I_t \).

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:

\[ \varphi_t(A_{t-1}(x,y), A_t(x,y)) = \frac{\min(A_{t-1}(x,y), A_t(x,y)) - \mu_t}{\sigma_t}, \]  

(4.9)
Phase-based Modification Propagation

![Image: STAR!]  

(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.

Figure 4.4: Improvement of our algorithm for applications which change the frequency content.

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 $\varphi_t$ larger than some threshold $\tau_\varphi$

$$q_t(A_{t-1}(x,y), A_t(x,y)) > \tau_\varphi$$

are defined to have a relevant motion.

The combination of these two criteria, Equation 4.8 and 4.10, 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:

$$I_A(x,y) = (\varphi_t > \tau_\varphi) \land (\varphi_t < \tau_\varphi).$$

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

4.2.3 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 4.10), 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 $\phi_{\text{diff}}^k$ 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:

$$\hat{\phi}_l(x,y) = \begin{cases} 
\hat{\phi}_{l-1} - \lambda^{k-l} \phi_{\text{diff}}^k & \text{if } I_A(x,y) = 1, \\
\hat{\phi}_{l-1} - \phi_{\text{diff}}^l & \text{otherwise.}
\end{cases}$$

(4.12)

### 4.2.4 Correction of Amplitudes

Although the above algorithm improves the results there is still the problem of the diminishing response in the amplitude, see Figure 4.5 (second plot), 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.

Figure 4.6: 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.

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 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 $[1, \max(\hat{A}_{t-1}) / \max(\hat{A}_t)]$:

$$\eta(\hat{A}_t(x, y)) = \frac{\max\left(\frac{\max(\hat{A}_{t-1})}{\max(\hat{A}_t)} - 1, 0\right)}{1 + \left(\frac{\hat{A}_t(x, y)}{\alpha \max(\hat{A}_t) + \epsilon}\right)^{\beta}} + 1,$$  \hspace{1cm} (4.13)

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 4.6.

### 4.3 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.

**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 4.6. Figure 4.7 shows the result of a local recolorization, while Figure 4.8 shows the propagation result of applying an artistic filter operation.
Phase-based Modification Propagation

Figure 4.7: Propagation result for local recolorization.

Figure 4.8: 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.
4.3 Results

Figure 4.9: Propagation result ($t = 10$) for adding image content on a wall and comparison to using optical flow [Brox et al., 2004] based propagation.

Figure 4.10: 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.

Adding image content. Due to our detection and correction algorithm we can also propagate image edits which significantly change the frequency content, see Figure 4.9. Furthermore, our method is especially suitable to handle edits on homogeneous, textureless surfaces, see Figure 4.10.

As the filter responses used in our method have local support, the method can also be applied to scenes with additional moving objects. Figure 4.10 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 [Brox et al., 2004] and using it to warp the modified image. Figure 4.9 shows that we obtain visually similar results. Because optical flow based approaches use explicit matching they introduce less blur and can naturally handle longer propagation sequences better, see Figure 4.11.

**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 thesis 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 $\tau_\phi = \tau_\theta = 3$. 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 of the
4.4 Discussion and Limitations

Figure 4.11: 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 [Brox et al., 2004] based propagation. Results are shown at $t = 1$ and $t = 10$.

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 4.6. In general, artifacts such as ringing and blurriness become more visible the further the edits get propagated resulting in a degeneration of quality. However, our phase-based propagation algorithm represents an interesting alternative approach for modification transfer and Figure 4.11 shows our improvements over a naive phase-based approach.
Phase-based Modification Propagation
Neural Network for Color Propagation

In the previous Chapter 4 we have presented a general approach for video edit propagation using a phase-based approach. While this approach can also be used for color propagation, the results are limited to small motions and degrades quickly in quality over time as artifacts accumulate, see Figure 5.1.

Traditional approaches for color propagation in videos often use appearance descriptors, based on which colors are then propagated both spatially and temporally. These methods, however, are computationally expensive and do not take advantage of semantic information of the scene. Recently, deep learning methods have been proposed to take advantage of semantics for color propagation in images [Zhang et al., 2017a] and videos [Jampani et al., 2017]. Still, these approaches have some limitations and do not yet achieve satisfactory results on video content. In this chapter, however, we present a deep learning architecture for color propagation in videos combining warping and semantical information. We show that our method is able to maintain better colorization results over a longer time interval compared to a wide range of methods, see Figure 5.2.
Figure 5.1: Phase-based color propagation. The phase-based approach presented in Chapter 4 works for small motions \((k = 1)\) but degrades quickly for longer ranges \((k = 30)\). (Input image: [Pont-Tuset et al., 2017])

Figure 5.2: Color propagation after 30 frames \((k = 30)\). Our approach is superior to existing strategies for video color propagation: Style transfer [Li et al., 2018], SepConv [Niklaus et al., 2017b], Video PropNet [Jampani et al., 2017] and Bilateral Solver [Barron and Poole, 2016]. The closest performing competitor [Xia et al., 2016] requires computation time of several hours whereas ours result is obtained in about one minute for 30 frames.
The goal of our method is to colorize a grayscale image sequence by propagating the given color of the first frame to the following frames. Our proposed approach takes into account two complementary aspects: short range and long range color propagation, see Figure 5.3. We implement this idea using a neural network framework consisting of three parts: The local propagation, the global propagation and the fusion network, see Figure 5.4.

The objective of the short range propagation network is to propagate colors on a frame by frame basis. It takes as input two consecutive grayscale frames and estimates a warping function. This warping function is used to transfer the colors of the previous frame to the next one. Following recent trends [Xue et al., 2016; Jia et al., 2016; Niklaus et al., 2017b], warping is expressed as a convolution process. In our case we choose to use spatially adaptive kernels that account for motion and re-sampling simultaneously [Niklaus et al., 2017b], but other approaches based on optical flow could be considered as well.

For longer range propagation, simply smoothing warped colors according to the grayscale guide image is not sufficient. Semantic understanding of the scene is needed to transfer color from the first colored frame of the video to the rest of the video sequence. In our case, we find correspondences between pixels of the first frame and the rest of the video. Instead of matching pixel colors directly we incorporate semantical information by matching deep features extracted from the frames. These correspondences are then used in order to sample colors from the first frame. Besides the advantage for long range color propagation, this approach also helps to recover missing colors due to occlusion/disocclusion.

To combine the intermediate images of these two parallel stages, we use a...
Figure 5.4: **Overview.** To propagate colors in a video we use both short range and long range color propagation. First, the local color propagation network $F_w$ uses consecutive grayscale frames $G_{k-1}$ and $G_k$ to predict spatially adaptive kernels that account for motion and re-sampling from $I_{k-1}$. To globally transfer the colors from the reference frame $I_1$ to the entire video a matching based on deep image features is used. The results of these two steps, $I^w_k$ and $I^r_k$, are together with $G_k$ the input to the fusion and refinement network which estimates the final current color frame $I_k$. (Input image: [Pont-Tuset et al., 2017])

convolutional neural network. This corresponds to the fusion and refinement stage. As a result, the final colored image is estimated by taking advantage of information that is present in both intermediate images, i.e. local and global color information.

### 5.2 Approach

Let’s consider a grayscale video sequence $G = \{G_1, G_2, \ldots, G_n\}$ of $n$ frames, where the colored image $I_1$ (corresponding to $G_1$) is available. Our objective is to use the frame $I_1$ to colorize the set of grayscale frames $G$. Using a local (frame-by-frame) strategy, colors of $I_1$ can be sequentially propagated to the entire video using temporal consistency. With a global strategy, colors present in the first frame $I_1$ can be simultaneously transferred to all the frames of the video using a style transfer like approach. In this work we propose a unified solution for video colorization combining local and global strategies.
5.2 Approach

![Figure 5.5: Local color propagation. Colors are propagated to frame $I^w_k$ by a convolution of the color frame $I_{k-1}$ with kernels $K^h_k$ and $K^v_k$ predicted from the two consecutive grayscale frames $G_{k-1}$ and $G_k$. (Input image: [Pont-Tuset et al., 2017])](image)

5.2.1 Local Color Propagation

Relying on temporal consistency, our objective is to propagate colors frame by frame. Using the adaptive convolution approach developed for frame interpolation [Niklaus et al., 2017b], one can similarly write color propagation as convolution operation on the color image, see Figure 5.5: given two consecutive grayscale frames $G_{k-1}$ and $G_k$, and the color frame $I_{k-1}$, an estimate of the colored frame $I_k$ can be expressed as

$$I^w_k(x, y) = P_{k-1}(x, y) * K_k(x, y),$$

where $P_{k-1}(x, y)$ is the image patch around pixel $I_{k-1}(x, y)$ and $K_k(x, y)$ is the estimated pixel dependent convolution kernel based on $G_k$ and $G_{k-1}$. This kernel is approximated with two 1D-kernels as

$$K_k(x, y) = K^v_k(x, y) * K^h_k(x, y).$$

The convolutional neural network architecture used to predict these kernels is similar to the one originally proposed for frame interpolation [Niklaus et al., 2017b], with the difference that 2 kernels are predicted (instead of 4 in the interpolation case). Furthermore, we use a softmax layer for kernel prediction which helps to speedup training [Vogels et al., 2018]. If we note $F_w$ the prediction function, the local color propagation can be written as

$$I^w_k = F_w(G_k, G_{k-1}, I_{k-1}; \Lambda_w),$$

with $\Lambda_w$ being the set of trainable parameters.
Figure 5.6: **Global color transfer.** To transfer the colors of the first frame $I_1$, feature maps $\Phi_{G_1}$ and $\Phi_{G_k}$ are extracted from both inputs $G_1$ and $G_k$. First, a matching is estimated at low resolution. This matching performed on features from a deep layer ($l_{\text{coarse}}$) allows to consider more abstract information. It is however too coarse to directly copy corresponding image patches. Instead, we use this initial matching to restrict the search region when matching pixels using low level image statistics (from level $l_{\text{fine}}$ feature map). Here we show the region of interest (in blue) used to match the pixel in violet. All the pixels sharing the same coarse positions (in violet dotted rectangle) share the same Region Of Interest (ROI). Using the final matching, $I_1$ colors are transferred to the current grayscale image $G_k$. (Input image: [Pont-Tuset et al., 2017])

### 5.2.2 Global Color Transfer

The local propagation strategy becomes less reliable as the frame to colorize is further away from the first frame. This can be due to occlusions/disocclusions, new elements appearing in the scene or even complete change of background (due to camera panning for example). In this case, a global strategy with semantic understanding of the scene is necessary. It allows to transfer color within a longer range both temporally and spatially. To achieve this, we leverage deep feature extracted with convolutional neural networks trained for classification and image segmentation. Similar ideas have been developed for style transfer and image inpainting [Li and Wand, 2016; Yang et al., 2017b].

Formally, we note $\Phi_{I,l}$ the feature map extracted from the image $I$ at layer $l$ of a discriminatively trained deep convolutional neural network. We can estimate a pixel-wise matching between the reference frame $G_1$ and the current frame to colorize $G_k$ using their respective features maps $\Phi_{G_1,l}$ and $\Phi_{G_k,l}$. Similarity for two positions $x, x'$ is measured as:

$$S_{G_k,G_1}(x, x') = \|\Phi_{G_k,l}(x) - \Phi_{G_1,l}(x')\|_2^2. \quad (5.4)$$
Transferring the colors using pixel descriptor matching can be written as:

\[ I^s_k(x) = I_1(\arg \min_{x'} S_{G_k, G_1}(x, x')) . \]  

(5.5)

To maintain good quality for the matching, while being computationally efficient, we adopt a two stage coarse-to-fine matching. Matching is first estimated for features from a deep layer \( l = l_{\text{coarse}} \). This first matching, at lower resolution, defines a region of interest for each pixel in the second matching step of features at level \( l = l_{\text{fine}} \). The different levels \( l \) of the feature maps correspond to different abstraction level. The coarse level matching allows to consider regions that have similar semantics, whereas the fine matching step considers texture-like statistics that are more effective once a region of interest has been defined. We note \( F_s \) the global color transfer function

\[ I^s_k = F_s(G_k, I_1, G_1; \Lambda_s) , \]  

(5.6)

with \( \Lambda_s \) being the set of trainable parameters. Figure 5.6 illustrates all the steps from feature extraction to color transfer. Any neural network trained for image segmentation could be used to compute the features maps. In our case we use ResNet-101 [He et al., 2016] architecture fine tuned for semantic image segmentation [Chen et al., 2018]. For \( l_{\text{coarse}} \) we use the output of the last layer of the \textit{conv3}-block, while for \( l_{\text{fine}} \) we use the output of the first \textit{conv1}-block (but with stride 1).

### 5.2.3 Fusion and Refinement Network

The results we obtain from the local and global stages are complementary. The local color propagation result is sharp with most of the fine details preserved. Colors are mostly well estimated except at occlusion/dis-occlusion boundaries where some color bleeding can be noticed. The result obtained from the global approach is very coarse but colors can be propagated to a much larger range both temporally and spatially. Fusing these two results is learned with a fully convolutional neural network.

For any given grayscale frame \( G_k \), the local and global steps result in two estimates of the color image \( I_k \): \( I^w_k \) and \( I^s_k \). These intermediate results are leveraged by the proposed convolutional network (Figure 5.7) to predict the final output:

\[ I_k = F_f(G_k, I^w_k, I^s_k; \Lambda_f) , \]  

(5.7)

where \( F_f \) notes the prediction function and \( \Lambda_f \) the set of trainable parameters.
Neural Network for Color Propagation

**Figure 5.7: Fusion and refinement network.** The intermediate results $I_w^k$ and $I_s^k$ are together with the current grayscale image $G_k$ the input to the final fusion network. By passing it through a set of convolution layers without reducing the resolutions we get the final refined result $I_k$. (Input image: [Pont-Tuset et al., 2017])

**Architecture details.** The proposed fusion and refinement network consists of 5 convolutional layers with 64 output channels each followed by a relu-activation function. To keep the full resolution we use strides of 1 and increase the receptive field by using dilations of 1, 2, 4, 1 and 1, respectively. To project the output to the final colors we use another convolutional layer without any activation function. To improve training and the prediction we use instance normalization [Ulyanov et al., 2016] to jointly normalize the input frames. The computed statistics are then also used to renormalize the final output.

**5.2.4 Training**

Since all the layers we use are differentiable, the proposed framework is end-to-end trainable, and can be seen as predicting the colored frame $I_k$ from all the available inputs

$$I_k = \mathcal{F}(G_k, G_{k-1}, I_{k-1}, I_1; \Lambda_w, \Lambda_s, \Lambda_f).$$

The network is trained to minimize the total objective function $\mathcal{L}$ over the dataset $\mathcal{D}$ consisting of sequences of colored and grayscale images.

$$\Lambda_f^*, \Lambda_w^* = \arg\min_{\Lambda_f, \Lambda_w} \mathbb{E}_{I_1, I_2, G_1, G_2 \sim \mathcal{D}} \left[ \mathcal{L} \right].$$
### 5.2 Approach

<table>
<thead>
<tr>
<th></th>
<th>DAVIS2017</th>
<th>Youtube</th>
</tr>
</thead>
<tbody>
<tr>
<td>#video clips</td>
<td>135</td>
<td>605</td>
</tr>
<tr>
<td>#video frames</td>
<td>9385</td>
<td>19897</td>
</tr>
<tr>
<td>mean #frames per clip</td>
<td>69.5</td>
<td>32.9</td>
</tr>
<tr>
<td>resolution</td>
<td>480p</td>
<td>720p / 1080p</td>
</tr>
</tbody>
</table>

Table 5.1: *Training dataset.* Statistics of the DAVIS2017 dataset [Pont-Tuset et al., 2017] and selected video clips from Youtube we use to train our network.

**Image loss.** We use the $\ell_1$-norm of pixel differences which has been shown to lead to sharper results than $\ell_2$ [Niklaus et al., 2017b; Long et al., 2016; Mathieu et al., 2015]. This loss is computed on the final image estimate:

$$L_1 = ||I_k - \hat{I}_k||_1 .$$  \hspace{1cm} (5.10)

**Warp loss.** The local propagation part of the network has to predict the kernels used to warp the color image $I_{i-1}$. This is enforced through the warp loss. It is also computed as the $\ell_1$-norm of pixel differences between the ground truth image $I_i$ and $I^w$:

$$L_w = ||I_k - I^w_k||_1 .$$  \hspace{1cm} (5.11)

Since $I^w_k$ is an intermediate result, using more sophisticated loss functions such as feature loss [Gatys et al., 2015] or adversarial loss [Goodfellow et al., 2014] is not necessary. All the sharp details will be recovered by the fusion network.

**Training procedure.** To train the network we used pairs of frames from video sequences obtained from the DAVIS [Perazzi et al., 2016; Pont-Tuset et al., 2017] dataset and Youtube. Table 5.1 summarizes the statistics of the chosen datasets. In total we have 740 video clips and approx. 30k frames from which we randomly extract patches of $256 \times 256$.

We train the fusion network with a batch size of 16 over 12 epochs. To efficiently train the fusion network we first apply $F_w$ and $F_s$ separately to all training video sequences. The resulting images $I^w_k$ and $I^s_k$ show the limitations of their respective generators $F_w$ and $F_s$. The fusion network can then be trained to synthesize the best color image from these two intermediate results. As input we provide $G_k$ and the intermediate images $I^w_k$ and $I^s_k$ converted to Yuv-color space. Using the luminance channel helps the prediction process as it can be seen as an indicator on the accuracy of the intermediate

---

49
results. The final image consists of the chrominance values estimated by the fusion network and $G_k$ as luminance channel.

**Running time.** At test time, the matching step is the most computationally involved. Still, our naive implementation with TensorFlow computes high resolution ($1280 \times 720$) edit propagation within 5s per frame on a Titan X (Pascal).

### 5.3 Results

For our evaluation we used various types of videos. This includes videos from DAVIS [Perazzi et al., 2016; Pont-Tuset et al., 2017], using the same test set as in [Jampani et al., 2017], from [Bonneel et al., 2015] as well as from Sintel [Butler et al., 2012]. We also test our approach on HD videos from the video compression dataset [Wang et al., 2017].

**Ablation Study.** To show the importance of both the local and global strategy, we evaluate both configuration. The local strategy is more effective for temporal stability and details preservation but is sensitive to occlusion/disocclusion. Figure 5.8 shows an example where color propagation is not possible due to occlusion, and a global strategy is necessary. Using a global strategy only is not sufficient, as some details are lost during the matching step and temporal stability is not maintained. See also Figure 5.12 and 5.13 for the quantitative advantage of combining the two strategies.

**Comparison with image color propagation.** Given a partially colored image, propagating the colors to the entire image can be achieved using the bilateral space [Barron and Poole, 2016] or deep learning [Zhang et al., 2017a]. To extend these methods to video, we compute optical flow between consecutive frames [Zach et al., 2007] and use it to warp the current color image to the next frame. In detail, we compute a confidence measure for the warped colors by warping the grayscale image and taking the difference in intensities with the original gray frame. The warped colors, the confidence maps and the reference grayscale image can be used to color the second frame using the fast bilateral solver [Barron and Poole, 2016]. Using a very conservative threshold, the confidence map is binarized to indicate regions where colors should be propagated using deep priors [Zhang et al., 2017a]. These image based color methods achieve satisfactory color propagation on
5.3 Results

Figure 5.8: Ablation study. Using local color propagation based on [Niklaus et al., 2017b] only preserve details but is sensitive to occlusion/disocclusion. Using only global color transfer does not preserve details and is not temporally stable. Best result is obtained when combining both strategies. See Figure 5.12 and 5.13 for quantitative evaluation. (Input image: [Wang et al., 2017; Butler et al., 2012])

Comparison with video color propagation. Relying on optical flow to propagate colors in a video is the most common approach. In addition to this, Xie et al. [Xia et al., 2016] also consider frame re-ordering and use multiple reference frames. However, this costly process is limiting as processing 30 HD frames requires several hours. Figure 5.2 and Figure 5.10 show that we achieve similar or better quality in one minute. Phase-based representation can also be used for edit propagation in videos, see Chapter 4. This approach to color propagation is however limited by the difficulty in propagating high frequencies. Recently, video propagation networks [Jampani et al., 2017] were proposed to propagate information forward through a video. Color propagation is a natural application of such networks. Contrary to the
Figure 5.9: **Comparison with image color propagation methods.** Methods propagating colors in a single image [Zhang et al., 2017a; Barron and Poole, 2016] achieve good results on the first frame. The quality of the results degrades as the frame to colorize is further away from the reference image. (Input image: [Wang et al., 2017])

Figure 5.10: **Comparison with video color propagation methods.** Our approach best retains the sharpness and colors of this video sequence compared to our method in Chapter 4 and [Jampani et al., 2017], respectively. Our result was obtained in less than one minute while the optical flow method [Xia et al., 2016] needed 5 hours for half the original resolution. (Input image: [Wang et al., 2017])
fast bilateral solver [Barron and Poole, 2016] that only operates on the bilateral grid, video propagation networks [Jampani et al., 2017] benefits from a spatial refinement module and achieve sharper and better results. Still, by relying on standard bilateral features (i.e. colors, position, time) colors can be mixed and propagated from incorrect regions, which leads to the global impression of washed out colors.

**Comparison with photo-realistic style transfer.** Propagating colors of a reference image is the problem solved by photo-realistic style transfer methods [Luan et al., 2017; Li et al., 2018]. These method replicate the global look but little emphasize is put on transferring the exact colors (see Figure 5.11).

**Quantitative evaluation.** Our test set consists of 69 videos which span a large range of scenarios with videos containing various amounts of motions, occlusions/disocclusion, change of background and object appearing/disappearing. Due to their prohibitive running time, some methods [Xia et al., 2016; Luan et al., 2017] are not included in this quantitative evaluation. Table 5.2 and Figure 5.12 and 5.13 show the details of this evaluation. For a better understanding of the temporal behavior of the different
Neural Network for Color Propagation

<table>
<thead>
<tr>
<th>N</th>
<th>Gray</th>
<th>BSolver</th>
<th>Style</th>
<th>VideoProp</th>
<th>SepConv (local only)</th>
<th>Matching (global only)</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>33.65</td>
<td>41.00</td>
<td>32.94</td>
<td>34.96</td>
<td>42.72</td>
<td>38.90</td>
<td><strong>43.64</strong></td>
</tr>
<tr>
<td>20</td>
<td>33.66</td>
<td>39.57</td>
<td>32.81</td>
<td>34.65</td>
<td>41.01</td>
<td>37.97</td>
<td><strong>42.64</strong></td>
</tr>
<tr>
<td>30</td>
<td>33.66</td>
<td>38.59</td>
<td>32.70</td>
<td>34.45</td>
<td>39.90</td>
<td>37.43</td>
<td><strong>42.02</strong></td>
</tr>
<tr>
<td>40</td>
<td>33.67</td>
<td>37.86</td>
<td>32.61</td>
<td>34.26</td>
<td>39.08</td>
<td>37.02</td>
<td><strong>41.54</strong></td>
</tr>
<tr>
<td>50</td>
<td>33.68</td>
<td>37.40</td>
<td>32.54</td>
<td>34.13</td>
<td>38.56</td>
<td>36.75</td>
<td><strong>41.23</strong></td>
</tr>
</tbody>
</table>

Table 5.2: Quantitative evaluation: Using PSNR in Lab-space we compute the average error over the first N frames. We compare to the following methods: Bilateral Solver [Barron and Poole, 2016], Style-transfer [Li et al., 2018], Video PropNet [Jampani et al., 2017] and SepConv [Niklaus et al., 2017b], which corresponds to the local only part.

methods, we plot the error evolution over time averaged for all sequences. As error measurement we use the PSNR in Lab-space. On the first frames, our results are almost indistinguishable from a local strategy (with very similar error values) but we quickly see the benefit of the global matching strategy. Our approach consistently outperforms related approaches for every frame and is able to propagate colors within a much larger time frame. Results of the video propagation networks [Jampani et al., 2017] vary largely depending on the sequence, see Figure 5.13. This explains the inconsistent numerical performance averaged over our large test set compared to the visual quality of the selected images shown in the qualitative comparisons before. The results of the propagation network by [Jampani et al., 2017] often suffer from some color shift and temporal instability, see Figure 5.14.
5.3 Results

**Figure 5.12:** Temporal evaluation. The average PSNR error per frame shows the temporal stability of our method and its ability to maintain a higher quality over a longer period.

**Figure 5.13:** Temporal evaluation of examples sequences. The average PSNR error per frame shown for individual sequences. See Figure 5.15 for example images. Except for Video PropNet, the general behavior is consistent over different sequences.
Neural Network for Color Propagation

Figure 5.14: Comparison with Video PropNet. The results of the Video PropNet by [Jampani et al., 2017] suffer from color shift leading to the decreased accuracy in the quantitative evaluation. (Input images: [Pont-Tuset et al., 2017; Bonneel et al., 2015])

Figure 5.15: Example images. Selection of sequences used for temporal evaluation. (Image sources: [Butler et al., 2012; Pont-Tuset et al., 2017; Bonneel et al., 2015])

5.4 Discussion and Limitations

Our proposed method for color propagation significantly benefits from the global matching. While the local and global method perform individually about the same in the quantitative evaluation for long range propagation, only the combination leads to the superior performance of our method. As a consequence, improving each individual part has the potential to lead to further improvements of the overall method. The local propagation using warping may benefit from any improvement in motion estimation between frames. Our proposed matching algorithm currently only consists of nearest neighbour matching on two levels without any additional constraints. Using a different similarity metric or considering for example the k-nearest matching could lead to a more robust result.

While we have only applied our algorithm to the propagation of colors, it
5.4 Discussion and Limitations

Reference \((k = 0)\) Propagation result \((k = 10)\)

**Figure 5.16: Other applications.** Applying our network unmodified to other applications only works in case of recolorization and fails for more complex transformations such as stylization (second row). (Input image: [Wang et al., 2017], Style 1: Adobe Photoshop’s Filter ‘Oil Paint’, Style 2: Udine, by Francis Picabia, with [Engstrom, 2016])

Reference Y-channel u-channel v-channel

**Figure 5.17: Yuv-decomposition of various transformations.**
would be interesting to see, how the propagation of other image transformation can benefit from the combination of a local and global strategy. Applying our algorithm without any modifications only works for re-colorization applications and fails for more complex transformations, see Figure 5.16. The main reason lies in the design of the fusion network. For colorization it is enough to only predict the chrominance values and the luminance channel can be used for guidance. In the example of style transfer however, we observe that most of the change is represented in the luminance channel, see Figure 5.17. This would require to adapt and retain the network for this different situation. Furthermore we would also need to adapt the matching. For style transfer we are more interested in transferring the global structure instead of the fine details. The result of the coarse matching is shown in Figure 5.18. Improving the result would require some additional constraints to remove the blocking artifacts of these preliminary results.
Phase-based Frame Interpolation

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.

In this chapter we propose a new approach for frame interpolation, *PhaseNet*, that addresses the limitations of previous methods, see Figure 6.1. Our method is designed to robustly handle challenging scenarios while also coping with larger motion than previous phase-based interpolation methods. Our approach consists of a neural network decoder that directly estimates the phase decomposition of the intermediate frame instead of pixel color values.

### 6.1 Challenges

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 [Meyer *et al.*, 2015; Wadhwa *et al.*, 2013], see also Chapter 3. The idea is to use the phase representation of the images as input to the network. Our goal is then to directly estimate the phase value of the intermediate image. To illustrate our motivation, we adapt the example used in Chapter 3.
Phase-based Frame Interpolation

**Figure 6.1:** Video frame interpolation. Compared to recent kernel based method [Niklaus et al., 2017b], our approach is able to handle complex scenarios containing motion blur or light changes. It also improves over existing phase-based interpolation methods [Meyer et al., 2015] relying on heuristics, which are limited in their motion range. (Input image: [Li et al., 2016])

**Motivation.** We first introduce the concept and challenges of phase-based motion representation for frame interpolation. To illustrate them we use one dimensional sinusoidal functions \( y = A \sin(\omega x - \phi) \), where \( A \) is the amplitude, \( \omega \) the angular frequency and \( \phi \) 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 6.2. 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 \( 2\pi \)-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. [Meyer et al., 2015] describes a heuristic on how to correct the phase difference to correspond to the actual spatial motion. In this thesis, instead, we propose to learn to directly predict the phase value of the desired intermediate result.

**Phase prediction.** The goal of our network is to predict the phase values of the intermediate frame, based on the steerable pyramid decomposition of
6.2 PhaseNet

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.

Figure 6.2: 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.

the input frames. Each level of the multi-scale pyramid represents a band of spatial frequencies. The phase computation according to Equation (3.9) 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 [Meyer et al., 2015] to propagate the information upwards in the pyramid, we propose using a convolutional network to learn how to combine the available phase information. Furthermore, by using a neural network design we are able to incorporating not only information from previous frequency levels (as in [Meyer et al., 2015]) but also from other orientations as well as neighbouring pixels.

6.2 PhaseNet

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
Phase-based Frame Interpolation

Figure 6.3: PhaseNet architecture. Given two consecutive frames, their decomposition can be obtained by applying the steerable pyramid filters ($\Psi$). 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.

values, our network predicts the values of the steerable pyramid decomposition.

6.2.1 Learning Phase-based Interpolation

The color input frames $I_1$ and $I_2$ are decomposed using the steerable pyramid (Eq. (3.5)). We denote the obtained decomposition as $R_1$ and $R_2$, respectively:

$$R_i = \Psi(I_i) = \{(\phi^i_{\omega,\theta}, A^i_{\omega,\theta})|\omega, \theta\}, r^i, r^i_h\}.$$  \hspace{1cm} \text{(6.1)}

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 $\Lambda$. The interpolated frame $\hat{I}$ is given by

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

where $\Psi^{-1}$ the reconstruction function.
6.2 PhaseNet

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, $I$:

$$\Lambda^* = \arg\min_{\Lambda} \mathbb{E}_{I_1, I_2, I \sim \mathcal{D}} [\mathcal{L}(F(R_1, R_2; \Lambda), I)].$$  \hfill (6.3)

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 $\ell_1$-norm of pixel differences which has been shown to lead to sharper results than $\ell_2$ [Long et al., 2016; Mathieu et al., 2015; Niklaus et al., 2017b]:

$$\mathcal{L}_1 = ||I - \hat{I}||_1.$$ \hfill (6.4)

**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 $\phi$. The phase loss is then defined as the $\ell_1$ loss of the phase difference values over all levels ($\omega$) and orientations ($\theta$):

$$\mathcal{L}_{\text{phase}} = \sum_{\omega, \theta} ||\Delta \phi_{\omega, \theta}||_1,$$ \hfill (6.5)

where $\Delta \phi$ is defined as

$$\Delta \phi = \text{atan2}(\sin(\phi - \hat{\phi}), \cos(\phi - \hat{\phi})).$$ \hfill (6.6)

We use $\text{atan2}$, the four-quadrant inverse tangent, which returns the smaller angular difference between $\phi$ 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:

$$\mathcal{L} = \mathcal{L}_1 + \nu \mathcal{L}_{\text{phase}}.$$ \hfill (6.7)
In our experiments the weighting factor $v$ is chosen such that the phase loss is one order of magnitude larger than $L_1$, i.e. $v = 0.1$.

### 6.2.2 Network Architecture

The architecture of PhaseNet is visualized in Figure 6.3. 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 $\omega$ and orientation $\theta$, as well as the low pass residual. Before passing them through the network we normalize the phase values by dividing by $\pi$. 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 6.4) 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 [Ioffe and Szegedy, 2015] and ReLU nonlinearity [Nair and Hinton, 2010], which have shown to help training. Each convolution layer produces 64 feature maps by either using $1 \times 1$ or $3 \times 3$ convolution filters, see Table 6.1 for details. In general, we observe, that smaller kernels are preferable for lower resolution. Between levels the resolution is increased by the scaling factor $\lambda$, which has been used to produce the steerable pyramid. Resizing is done by bilinear interpolation noted as $up()$ in Table 6.1. 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
Table 6.1: Details of the PhaseNet architecture. The numbers of the channels and resolutions correspond to the case of using one color channel (weights reused for the other two) and a pyramid constructed with $\lambda = \sqrt{2}$ and 4 orientations. The $+$ in the input column corresponds to concatenating the channels. Each PhaseNet block PNB consists of two convolution layers, both followed by batch normalization and leaky ReLU nonlinearity with a factor of 0.2. The prediction layers pred$_i$ consist each of one convolution layer followed by the hyperbolic tangent function. pyr$_i$ summarizes the steerable pyramid decomposition information of the input images at the corresponding level $i$, i.e. low level residuals for $i = 0$ and phase and amplitude information for $i > 0$. In total this network has about 460k trainable parameters.
Figure 6.4: 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.

through one convolution layer with filter size $1 \times 1$ 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 6.2.3. 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 number of 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 6.3.
6.2 PhaseNet

6.2.3 Image Reconstruction

In general we can reconstruct an image from the steerable pyramid decomposition by integrating over all pyramid levels according to Equation 3.1 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 $\omega$ and orientation $\theta$.

To compute the phase values $\hat{\phi}$ of $\hat{R}$ we scale the predicted values by multiplying them with $\pi$. To approximate the low level residuals and the amplitudes of the intermediate frame [Meyer et al., 2015] 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 $R_2$:

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

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

where $\alpha$ and $\beta$ 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.

6.2.4 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 6.5.

This training procedure can be seen as a form of curriculum learning [Bengio et al., 2009] 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
Phase-based Frame Interpolation

Figure 6.5: 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.

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 [Gardner et al., 2017] 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.

Our training dataset consists of about 10k triplets of frames from the DAVIS2017 video dataset [Perazzi et al., 2016; Pont-Tuset et al., 2017]. Table 6.2 summarizes the statistics of the dataset. At each iteration we randomly select patches of $256 \times 256$ pixels. We perform data augmentation through horizontal and vertical flipping of the patches. To build the pyramid decomposition we use a scale factor of $\lambda = \sqrt{2}$ leading to a pyramid of 10 levels. Due to reusing the weights across the color channels and some of the layers, our network has only about 460k trainable parameters in total.

We use Adam optimizer [Kingma and Ba, 2014] with $\beta_1 = 0.9$, $\beta_2 = 0.999$
6.3 Results

<table>
<thead>
<tr>
<th></th>
<th>DAVIS2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>#video clips</td>
<td>145</td>
</tr>
<tr>
<td>#video frames</td>
<td>10127</td>
</tr>
<tr>
<td>mean #frames per clip</td>
<td>69.8</td>
</tr>
<tr>
<td>resolution</td>
<td>480p</td>
</tr>
</tbody>
</table>

Table 6.2: Training dataset. Statistics of the DAVIS2017 dataset [Pont-Tuset et al., 2017] we use to train our network.

and learning rate 0.001. The batch size used is 32 and reduced to 16 and 12, respectively, for the highest two training stages due to memory limitations. We train the lower levels for 12 epochs each and the highest two for 6 epochs due to the reduced batch size to have approximately the same number of iteration steps for each hierarchical level.

**Computation Time.** PhaseNet is implemented in Tensorflow and takes advantage of efficient spectral decomposition layers. With one Nvidia Titan X (Pascal), training our model (~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).

6.3 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 [Xu et al., 2012], which is on of the best ranked methods 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 [Baker et al., 2011]. According to Middlebury, MDP-Flow2 is followed closely by [Niklaus et al., 2017b], a neural network based method learning seperable convolution filters for frame interperolation (SepConv). In terms of phase-based representation methods for frame interpolation we compare to [Meyer et al., 2015] (Phase). The image sequences used are from the footage of [Li et al., 2016], Blender Foundation [Blender, 2008], Vision Research [Vision Research, 2015] and YouTube [Lucid, 2014; Explosion, 2017]. To produce the results of these methods, we use the code and trained models provided by the original authors.
**Phase-based Frame Interpolation**

![Figure 6.6: Phase loss. Using the phase loss gives sharper results compared to only using the image loss. (Input image: [Lucid, 2014])](image)

**Figure 6.6: Phase loss.** Using the phase loss gives sharper results compared to only using the image loss. (Input image: [Lucid, 2014])

![Figure 6.7: GAN loss. Adding an adversarial loss can help to restore some sharpness, e.g. the zipper, but it also adds false positives such as additional hair. (Input image: [Lucid, 2014])](image)

**Figure 6.7: GAN loss.** Adding an adversarial loss can help to restore some sharpness, e.g. the zipper, but it also adds false positives such as additional hair. (Input image: [Lucid, 2014])

**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 ($\mathcal{L}_{\text{phase}}$). 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 6.6. 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. (6.7). 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. We also experimented with additional losses such as adding a discriminator and train it together with our PhaseNet as generator in a generative adversarial network setup. While adding such a GAN loss helps to increase the amount of details in the prediction it also adds additional details which are not present in the original frame. See Figure 6.7 for a visual comparison.

**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 $256 \times 256$ leading to a pyramid of 10 levels. To produce
6.3 Results

For images larger than the training patches, reusing last layers weights (i.e. ours) over averaging the lowest levels of the decomposition is beneficial. (Input image: [Lucid, 2014; Li et al., 2016])

Figure 6.8: High resolution. For images larger than the training patches, reusing last layers weights (i.e. ours) over averaging the lowest levels of the decomposition is beneficial. (Input image: [Lucid, 2014; Li et al., 2016])

Qualitative comparisons. We evaluate our method on a set of challenging image pairs including motion blur and extreme light changes, see Figure 6.9 and Figure 6.10. 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
Phase-based Frame Interpolation

Figure 6.9: Visual comparison with frame interpolation methods on challenging scenarios with motion blur. See text for details and discussion. (Input images: [Li et al., 2016])

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 6.11.

Quantitative comparisons. We use the same set of sequences as in [Meyer et al., 2015], consisting of representative scenes with many moving parts and challenging lighting conditions as well as one synthetic example (Roto) containing many high frequencies. Example images of the evaluated sequences are shown in Figure 6.14. 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 6.12 we report the error measurements using the structural similarity (SSIM) measure. In general, the optical flow method and SepConv achieve a better error 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 6.13. For the light sequence (second row), our approach produces noticeably better results. For the fireman
6.3 Results

**Figure 6.10:** Visual comparison with frame interpolation methods on challenging scenarios with brightness change. See text for details and discussion. (Input images: © Vision Research [Vision Research, 2015], [Explosion, 2017])

**Figure 6.11:** Advantage of a data driven approach. Using heuristics [Meyer et al., 2015] 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. (Input images: © Blender Foundation [Blender, 2008], © Vision Research [Vision Research, 2015])
Phase-based Frame Interpolation

![Figure 6.12: Error measurements of different methods for different sequences by computing the structural similarity measurement (SSIM) averaged over several frames.](image)

![Figure 6.13: Comparison of interpolation results with our method and separable convolution filters to the ground truth including a difference map using absolute differences. (Input images: © Vision Research [Vision Research, 2015])](image)

sequence (first row), although the difference map shows a global degeneration for high frequency content for our method, there is no perceptual difference between the different methods. Furthermore, our method produces visually preferable results over the purely phase-based method [Meyer et al., 2015], see Figure 6.11. However, this is also not reflected in the quantitative comparisons. A possible explanation for this is the approach we have chosen to handle larger resolutions. We suggested to reuse weights for additional pyramid levels if the input images are larger than the training patches. Even though this generalization works perceptually well it introduces some low frequency error visible as flickering. This is also illustrated in Figure 6.15, left side.
6.3 Results

**Figure 6.14:** *Example images* from the sequences used for the error measurements. (Image source: [Lucid, 2014])

**Figure 6.15:** *Patch by patch.* Processing the images patch by patch results in less flickering errors, visible here in the reduction of the shining areas in the image differences. (Input image: [Lucid, 2014])
Phase-based Frame Interpolation

Figure 6.16: Resizing input images before processing them patch by patch helps to restore some high frequency details. (Input image: [Lucid, 2014])

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Patch</th>
<th>Resize</th>
<th>Adversarial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier</td>
<td>SSIM</td>
<td>0.940</td>
<td>0.941</td>
<td>0.962</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>36.518</td>
<td>37.107</td>
<td>39.576</td>
</tr>
<tr>
<td>Couple</td>
<td>SSIM</td>
<td>0.947</td>
<td>0.948</td>
<td>0.961</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>37.833</td>
<td>38.242</td>
<td>40.322</td>
</tr>
<tr>
<td>Face</td>
<td>SSIM</td>
<td>0.927</td>
<td>0.927</td>
<td>0.925</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>39.921</td>
<td>40.416</td>
<td>40.602</td>
</tr>
<tr>
<td>Hair</td>
<td>SSIM</td>
<td>0.939</td>
<td>0.942</td>
<td>0.941</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>37.566</td>
<td>38.233</td>
<td>38.285</td>
</tr>
<tr>
<td>Sand</td>
<td>SSIM</td>
<td>0.892</td>
<td>0.898</td>
<td>0.896</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>36.238</td>
<td>37.148</td>
<td>37.265</td>
</tr>
<tr>
<td>Rotozoom</td>
<td>SSIM</td>
<td>0.941</td>
<td>0.945</td>
<td>0.948</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>38.297</td>
<td>38.922</td>
<td>39.324</td>
</tr>
<tr>
<td>Fireman</td>
<td>SSIM</td>
<td>0.938</td>
<td>0.947</td>
<td>0.975</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>35.354</td>
<td>36.046</td>
<td>38.574</td>
</tr>
<tr>
<td>Light</td>
<td>SSIM</td>
<td>0.973</td>
<td>0.981</td>
<td>0.983</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>43.753</td>
<td>45.474</td>
<td>45.937</td>
</tr>
<tr>
<td>Average</td>
<td>SSIM</td>
<td>0.937</td>
<td>0.941</td>
<td>\textbf{0.949}</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>38.185</td>
<td>38.949</td>
<td>\textbf{39.986}</td>
</tr>
</tbody>
</table>

Table 6.3: Variations of PhaseNet. Quantitative comparison of different training and testing procedures of PhaseNet. We evaluate the baseline, baseline+patch processing and baseline+patch+resizing as well as adversarial trained+patch+resizing. As error measurement we used SSIM as well as PSNR computed in the LAB color space.
6.3 Results

<table>
<thead>
<tr>
<th></th>
<th>PhaseNet</th>
<th>SepConv</th>
<th>Phase</th>
<th>MDP-Flow2</th>
<th>TOFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier</td>
<td>SSIM</td>
<td>0.962</td>
<td>0.967</td>
<td>0.979</td>
<td>0.984</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>39.576</td>
<td>41.042</td>
<td>39.799</td>
<td>41.690</td>
</tr>
<tr>
<td>Couple</td>
<td>SSIM</td>
<td>0.961</td>
<td>0.964</td>
<td>0.960</td>
<td>0.965</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>40.322</td>
<td>41.263</td>
<td>40.296</td>
<td>39.926</td>
</tr>
<tr>
<td>Face</td>
<td>SSIM</td>
<td>0.925</td>
<td>0.918</td>
<td>0.921</td>
<td>0.930</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>40.602</td>
<td>40.606</td>
<td>40.393</td>
<td>41.133</td>
</tr>
<tr>
<td>Hair</td>
<td>SSIM</td>
<td>0.941</td>
<td>0.947</td>
<td>0.954</td>
<td>0.963</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>38.285</td>
<td>39.558</td>
<td>38.674</td>
<td>39.926</td>
</tr>
<tr>
<td>Sand</td>
<td>SSIM</td>
<td>0.896</td>
<td>0.888</td>
<td>0.895</td>
<td>0.903</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>37.265</td>
<td>37.140</td>
<td>37.324</td>
<td>37.755</td>
</tr>
<tr>
<td>Rotozoom</td>
<td>SSIM</td>
<td>0.948</td>
<td>0.953</td>
<td>0.941</td>
<td>0.956</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>39.324</td>
<td>41.208</td>
<td>38.445</td>
<td>41.432</td>
</tr>
<tr>
<td>Fireman</td>
<td>SSIM</td>
<td>0.975</td>
<td>0.991</td>
<td>0.941</td>
<td>0.984</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>38.574</td>
<td>42.770</td>
<td>36.498</td>
<td>40.664</td>
</tr>
<tr>
<td>Light</td>
<td>SSIM</td>
<td>0.983</td>
<td>0.982</td>
<td>0.983</td>
<td>0.986</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>45.937</td>
<td>46.514</td>
<td>46.095</td>
<td>47.036</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SSIM</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.949</td>
<td>39.986</td>
</tr>
<tr>
<td></td>
<td>0.951</td>
<td>41.263</td>
</tr>
<tr>
<td></td>
<td>0.947</td>
<td>39.691</td>
</tr>
<tr>
<td></td>
<td>0.959</td>
<td>41.195</td>
</tr>
<tr>
<td></td>
<td>0.949</td>
<td>40.738</td>
</tr>
</tbody>
</table>

**Table 6.4:** Other interpolation methods. Quantitative comparison of the improved procedure of PhaseNet to other interpolation methods. We compare to SepConv [Niklaus et al., 2017b], Phase [Meyer et al., 2015], MDP-Flow2 [Xu et al., 2012] and TOFlow [Xue et al., 2017].

**Additional experiments.** In order to improve our results, we conduct for this thesis some additional experiments going beyond what has been presented in the original paper. Without altering the architecture of the original PhaseNet, we investigate how training and testing procedures influence the results visually and numerically. As a baseline we retrain the network on a larger dataset. We extend the dataset described in Table 6.2 with the Vimeo-90K dataset from [Xue et al., 2017]. Because this additional dataset contains much smaller motions between the frames, the hierarchical training is not necessary anymore. The obtained error measurements are listed in the first column of Table 6.3. Instead of sharing the weights for additional pyramid layers if the images are larger, it is also possible to process the images by patches of the same size as the training patches. This helps to reduce the flickering visually, see Figure 6.15. Consistently, it also results in better error measurements. We can further improve the results by resizing the images by a factor of 2 before extracting the patches. Frequency content which is not
Phase-based Frame Interpolation

Figure 6.17: Large motion with patches. Processing the images patch by patch increases the accuracy by capturing more details but it reduces the motion range the method can interpolate correctly. (Input image: [Li et al., 2016])

represented in the pyramid levels and only captured in the high frequency does not contain phase information and is therefore not considered in our implementation. By resizing the images first, however, we can artificially reduce the information in this high frequency residual and restore some details as a result. Figure 6.16 illustrates this improvement visually. However this happens with the cost of reducing the range of motion the method can handle, see Figure 6.17. While these changes only alter the way how the images are processed at test time, retraining the network with an adversarial loss can also help to restore some sharpness. However, training with an adversarial loss also adds false positives, see Figure 6.7, thus it does not help to improve the overall result in terms of error measurements. See Table 6.3 for the full numerical comparisons. In Table 6.4 we compare our best performing version (processing images by patches with resizing) to other frame interpolation methods. Additionally to the previous used methods we also added the concurrent work of [Xue et al., 2017] to our comparisons. With the presented improvements we are able to reduce or even close the gap to the other methods.

6.4 Discussion and Limitations

Our method significantly improves over previous phase-based methods for frame interpolation, both in terms of motion range and high frequencies. We still however do not reach the same level of detail as methods which explicitly match and warp pixels. One reason for this is, that phase-based method decompose images into a pyramid, each level corresponding to a frequency band. High-frequency content which cannot be captured is stored in a residual for which we cannot compute phase information. Consequently we can also not interpolate that information. As a result, in the current design of PhaseNet we just ignore this information.
Ideally you would train the network on the full resolution of the images which is computationally not feasible. We suggested to split the images up into patches of the same size as the training patches if the input images are larger. But the decomposition of smaller images has fewer pyramid levels which limits the motion that can be interpolated, see Figure 6.17. How to improve the sharpness further without restricting the motion range remains an open research question.
Phase-based Frame Interpolation
CHAPTER 7

Conclusion

In this thesis we presented some novel methods for video frame interpolation and editing using implicit motion estimation techniques. Namely, we developed a purely phase-based method to propagate general modifications in a video as well as a learning based approach for the special case of color propagation. Furthermore we leveraged the phase-based motion representation with the potential of neural networks and showed its advantages for frame interpolation.

In the following, we will review the principal contributions of the thesis and discuss the limitations of the presented work as well as possible directions for further work.

7.1 Review of Principal Contributions

In this thesis we have shown how implicit method estimation methods can be used successfully for various video processing applications. By evaluating and addressing their limitations with newly developed algorithms, we were able to extend their motion range as well as their robustness.

In Chapter 4 we have shown how a purely phase-based approach can be used for edit propagation in videos. Because this method does not require any explicit matching it also works on textureless or homogeneous image regions. We have evaluated the flexibility of our method by applying it to various applications from adding novel image elements and video colorization to propagation of general image filters.
Conclusion

In Chapter 5 we have presented a new data driven approach for color propagation in videos. We combined a local method, that consists of a frame by frame image warping based on predicted kernels, with a global strategy using semantical information, based on feature matching and color transfer. We have shown how our developed framework extends the temporal range to which colors can be propagated. Our extended comparative results show that the proposed approach outperforms recent methods in image and video color propagation as well as style transfer.

In Chapter 6 we have presented a method for frame interpolation which combines the advantage of phase-based and data driven approaches. We implemented a neural network architecture that uses the phase-based representation of the images as input and synthesizes an interpolated frame from its predicted phase-values. By combining both a phase loss and standard $\ell_1$-norm we are able to produce visually preferable results over optical flow for challenging scenarios containing motion blur and brightness changes.

To conclude, we believe that, in particular in the context of the steady increase in video frame rate and resolution, approaches using implicit motion estimation provide an interesting and efficient alternative to traditional approaches that require explicit frame-to-frame pixel correspondences. We think that implicit motion estimation methods bear potential for more interesting research and applications, and hope that our work provides an important step in such a direction.

7.2 Future Work

While we have presented some advances on how to apply implicit motion estimation for some specific applications, the methods are still far from perfectly solving the complex challenge of motion estimation. While we have discussed the specific technical issues of each method in the corresponding chapters, we will discuss in the following the limitations of the presented work on a broader view as well as possible directions for future work.

Currently still the main limitation of phase-based methods is that the synthesized images cannot reach the same level of details as approaches that explicitly match and warp pixels. There are several causes for the introduced blur. One reason is the fact that the frequency content which cannot be captured in any pyramid level is summarized in high and low-pass residuals, respectively. Consequently the motion of them can currently not be estimated with the phase. Reintroducing or preserving the details captured in
the high-pass residual is an interesting area for future work. Possible directions could be using ideas from the area of super-resolution to “hallucinate” the high frequencies from the neighbouring images or the combination with optical flow based methods. The latter in general may be interesting to explore in order to combine the advantages of both approaches.

Also the amount of motion which implicit motion estimation methods can handle is limited. In the case of kernel-based method, motion is limited by the kernel size. Phase-based methods on the other hand can only capture motion within the spatial support of the filters. Increasing the spatial support results in decreasing the bandwidth in frequency. We cannot be arbitrary accurate in location both in space and frequency. By narrowing the frequency bands we lose the spatial localization property and therefore the ability to measure the spatial motion. As this is a conceptual limitation of the phase-based motion estimation it remains an open question whether other filter designs or even learning the filters could be more suitable and lead to improved results.

Being able to interpolate video sequences and propagate edits over time efficiently and accurately can be useful in other areas as well. As the methods are suitable for high resolution data it can be beneficial for example in the movie production pipeline. The color propagation method can reduce the number of frames manually edited in the case of color grading and correction. Frame interpolation can be used to reduce the rendering time and to create previews. As both methods can be used to synthesize missing images in a sequence based on existing ones, they can also be interesting for the efficient compression of videos.

While we investigated only the applications of frame interpolation and video editing in this thesis, implicit motion estimation methods may also be beneficial for other applications which require some notion of motion. Examples could be in the area of super resolution or video segmentation. They also bear potential for new applications that were not possible before as the phase-based method for example is suitable to capture even very small motions.

An issue we have not addressed in thesis is the evaluation of the perceptual quality of images. Comparison of images with numerical measurements do often not correspond well with the preferences of humans. This makes it difficult to evaluate the algorithms. In general, having a representation of images that corresponds well to the perceptual judgement of humans will be beneficial for a broad area of visual tasks.
Conclusion
References


References


[Didyk et al., 2013] Piotr Didyk, Pitchaya Sitthi-amorn, William T. Freeman, Frédéric Durand, and Wojciech Matusik. Joint view expansion and filtering for


References


[Hommos et al., 2018] O. Hommos, S. L. Pintea, P. S. M. Mettes, and J. C. van
References


References


References


References


[Reda et al., 2018] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanaro. Sdc-net: Video pre-


References


[Vondrick et al., 2016] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba.
References


References


References


