11-17-2022

Domain Knowledge as Motion-Aware Inductive Bias for Deep Video Synthesis: Two Case Studies

Long Mai
Portland State University

Domain Knowledge as Motion-Aware Inductive Bias for Deep Video Synthesis: Two Case Studies

by

Long Mai

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dissertation Committee:
Feng Liu, Chair
Bart Massey
Atul Ingle
Dan Hammerstrom

Portland State University
2022

Abstract

Deep neural networks have been part of many breakthroughs in computer graphics and vision research. In the context of visual content synthesis, deep learning models have achieved impressive performance in the image domain. However, adapting the successes of image synthesis models to the video domain has been difficult, arguably due to the lack of sufficiently strong inductive biases that encourage the models to capture the temporal-dynamic nature of video data. Inductive bias refers to the prior knowledge incorporated into the learning models to explicitly drives the learning process toward the solutions that capture meaningful structures from data, which is critical to help the model generalize beyond the training data. Successful deep neural network architectures, such as convolutional neural networks (CNN), while effective in representing image data thanks to the spatial inductive bias, often lack the inductive biases relating to the dynamic nature of videos. I argue that designing such inductive biases can benefit from the domain knowledge of video processing literature. My primary motivation in this thesis is to demonstrate that the knowledge acquired from traditional computer vision and graphics literature can serve as effective inductive biases for designing deep learning models for video synthesis. This dissertation provides the initial steps toward verifying that insight via two case studies.

In the first case study, I explored adapting the standard CNN architecture to perform video frame interpolation. Early CNN-based methods for frame generation

followed the direct prediction approach, thus ineffective in learning to capture motion information. That often results in visual distortions and blurry results. Inspired by traditional video frame interpolation techniques that established frame interpolation as a joint process of motion estimation and pixel re-sampling, I presented our CNN-based frame interpolation framework that incorporated such insight into the synthesis model via the novel AdaConv layer. That serves as a functional inductive bias and enables the first deep learning model for high-quality video frame interpolation.

In the second case study, I explored adapting the recent Implicit Neural Representation (INR) to a novel motion-adjustable video representation. Viewing modern INR frameworks as a form of non-linear transform from a frequency domain to the image domain, and inspired by the success of phase-based motion modelling in the classical computer vision literature, I presented a simple modification to the standard image-based INR model that allows for not only video reconstruction but also a variety of motion editing tasks.

Dedication

My PhD study is a journey full of joy, and excitement. Along the way, I am blessed with the opportunities to know and work with so many wonderful people. They helped me learn a great deal and make my whole journey not only memorable but also immensely enjoyable. I dedicate this dissertation to all of them.

I want to thank my advisor, Professor Feng Liu, for his unconditional support for me over many years. I learned from him not only the skills to do well in research, but also the mindset to become a good scientist. I'm sure I would not be where I am now without him.

I thank my friends and labmates at Portland State University and our Computer Graphics & Vision Lab for making my student life the best experience. I am grateful to my friends, Hoang le and Huy Tran, for the great friendships that span beyond two decades and two continents; and to Simon Niklaus, for being not only the best collaborator I have ever had but also one of my best friends for life.

I am forever in debt to my parents. Every step I took in life, they have always been there for me. Every success I have ever achieved in life, no matter how big or small, would never be possible without their loves.

And last but not least, I want to thank my dear wife Amy for always supporting every decision I made, no matter how crazy they were.

Table of Contents

List of Tables

List of Figures

1 Introduction

In recent years, video has become a major form of media for entertainment and communication. In 2022, it is estimated that video data contributed more than 80% of the overall internet traffic [1] . A typical internet user spends, on average, 19 hours every week watching video content online [2] . From streaming services (Netflix [3] , Hulu [4]) to modern social video-sharing platforms (Youtube [5] , Instagram [6] , TikTok [7]), influential stories are now delivered through video content. Besides entertainment, video is also an essential form for marketing and education [4, 17, 33, 86, 100, 103].

With great demands in video consumption come great demands in video production. The growing popularity of video platforms in recent years has turned video content creation into a fruitful career path [8] . To create impactful videos, content creators need to not only capture interesting content but also transform it in creative ways into their intended stories. Therefore, video editing and post-production have become critical components in modern video production workflows [63, 130, 180].

[1] https://cyrekdigital.com/uploads/content/files/white-paper-c11-741490.pdf

[2] https://www.wyzowl.com/video-marketing-statistics/

[3] https://www.netflix.com

[4] https://www.hulu.com

[5] https://www.youtube.com/

[6] https://www.instagram.com/

[7] https://www.tiktok.com/

[8] https://www.bloomberg.com/press-releases/2022-08-03/oxygen-announces-the-state-of-the-creator-economy-report

Such ever-increasing needs for video content creation have encouraged the rapid development of advanced video processing systems in recent years. In addition to the basic editing capabilities, such as cutting and assembling video footages, modern video creation often demand advanced editing tools that can support adjusting different aspects of the captured videos to fix the imperfection incurred during the captured process, or to achieve certain artistic appearances. Video processing, and video synthesis in particular, has long been a core research topic in computer graphics and computer vision.

1.1 Video Synthesis

In video synthesis, the main goal is to synthesize novel video results from the original source videos such that certain properties are transformed according to the users' desires. Over the past decades, computer graphics and vision research have enabled advanced systems for many low-level video processing tasks. Examples include removing sensor noise in the captured footage [6, 67, 85, 97, 175], reducing camera shaking from the videos [99, 102, 112], and re-targeting the aspect ratios of the captured video to fit the display device [31, 98, 203]. They have been incorporated into successful commercial video editing applications such as Adobe Premiere [9] and Apple's Final Cut [10] .

Modern video production workflows, however, demand more than adjusting low-level pixel information. They often requires solving more sophisticated video synthesis problems in which higher-level aspects of the video content, such as objects' appearance or dynamics, need to be manipulated to fit the users' creative purposes. One example is the task of video object removal. Removing objects from the video content requires

[9] https://www.adobe.com/products/premiere.html

[10] https://www.apple.com/final-cut-pro/

beyond manipulating low-level pixel information. It requires novel information to be generated to fill in the removed regions. Importantly, the generated content needs to be consistent with the rest of the videos in terms of both appearance and dynamics. Another example is the video re-timing task. When editing videos with dramatic actions, users often want to create the *slow-motion effect* in certain part of the video to emphasize the beauty of the captured actions. Adjusting the perceived speed of the motion in the video requires more than adjusting the existing content in the video. It requires additional frames to be synthesized, and the synthesized content needs to appear consistent with the original frames.

In those modern video synthesis problems, the fundamental challenge is to generate novel contents that are visually plausible both spatially and temporally. The algorithms need to synthesize natural looking individual frames. In addition, the synthesized frames, when put together, need to exhibit the plausible temporal dynamics that are consistent with both the original content and the users' constraints. That requires the video synthesis algorithms model not only the low-level pixel information but also mid- and high-level semantics information involving the appearance and dynamics of the synthesized content.

1.2 Learning-Based Video Synthesis – The Challenge of Temporal Dynamics Modeling

In recent years, advances in machine learning have revolutionized visual computing research. In particular, learning-based techniques have been highly successful in image synthesis. The ability to learn predictive models from large image/video databases has enabled many smart image editing systems, thanks to the effectiveness of deep neural networks in modeling and generating plausible visual appearance. Intelligent tools are now available to adjust image sizes without distorting important content [7, 11, 146],

replacing unwanted regions with better content [11, 101, 217], and perform image style transfer to transform the image appearance while preserving the scene content and structures [58, 134, 150, 206].

The successes in image synthesis, however, are not easily translated into successes in the video domain. A Video is not just a collection of images. Instead, a video is a sequence of images consistently relating to each other. For that reason, applying a known image editing technique frame-wise often fails to generate plausible video synthesis results. The fundamental challenge in learning to synthesize video content is to incorporate the temporal dynamics information into the synthesis model.

Early attempts in video synthesis with deep neural networks explored adapting the successful architectures in image modeling to video data by extending the models to handle an additional time dimension. For example, the successful 2D convolution in the successful convolutional network archicture was extended to 3D convolution [148, 191]. Given the successes of their image-based counterpart in capturing image priors, it was expected that the resulting models could similarly learn to capture the video priors, including temporal dynamics information. Those efforts, while showing early promising results, do not match the level of quality obtained in the image domain. Such difficulty demonstrated that learning the representation for videos is challenging for the models designed originally for images without explicitly incorporating the specific natures of videos. Arguably, directly learning temporal dynamics information is difficult without motion-aware inductive biases [185, 198].

Inductive biases refer to the assumptions and prior knowledge incorporated with the learning algorithms via either the model architecture designs or training strategies. This mechanism helps induce the learning models to effectively capture the structure in the data and, from that, generalize better to unseen data. Inductive biases have been playing critical roles in the success of modern deep learning systems, ranging

from the spatial inductive bias incorporated in Convolution Neural Network (CNN) architectures to the 3D inductive biases [**?**] incorporated through the volumetric rendering processes [121, 125].

My research in this dissertation was motivated by the question, "What inductive biases can be incorporated to encourage deep learning models to capture useful temporal dynamics information for video synthesis tasks?" This question can be approached from different directions. Machine learning researchers and practitioners have been actively exploring this through novel architecture designs, datasets, and training strategies. In this dissertation, I am interested in exploring this question from a slightly different perspective. I argue that, orthogonal to looking for future architectural advances, it is also beneficial to look into the past by revisiting the valuable insights that have been discovered in the rich literature in more traditional computer graphics and vision research. Those insights, which rooted in the well-researched understanding about the nature of video data, can potentially serve as useful inductive biases for deep learning models.

1.3 Learning to Synthesize Videos – Domain Knowledge as Inductive Biases

To this end, it's worth noting that traditional vision/graphics methods have provided successful editing systems by leveraging what we understand about the visual data.

Understanding and modeling temporal dynamics information in video content is a long-standing problem in computer vision and computer graphics. From the foundational works in vision research that established the connection between optical flows information extracted from video frames and motion perception in the human visual system [49, 109, 181, 199], many computational methods have been developed

for motion estimation from video data [129, 145, 177, 200]. Motion representation has also been studied in depth from the signal processing perspective. Treating images as high-dimensional signals represented in the frequency domain via Fourier or wavelet decomposition, motion information in videos can be estimated from the phase information in the frequency domain. These insights in motion representation from video data provide the foundations for many video analysis tasks, ranging from camera tracking [68, 170] to object tracking [24, 215] and video coding [178, 204].

Notably, those motion modeling insights have been proven beneficial not only for video analysis but also for the video synthesis. Such domain knowledge harnessed from the foundational understanding of video signals have enabled many important works in video synthesis. For example, the connection between optical flow and motion perception has been leveraged to support video frame interpolation [8, 201, 218]. The relation between the phase shift information in frequency domain and the observed motion in the pixel domain was successfully leveraged to enable a range of motion manipulation tasks [116, 118, 192].

Comparing the traditional computer graphics- and vision-based approaches with modern learning-based approaches to video synthesis, it is interesting to observe their complementing strengths. On the one hand, traditional approaches benefit from solid domain knowledge in motion modeling and establish better-controlled synthesis processes to support high-quality synthesis. Yet, they have difficulty modeling appearance due to the lack of high-level semantics priors and adaptability to data. On the other hand, Learning-based models are good at capturing appearance thanks to powerful image models and the ability to learn from data. However, they often fail to capture motion due to the lack of motion-aware inductive bias. *Is it possible to combine the best of both worlds?*.

In this dissertation, I describe my exploration so far toward combining the strength

of deep learning models and motion modeling techniques from the more traditional computer graphics and vision research. *The key thesis statement is that the insights underlying the motion-modeling techniques in traditional techniques can effectively serve as inductive biases for learning-based approaches.* With this direction, I hope to provide a new perspective to re-think the insights offered by traditional graphics and vision works in the data-driven age. Instead of leveraging that domain knowledge in a heuristic manner, I argue that they can be leveraged as effective implicit guidance in the learning process to assist in modeling temporal dynamics information.

1.4 Two Case Studies

This dissertation verifies the key thesis above through two case studies, involving two different video synthesis settings.

First, I explore the problem of learning to perform video frame interpolation. By leveraging the insights from traditional frame interpolation works, I introduce a novel deep-learning-based video synthesis technique that can generate high-quality frame interpolation results.

Second, I explore the problem of learning implicit neural representation for videos. Inspired by the relation between the phase information of visual signals in frequency domain and their motion in space, I propose a simple modification to the existing image-based implicit neural representation network that can not only capture video data but also allows for manipulation of temporal dynamics information in the video.

1.4.1 Video Frame Interpolation with Adaptive Convolution

Frame interpolation is a classic problem in computer graphics and computer vision. Given an input video, the goal of frame interpolation is to synthesize the intermediate frames between each pair of consecutive original input frames, effectively increasing

the frame rate of the video. It is important for many video editing applications. Video frame interpolation allows users to create slow-motion effects or to fit a captured video into a new desired playback time.

Prior to this research, early learning-based systems for video frame interpolation focus on directly predicting the pixel values in the intermediate frame. This was done, for example, by extending convolutional neural network (CNN) structures from image to video domain with 3D convolutional layers. While such direct synthesis approaches showed promising early results, the synthesized videos often have low visual quality with low resolution and contain significant visual distortions. The models tend not to learn to capture the underlying temporal dynamics reasoning when trained with the only objective of predicting the pixel values.

Earlier research on video frame interpolation from computer graphics and computer vision have long established the important insight that the interpolated frames need not be considered entirely new content. Instead, they can mostly be re-sampled from the already provided input frames. The goal of video frame interpolation system is thus to capture *where* to sample from and *how* to generate the color values from the sampled regions. Traditional frame interpolation methods, therefore, are often composed of two steps: motion estimation, usually optical flow, and pixel synthesis [8]. These approaches – via explicit motion reasoning and well-engineered pixel synthesis – can enable high-quality interpolation results but suffer from two drawbacks. First, flow-based pixel synthesis cannot reliably handle the occlusion problem and often lead to noticeable artifacts in interpolated video frames [117]. Second, they rely entirely on hand-crafted procedures and thus cannot learn from data.

To this end, I present our approach to incorporate the re-sampling insight from traditional graphics-based methods into an end-to-end learning system. In particular, our method considers pixel interpolation as convolution over corresponding image

patches in the two input video frames. Our system estimates the convolutional kernel instead of directly regressing to pixel values. The convolution kernel captures both the local motion between the input frames and the coefficients for pixel synthesis. This allows us to model video interpolation as a single process. This frame interpolation deep convolutional neural network can be directly trained end-to-end using widely available video data.

1.4.2 Motion-Adjustable Neural Implicit Representation for Video with Phase-Varying Positional Encoding

Implicit Neural Representation (INR) is an emerging paradigm for visual data representation. INR represents visual data as continuous functions rather than discretized structures, making it a faithful representation of the underlying signals. Initially developed for 3D shape and scene representation, INR has recently been adapted to image modeling to enable applications such as image generation [5, 165], image compression [48], and image super-resolution [29].

Compared to their image-based counterpart, video-based INR has been relatively under-explored. Existing works often consider videos as straightforward extensions of images, treating videos as 3D volumes and applying a direct video fitting approach without explicitly modeling temporal dynamics information [113, 163]. Such direct fitting approaches, therefore, cannot allow motion in the fitted videos to be edited due to the lack of explicit motion modeling.

In this work, I explore temporal dynamics modeling in the context of implicit neural video representation. Observing that contemporary image-based INR – with the use of Fourier-based positional encoding – can be viewed as a mapping from sinusoidal patterns with different frequencies to image content, I hypothesize that it is possible to generate temporally varying content with a single image-based INR model by displacing

its sinusoidal input patterns over time. Inspired by phase-based motion processing approaches in computer graphics and computer vision literature [51, 59, 118, 192] that built on the connection between motion information in a video and its phase information extracted through frequency domain analysis [195] to enable various motion editing applications, I proposed to explore leveraging phase information embedded in the Fourier-based positional encoding to help implicit neural representation models learn temporal dynamics information in video data. Instead of directly extracting phase information from the video, the proposed method exploit phase-based motion modeling as an inductive bias in INR model design, enabling an INR model that can not only learn to fit the video data but also enables re-synthesizing the videos with different motion manipulation tasks using the same framework.

1.5 Outline

The remaining of this dissertation is structured as follows. I first provide a brief overview of the literature on video synthesis and position my research in this rapidly developing field (chapter 2). I will go into detail the two case studies in toward incorporating domain knowledge from traditional computer graphics and vision research as inductive biases for motion-aware learning in deep video synthesis models. In particular, chapter 3 elaborates our adaptive convolution framework for video frame interpolation. In chapter 4, I then introduce our idea of incorporating phase-based motion modeling into neural implicit representation to enable a novel motion-adjustable neural video representation. I will finally summarize our findings and discuss the potential directions for future work in chapter 5.

2 Relevant Literature

In this chapter, I will first briefly review the early developments that made photo-realistic image and video synthesis possible. I will discuss the historical works in computer graphics and computer vision along with the valuable insights they have developed on understanding and representing visual data. Finally, I will discuss more recent efforts in renovating image/video synthesis systems with deep learning advances.

2.1 Visual Computing Research and Early Successes in Visual Content Synthesis

Synthesizing realistic imagery requires a deep understanding of visual data. By combining advances in computer graphics, computer vision, and signal processing, research in visual computing provides a rich source of computational models to represent, manipulate, and generate visual data. On one side, computer vision and signal processing research provide effective techniques to extract meaningful information from the raw captured data. On the other side, computer graphics research provides techniques to generate realistic-looking imagery. Combining the two fronts, many essential foundations for successful systems that synthesize visual content have been established.

2.1.1 Classical Image-Based and Video-Based Rendering

Photorealistic rendering is the ultimate goal in computer graphics. Traditional computer graphics rely on detailed scene descriptions and sophisticated rendering pipelines to generate realistic content. Scene construction is particularly challenging. It is very labor-intensive to derive detailed 3D scene description with realistic geometries, materials, and lighting information. It is is also extremely challenging to manually model sophisticated phenonmena that involve dynamic and non-rigid content. In addition, rendering is also a fundamental challenge. To achieve a high level of realism, computationally expensive physically based rendering processes on high-end graphics processing units are required to simulate global illumination effects.

Early breakthroughs in enabling practical realistic image and video synthesis systems came from the idea of image-/video-based modelling and rendering [157] in the context of the novel-view synthesis problems. Observing that photo-realism is challenging to achieve with conventional 3D graphics pipelines, researchers have explored methods that can directly leverage real images in the rendering process. Unlike the traditional computer graphics rendering pipeline where 3D scene geometries and materials must be known, image-based rendering (IBR) techniques render novel views directly from input images, which are by-definition photorealistic.

In early IBR systems such as Lightfield Rendering [90] and Lumigraph Rendering [35], special camera arrays were constructed to sample the plenoptic function of the scene [157]. Leveraging the known relation between the cameras in the array, it was possible to synthesize any novel view along a certain range of viewpoints by selecting and interpolating appropriate pixels in the original cameras. Lumigraph Rendering was later extended to handling unstructured camera array by incorporating coarse geometric information about the scene [20]. By leveraging different representations

cleverly designed to encode simplified geometric information, modern IBR methods can support view synthesis from a sparse set of input views [37, 73, 154].

The success in novel view synthesis was also extended to the video domain. Multi-camera systems and dedicated multi-view analysis techniques have been developed to enable free-viewpoint videos in which novel views can be synthesized for any moment in the captured videos [157]. Later on, the commercial successes of the *Bullet-Time effect* [1] [2] brought attention and interest to the possibility to manipulate not only the viewpoints but also the temporal dynamics information such as timing and motion. The original systems for bullet-time effect require special camera setups which dictate the novel views and speed that can be synthesized. To enable more flexible systems, motion information needs to be explicitly extracted from the original captured videos. For this, modelling the motion information in videos has become increasingly critical.

2.1.2 Flow-Based Video Synthesis

Human visual systems are sensitive to motion. Foundational works in early vision research have established the critical connection between the optical flow information from image sequences and how the motion information is perceived when viewing those sequences [181, 199]. The patterns of the extracted optical flows and their variations over time were found to be critical cues for human visual systems to reconstruct self-motion, object motion, time-to-contact, and scene layout [49]. Those findings have been put into computational models that form the foundations for important machine vision tasks such as object tracking [24, 215], Simultaneous Localization and Mapping (SLAM) [23] and video coding [178, 204].

[1] https://www.newworlddesigns.co.uk/bullet-time-photography-what-is-it-and-how-to-get-started/

[2] https://reframe.sussex.ac.uk/post-cinema/3-2-sudmann/

Modern video synthesis systems also take advantage of such connections to achieve synthesis results with plausible temporal dynamics. Regularizing the flow maps of the synthesized videos is a widely used strategy to encourage temporal coherence in the synthesized results. Such regularization is often achieved by enforcing the flow map of the synthesized videos to that of a reference videos [27, 28, 74, 197]. Alternatively, smoothness constraints such as total variation [19, 46, 87, 140] can be incorporated into the objective function to regularize the flow map of the synthesized video [108, 212].

Smoothing out the (sparse) flow trajectories and re-synthesizing the video to respect the adjusted flow information has also been successfully used to stabilize videos [99, 102]. By analyzing the optical flow information over the whole video clip and determining the moment in time at which the flows are similar at each pixel, Hoppe et al. [94] devised an effective method to synthesize a seamless loop from a short video clip. Flow-based view synthesis techniques have also been successfully employed for video frame interpolation. Using flow to represent the motion at each pixel, it is possible to generate the slow-motion effect by synthesizing the pixels at the intermediate positions along each flow vector to generate the interpolated frames between each pair of original frames. [8, 201, 218].

2.1.3 Patch-Based Video Synthesis

An important insight that formed the foundation for many modern visual synthesis tasks was that novel content can be synthesized by properly re-sampling the source content. In Video Textures [152], Schödl et al. determined similar frames at different times and used them to produce a new seamless video. Agarwala et al. applied the same technique on dynamic regions of the video captured with a panning camera while stitching the static part to enable seamlessly looping video panorama [3].

More recent works have extended this idea by considering much more local scope, the small image patches. By sampling local image patches while respecting their local coherence, Lin et al. showed that it was possible to perform texture synthesis from a small sample texture image [93]. By coupling the patch re-sampling strategy with the patch-recurrence properties of natural image patches [226, 227], patch-based methods have been successful in image synthesis tasks such as image enhancement [119, 120, 227] and image retargetting [11, 96].

The success of the patch-based synthesis methodology has also been adapted to the video domain. To capture temporal dynamics information, 3D spatiotemporal patches were used. Wexler et al. replicated the success of patch-based synthesis techniques to perform video completion [202]. That idea of re-sampling videos with spatiotemporal patches were also successfully employed by Shahar et al. to support super-resolution and video frame interpolation [155]. Recently, Haim et al. revisited the idea of patch-based synthesis and introduced an effective system that could synthesize different plausible variations of a source video by sampling and assembling patches from the input video [66].

2.1.4 Phase-Based Video Synthesis

Interpreting image and video data from a signal processing perspective, researchers have revealed that the spatial frequency, temporal frequency, and speed of image motion are highly related [49, 181]. Such observations allowed many motion analysis tasks to be effectively performed in the frequency domain. In particular, the wavelet-based steerable pyramid [160] has been invented to represent image data in the frequency domain using the basis functions that resemble the sinusoids windowed by a Gaussian envelope. This representation was shown effective for analyzing motion in videos [52, 54, 59, 159].

Going beyond motion analysis, Wadhwa et al. cleverly leveraged the relation between motion in the video and the phase information in steerable pyramids to devise a video synthesis method that could perform temporal processing without explicit optical flow computation [196]. Just as the phase shifts in sinusoidal functions encode their translation, the phase variations in the steerable pyramid can be used to control local motions in the image domain. Such phase-based representation of motion was exploited by Wadhwa et al. to enable re-synthesizing videos with modified temporal dynamics content such as motion magnification and motion denoising [193, 196].

In followed-up works, the phase-based motion processing strategy was further extended to support other video synthesis tasks. Meyer et al. improved upon the original phase-based motion processing formulation to address the phase-wrapping ambiguity to allow for synthesizing larger motions, making it applicable to video frame interpolation tasks [117]. Phase-based motion representation has also been applied to transfer motion from a reference video to an image to support image animation [136].

2.2 Learning to Synthesize Video Content

Computational models developed in traditional graphics and vision works reflect the researchers' domain knowledge and sophisticated understanding about visual data. The successful applications mentioned in the last sections have demonstrated that such scientific knowledge is invaluable. However, as good as they were, the discovered knowledge was likely still incomplete and often contains simplified assumptions. Importantly, they were often implemented as heuristics when used for constructing computational models. That heuristic nature is arguably an important limitation of traditional graphics- and vision-based approaches in visual synthesis. Relying on hand-crated heuristics made the resulting methods inflexible in adapting to challenging scenarios that required handling complex phenomena such as non-rigid scenes and

occlusion. That often caused artifacts and distortions, especially when the simplifying assumptions were violated.

Data-driven approach offer an attractive direction to address that challenge. Data-driven techniques can complement the models by making them adaptable to data rather than constrained to rigid sets of simplification assumptions and heuristics. Exploring machine learning techniques, therefore, has recently become a major theme in modern visual computing research.

2.2.1 Deep Neural Networks for Visual Synthesis

In recent years, the advances in deep learning research have enabled many technical breakthroughs, especially in visual data modelling [69, 187]. The availability of large-scale datasets, along with the ever-increasing computing power, makes it possible to train deep neural networks with unprecedented levels of complexity. With modern deep learning methodologies, it is possible to learn not only the top-level predictive models but also the relevant features from raw data.

Deep learning methods have brought tremendous successes to image synthesis tasks. Deep image generation architectures are now able to generate image content with unprecedented level of realism, thanks to the advanced generative modelling techniques such as GAN [82, 134] and Diffusion models [40, 71, 166]. In unconditional image synthesis settings, state-of-the-art models [18, 40, 71, 82, 166] can now generate natural images with the level of realism that can trick human perception [2]

Those models have also been successfully adapted to conditional image synthesis settings, enabling controllability in the image generation process. Isola et al. investigate conditional adversarial networks [123] as a generic approach to synthesizing images from semantic maps or edge maps [76]. Reed et al. [143] further extend conditional GANs to generate natural images based on textual descriptions. Building on the

power of diffusion models in high-fidelity image synthesis, the text-to-image generation is advanced significantly by the recent effort of DALL-E 2 [139] and Imagen [147].

Despite the immense success in synthesizing images with deep learning models, translating those successes to video modelling was challenging. Video is a separate form of media in its own, not simply a collection of independent images. Because of that, applying image-based models frame-by-frame to video synthesis usually leads to sub-optimal results [16].

Early efforts to extend well-known deep image models to handle video data have proven highly non-trivial. Inspired by the success of the 2D convolutional neural network (CNN) in image synthesis architectures, many follow-up works have developed video-based synthesis techniques on top of 3D-CNN models [34, 148, 191]. Recent efforts in video modelling with deep learning have been devoted to explicitly encourage motion modelling in the video synthesis process. Denton *et al.* leveraged the recurrent neural network (RNN) architecture to model dynamic progression in videos to support frame prediction [39]. In another direction, Tulyakov et al. extended the GAN-based approach into a two-stream architecture and devise a training strategy to explicitly encourage the model to learn disentangled representation for motion and appearance [185].

Those efforts, while showing early promising results, do not match the level of quality obtained in the image domain. Such difficulty demonstrated that learning the representation for videos is challenging for the models designed originally for images without explicitly incorporating the specific natures of videos. In other words, such models lack the inductive bias to aid the learning of meaningful representation from videos.

The most successful strategy for synthesizing high-quality video results remains applying high-quality image-based synthesis for each frame and adjusting the results

to optimize the temporal coherence using optical flow [74, 97, 108]. This strategy, however, is only applicable to the the problem setting where the reference optical flow can be obtained, such as video denoising or super-resolution. It is not applicable for more general video synthesis problems such as video frame interpolation and video generation.

2.2.2 Inductive Biases in Representation Learning

Inductive bias is an essential concept in machine learning. Its role has been shown to be critical in the success of modern deep learning systems. Inductive biases refer to the assumptions incorporated with the learning algorithms to generalize a set of training data. This mechanism encourages the learning algorithms to prioritize solutions with specific properties. It helps induce the learning models to effectively capture the structure in the data and, from that, generalize better to unseen data.

Spatial inductive bias is perhaps one of the best-known types of inductive biases that have enjoyed great success in deep learning. Spatial inductive bias is particularly helpful in the Convolutional Neural Network (CNN) architecture, which was designed to exploit the spatial equivariance nature of vision data. It is highly effective in processing and synthesizing image data [189].

Structured perception and relational reasoning is another type of inductive bias that has proven highly useful, especially in reinforcement learning settings where the learning agent must encode the meaningful structure of the environment [14]. Introducing structured relation information into deep RL architectures makes it possible for the learning agents to learn interpretable representations to improve their prediction accuracy, sample complexity, and ability to generalise [14].

Recently, 3D geometry has been incorporated into image synthesis architecture to serve as a type of structural inductive bias. In [125], Nguyen-Phuoc et al. made

the important finding that shaping the learnable features of the GAN model into the 3D volume that can be freely transformed during training is sufficient to induce the model to generate and manipulate images in a multi-view consistent manner. Representing the learned 3D scene, coupled with volumetric rendering for synthesis, is also the key idea behind the success of the Neural Radiance Fields framework for novel view synthesis [12, 122, 124] as well as the state-of-the-art 3D-aware GAN models [25, 38, 64].

Inductive biases are not easy to define and incorporate. Good inductive biases should represent the knowledge that is universally true, i.e. the correct priors. In this thesis, the main direction I set out to explore is to look for and design inductive biases from the insights and understanding of video data modelling already exists in the rich literature of traditional visual computing research. I study that direction in this thesis with two case-studies: a deep learning model that successfully extends 2D convolution-based neural networks to enable high-quality frame interpolation results; and a simple technique to extend image-based neural implicit representation to video data that also enables the flexibility of temporal dynamics manipulation.

3 Adaptive Convolution for Video Frame Interpolation

Video frame interpolation is a classic problem in computer graphics and computer vision. The goal is to synthesize the intermediate frames between each pair of consecutive original input frames. Before 2016 when this research started, high-quality frame interpolation results could not be obtained with contemporary deep learning approaches, despite their tremendous successes in other image analysis and synthesis domains. Existing deep-learning-based video frame interpolation methods followed the direct prediction approach. Standard image-based convolutional neural network (CNN) or recurrent neural network (RNN) architectures were extended to take a pair of frames as input and directly predict the intermediate frames [39, 104]. Such direct prediction approach failed to generate high-quality results, often with low-resolution and severe visual distortion. Hypothesizing that more explicit inductive biases were needed to enable plausible frame synthesis results, I started exploring the idea of incorporating domain knowledge as inductive bias into deep neural network learning for video frame interpolation.

The rich literature on video frame interpolation has established that video frame interpolation fundamentally involves two processes: motion estimation and pixel synthesis. As an alternative to the direct prediction approach, this chapter presents AdaConv, a robust video frame interpolation method that explicitly incorporated these two steps into the model. Unlike conventional optimization-based approaches to video frame interpolation that implement those two steps as two separate processes,

the proposed method combines both steps into a single process implemented as a novel differentiable layer into the CNN architecture. Specifically, AdaConv employs a deep fully-convolutional neural network to estimate a spatially-adaptive convolution kernel for each pixel. This method considers pixel synthesis for the interpolated frame as a local convolution over two input frames. The convolution kernel captures both the local motion between the input frames and the coefficients for pixel synthesis. The resulting deep neural network can be directly trained end-to-end using widely available video data without difficult-to-obtain ground-truth data like optical flow. Experiments on a wide variety of real-world videos show that the formulation of video interpolation as a single convolution process allows our method to gracefully handle challenges like occlusion, blur, and abrupt brightness change and enables high-quality video frame interpolation.

This work was the result of the collaboration with Simon Niklaus and Professor Feng Liu. We published our work at the IEEE Conference of Computer Vision and Pattern Recognition (CVPR) 2017 for which Simon and I contributed as the co-first authors [127]. The writing of this chapter was adapted from the published paper. The use of "*we*", "*our*", and "*ours*" throughout this chapter refer to the authors of the published paper (Long Mai, Simon Niklaus, and Feng Liu). In particular, my own contribution in the paper is the idea of modelling frame interpolation as adaptive convolution, and the design of the neural network to realize that idea.

3.1 Overview

Frame interpolation is a classic computer vision problem and is important for applications like novel view interpolation and frame rate conversion [117]. Traditional frame interpolation methods have two steps: motion estimation, usually optical flow, and pixel synthesis [8]. Optical flow is often difficult to estimate in the regions suffering

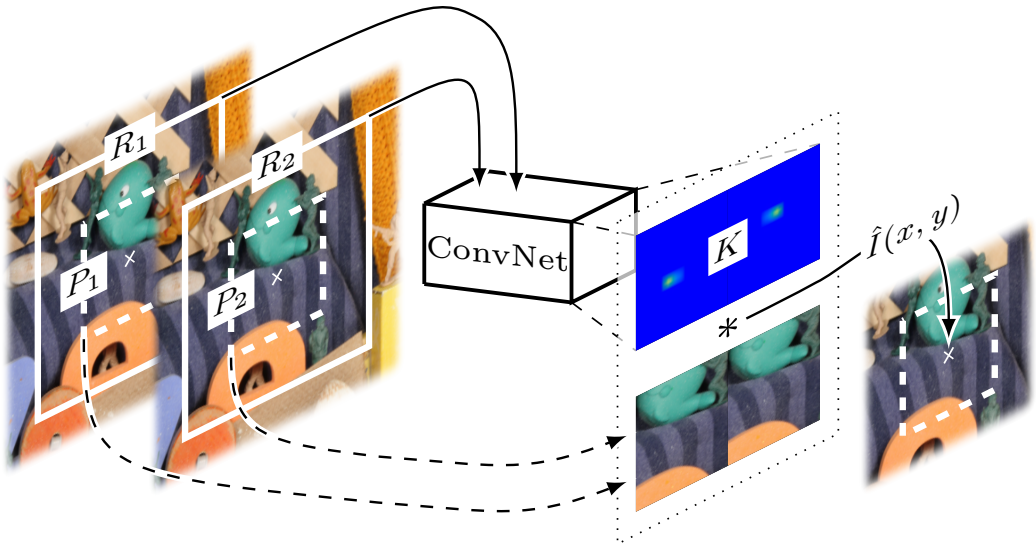


Figure 3.1: Pixel interpolation by convolution. For each output pixel (x, y), our method estimates a convolution kernel K and uses it to convolve with patches P_1 and P_2 centered at (x, y) in the input frames to produce its color $\hat{I}(x, y)$. This adaptive convolution formulation effectively combines motion estimation and pixel synthesis into a single operation for synthesizing each pixel.

from occlusion, blur, and abrupt brightness change. Flow-based pixel synthesis cannot reliably handle the occlusion problem. Failure of any of these two steps will lead to noticeable artifacts in interpolated video frames.

This chapter presents a robust video frame interpolation method that achieves frame interpolation using a deep convolutional neural network without explicitly dividing it into separate steps. Our method considers pixel interpolation as convolution over corresponding image patches in the two input video frames, and estimates the spatially-adaptive convolutional kernel using a deep fully convolutional neural network. Specifically, for a pixel (x, y) in the interpolated frame, this deep neural network takes two receptive field patches R_1 and R_2 centered at that pixel as input and estimates a convolution kernel K. This convolution kernel is used to convolve with the input

patches P_1 and P_2 to synthesize the output pixel, as illustrated in Figure 3.1.

An important aspect of our method is the formulation of pixel interpolation as convolution over pixel patches instead of relying on optical flow. This convolution formulation unifies motion estimation and pixel synthesis into a single procedure. It enables us to design a deep fully convolutional neural network for video frame interpolation without dividing interpolation into separate steps. This formulation is also more flexible than those based on optical flow and can better handle challenging scenarios for frame interpolation. Furthermore, our neural network is able to estimate edge-aware convolution kernels that lead to sharp results.

The main contribution of this work is a robust video frame interpolation method that employs a fully deep convolutional neural network to produce high-quality video interpolation results. This method has a few advantages. First, since it models video interpolation as a single process, it is able to make proper trade-offs among competing constraints and thus can provide a robust interpolation approach. Second, this frame interpolation deep convolutional neural network can be directly trained end-to-end using widely available video data, without any difficult-to-obtain ground truth data like optical flow. Third, as demonstrated in our experiments, our method can generate high-quality frame interpolation results for challenging videos such as those with occlusion, blurring artifacts, and abrupt brightness change.

3.2 Related Work

Frame interpolation for video is one of the basic computer vision and video processing technologies. It is a special case of image-based rendering where middle frames are interpolated from temporally neighboring frames. Good surveys on image-based rendering are available [80, 170, 219]. This section focuses on research that is

`http://graphics.cs.pdx.edu/project/adaconv`

specific to video frame interpolation and our work.

Most existing frame interpolation methods estimate dense motion between two consecutive input frames using stereo matching or optical flow algorithms and then interpolate one or more middle frames according to the estimated dense correspondences [8, 201, 218]. Different from these methods, Mahajan *et al.* developed a moving gradient method that estimates paths in input images, copies proper gradients to each pixel in the frame to be interpolated and then synthesizes the interpolated frame via Poisson reconstruction [106]. The performance of all the above methods depends on the quality of dense correspondence estimation and special care needs to be taken to handle issues like occlusion during the late image synthesis step.

As an alternative to explicit motion estimation-based methods, phase-based methods have recently been shown promising for video processing. These methods encode motion in the phase difference between input frames and manipulate the phase information for applications like motion magnification [193] and view expansion [41]. Meyer *et al.* further extended these approaches to accommodate large motion by propagating phase information across oriented multi-scale pyramid levels using a bounded shift correction strategy [117]. This phase-based interpolation method can generate impressive video interpolation results and handle challenging scenarios gracefully; however, further improvement is still required to better preserve high-frequency detail in the video with large inter-frame changes.

Our work is inspired by the success of deep learning in solving not only difficult visual understanding problems [60, 70, 81, 88, 153, 161, 168, 171, 205, 216, 222] but also other computer vision problems like optical flow estimation [44, 55, 65, 179, 182, 200], style transfer [47, 58, 78, 91, 188], and image enhancement [21, 42, 43, 167, 169, 208, 210, 221, 224]. Our method is particularly relevant to the recent deep learning algorithms for view synthesis [45, 53, 79, 89, 176, 214, 223]. Dosovitskiy *et al.* [45],

Kulkarni *et al.* [89], Yang *et al.* [214], and Tatarchenko *et al.* [176] developed deep learning algorithms that can render unseen views from input images. These algorithms work on objects, such as chairs and faces, and are not designed for frame interpolation for videos of general scenes.

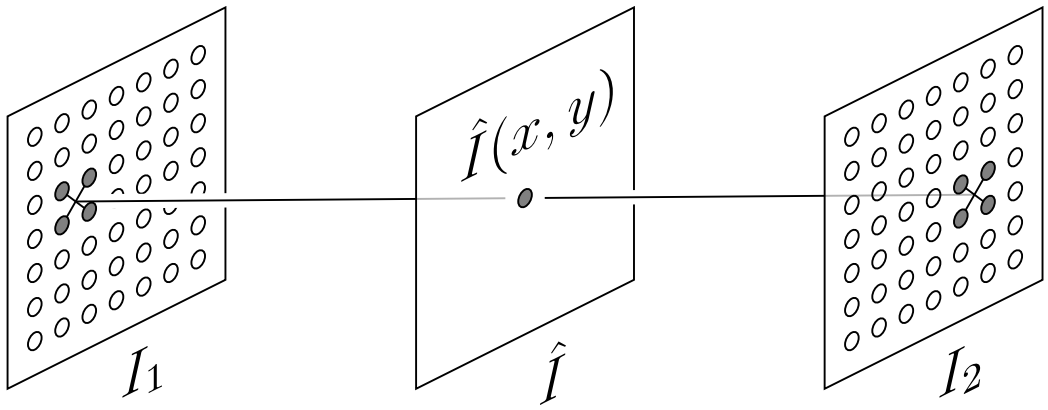
Recently, Flynn *et al.* developed a deep convolutional neural network method for synthesizing novel natural images from posed real-world input images. Their method projects input images onto multiple depth planes and combines colors at these depth planes to create a novel view [53]. Kalantari *et al.* provided a deep learning-based view synthesis algorithm for view expansion for light field imaging. They break novel synthesis into two components: disparity and color estimation, and accordingly use two sequential convolutional neural networks to model these two components. These two neural networks are trained simultaneously [79]. Long *et al.* interpolate frames as an intermediate step for image matching [104]. However, their interpolated frames tend to be blurry. Zhou *et al.* observed that the visual appearance of different views of the same instance is highly correlated, and designed a deep learning algorithm to predict appearance flows that are used to select proper pixels in the input views to synthesize a novel view [223]. Given multiple input views, their method can interpolate a novel view by warping individual input views using the corresponding appearance flows and then properly combining them together. Like these methods, our deep learning algorithm can also be trained end-to-end using videos directly. Compared to these methods, our method is dedicated to video frame interpolation. More importantly, our method estimates convolution kernels that capture both the motion and interpolation coefficients, and uses these kernels to directly convolve with input images to synthesize a middle video frame. Our method does not need to project input images onto multiple depth planes or explicitly estimate disparities or appearance flows to warp input images and then combine them together. Our experiments show that our formulation of

frame interpolation as a single convolution step allows our method to robustly handle challenging cases. Finally, the idea of using convolution for image synthesis has also been explored in the very recent work for frame extrapolation [50, 77, 213].

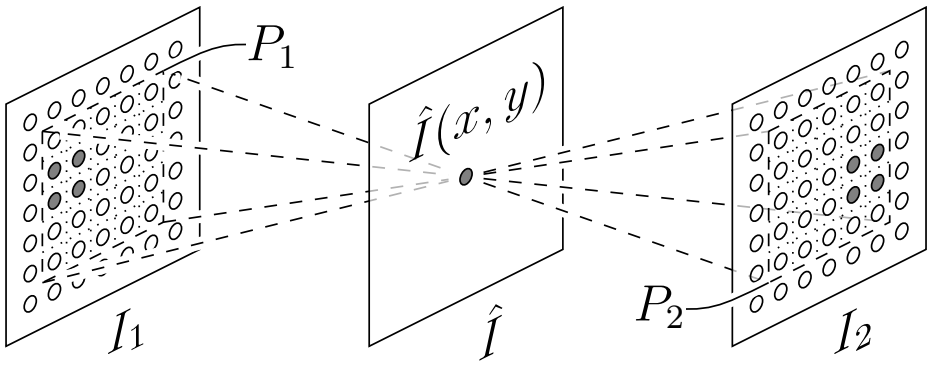
3.3 Method

Given two video frames I_1 and I_2, our method aims to interpolate a frame $\hat{I}$ temporally in the middle of the two input frames. Traditional interpolation methods estimate the color of a pixel $\hat{I}(x, y)$ in the interpolated frame in two steps: dense motion estimation, typically through optical flow, and pixel interpolation. For instance, we can find for pixel (x, y) its corresponding pixels (x_1, y_1) in I_1 and (x_2, y_2) in I_2 and then interpolate the color from these corresponding pixels. Often this step also involves re-sampling images I_1 and I_2 to obtain the corresponding values $I_1(x_1, y_1)$ and $I_2(x_2, y_2)$ to produce a high-quality interpolation result, especially when (x_1, y_1) and (x_2, y_2) are not integer locations, as illustrated in Figure 3.2 (a). This two-step approach can be compromised when optical flow is not reliable due to occlusion, motion blur, and lack of texture. Also, rounding the coordinates to find the color for $I_1(x_1, y_1)$ and $I_2(x_2, y_2)$ is prone to aliasing while re-sampling with a fixed kernel sometimes cannot preserve sharp edges well. Advanced re-sampling methods exist and can be used for edge-preserving re-sampling, which, however, requires high-quality optical flow estimation.

Our solution is to combine motion estimation and pixel synthesis into a single step and formulate pixel interpolation as a local convolution over patches in the input images I_1 and I_2. As shown in Figure 3.2 (b), the color of pixel (x, y) in the target image to be interpolated can be obtained by convolving a proper kernel K over input patches $P_1(x, y)$ and $P_2(x, y)$ that are also centered at (x, y) in the respective input images. The convolutional kernel K captures both motion and re-sampling coefficients for pixel



(a) Interpolation by motion estimation and color interpolation



(b) Interpolation by convolution

Figure 3.2: Interpolation by convolution. (a): a two-step approach first estimates motion between two frames and then interpolates the pixel color based on the motion. (b): our method directly estimates a convolution kernel and uses it to convolve the two frames to interpolate the pixel color. In this way, our method combines both steps of coventional frame interpolation techniques into a single operation under a unified convolution formulation.

synthesis. This formulation of pixel interpolation as convolution has a few advantages. First of all, the combination of motion estimation and pixel synthesis into a single step provides a more robust solution than the two-step procedure. Second, the convolution kernel provides flexibility to account for and address difficult cases like occlusion. For example, optical flow estimation in an occlusion region is a fundamentally difficult problem, which makes it difficult for a typical two-step approach to proceed. Extra steps based on heuristics, such as flow interpolation, must be taken. Our work provides a data-driven approach to directly estimate the convolution kernel that can produce visually plausible interpolation results for an occluded region. Third, if properly estimated, this convolution formulation can seamlessly integrate advanced re-sampling techniques like edge-aware filtering to provide sharp interpolation results.

Estimating proper convolution kernels is essential for our method. Encouraged by the success of using deep learning algorithms for optical flow estimation [44, 55, 65, 179, 182, 200] and image synthesis [53, 79, 223], we develop a deep convolutional neural network method to estimate a proper convolutional kernel to synthesize each output pixel in the interpolated images. The convolutional kernels for individual pixels vary according to the local motion and image structure to provide high-quality interpolation results. Below we describe our deep neural network for kernel estimation and then discuss implementation details.

3.3.1 Convolution kernel estimation

We design a fully convolutional neural network to estimate the convolution kernels for individual output pixels. The architecture of our neural network is detailed in Table 3.1. Specifically, to estimate the convolutional kernel K for the output pixel (x, y), our neural network takes receptive field patches $R_1(x, y)$ and $R_2(x, y)$ as input. $R_1(x, y)$ and $R_2(x, y)$ are both centered at (x, y) in the respective input images. The

type	BN	ReLU	size	stride	output
input	-	-	-	-	$6 \times 79 \times 79$
conv	✓	✓	7×7	1×1	$32 \times 73 \times 73$
down-conv	-	✓	2×2	2×2	$32 \times 36 \times 36$
conv	✓	✓	5×5	1×1	$64 \times 32 \times 32$
down-conv	-	✓	2×2	2×2	$64 \times 16 \times 16$
conv	✓	✓	5×5	1×1	$128 \times 12 \times 12$
down-conv	-	✓	2×2	2×2	$128 \times 6 \times 6$
conv	✓	✓	3×3	1×1	$256 \times 4 \times 4$
conv	-	✓	4×4	1×1	$2048 \times 1 \times 1$
conv	-	-	1×1	1×1	$3362 \times 1 \times 1$
spatial softmax	-	-	-	-	$3362 \times 1 \times 1$
output	-	-	-	-	$41 \times 82 \times 1 \times 1$

Table 3.1: The convolutional neural network architecture. It makes use of Batch Normalization (BN) [75] as well as Rectified Linear Units (ReLU). Note that the output only reshapes the result without altering its value.

patches P_1 and P_2 that the output kernel will convolve in order to produce the color for the output pixel (x, y) are co-centered at the same locations as these receptive fields, but with a smaller size, as illustrated in Figure 3.1. We use a larger receptive field than the patch to better handle the aperture problem in motion estimation. In our implementation, the default receptive field size is 79×79 pixels. The convolution patch size is 41×41 and the kernel size is 41×82 as it is used to convolve with two patches. Our method applies the same convolution kernel to each of the three color channels.

As shown in Table 3.1, our convolutional neural network consists of several convo-

color loss | color loss + gradient loss

Figure 3.3: Effect of using an additional gradient loss. (a) Using the color loss alone can lead to blurry results. (b) Combining the color loss and gradient loss enables our method to produce sharper interpolation results (most noticeable in front-wheel region of the middle car).

lutional layers as well as down-convolutions as alternatives to max-pooling layers. We use Rectified Linear Units as activation functions and Batch Normalization [75] for regularization. We employ no further techniques for regularization since our neural network can be trained end-to-end using widely available video data, which provides a sufficiently large training dataset. We are also able to make use of data augmentation extensively, by horizontally and vertically flipping the training samples as well as reversing their order. Our neural network is fully convolutional. Therefore, it is not restricted to a fixed-size input and we are, as detailed in Section 3.3.3, able to use a shift-and-stitch technique [61, 105, 153] to produce kernels for multiple pixels simultaneously to speed-up our method.

A critical constraint is that the coefficients of the output convolution kernel should be non-negative and sum up to one. Therefore, we connect the final convolutional layer to a spatial softmax layer to output the convolution kernel, which implicitly meets this important constraint.

3.3.1.1 Loss function

For clarity, we first define notations. The i^{th} training example consists of two input receptive field patches $R_{i,1}$ and $R_{i,2}$ centered at (x_i, y_i), the corresponding input patches $P_{i,1}$ and $P_{i,2}$ that are smaller than the receptive field patches and also centered at the same location, the ground-truth color $\tilde{C}_i$ and the ground-truth gradient $\tilde{G}_i$ at (x_i, y_i) in the interpolated frame. The ground-truth gradient $\tilde{G}_i^k$ is obtained by applying a gradient computation procedure on the ground-truth target frame $\tilde{C}_i$. For simplicity, we omit the (x_i, y_i) in our definition of the loss functions.

One possible loss function of our deep convolutional neural network can be the difference between the interpolated pixel color and the ground-truth color as follows.

$$E_c = \sum_i \|[P_{i,1}\ P_{i,2}] * K_i - \tilde{C}_i\|_1 \tag{3.1}$$

where subscript i indicates the i^{th} training example and K_i is the convolution kernel output by our neural network. Our experiments show that this color loss alone, even using ℓ_1 norm, can lead to blurry results, as shown in Figure 3.3. This blurriness problem was also reported in some recent work [104, 111, 142]. Mathieu *et al.* showed that this blurriness problem can be alleviated by incorporating image gradients in the loss function [111]. This is difficult within our pixel-wise interpolation approach, since the image gradient cannot be directly calculated from a single pixel. Since differentiation is also a convolution, assuming that kernels are locally equivalent, we solve this problem by using the associative property of convolution: we first compute the gradient of input patches and then perform convolution with the estimated kernel, which will result in the gradient of the interpolated image at the pixel of interest. As a pixel (x, y) has eight immediate neighboring pixels, we compute eight versions of gradients using finite difference and incorporate all of them into our gradient loss

function.

$$E_g = \sum_i \sum_{k=1}^{8} \|[G_{i,1}^k \; G_{i,2}^k] * K_i - \tilde{G}_i^k\|_1 \tag{3.2}$$

where k denotes one of the eight ways we compute the gradient. $G_{i,1}^k$ and $G_{i,2}^k$ are the gradients of the input patches $P_{i,1}$ and $P_{i,2}$, and $\tilde{G}_i^k$ is the ground-truth gradient. We combine the above color and gradient loss as our final loss $E_c + \lambda \cdot E_g$. We found that $\lambda = 1$ works well and used it. As shown in Figure 3.3, this color plus gradient loss enables our method to produce sharper interpolation results.

3.3.2 Training

We derived our training dataset from an online video collection, as detailed later on in this section. To train our neural network, we initialize its parameters using the Xavier initialization approach [62] and then use AdaMax [83] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 0.001 and 128 samples per mini-batch to minimize the loss function.

3.3.2.1 Training dataset

Our loss function is purely based on the ground truth video frame and does not need any other ground truth information like optical flow. Therefore, we can make use of videos that are widely available online to train our neural network. To make it easy to reproduce our results, we use publicly available videos from Flickr with a Creative Commons license. We downloaded $3,000$ videos using keywords, such as "driving", "dancing", "surfing", "riding", and "skiing", which yield a diverse selection. We scaled the downloaded videos to a fixed size of 1280×720 pixels. We removed interlaced videos that sometimes have a lower quality than the videos with the progressive-scan format.

To generate the training samples, we group all the frames in each of the remaining videos into triple-frame groups, each containing three consecutive frames in a video. We then randomly pick a pixel in each triple-frame group and extract a triple-patch group centered at that pixel from the video frames. To facilitate data augmentation, the patches are selected to be larger than the receptive-field patches required by the neural network. The patch size in our training dataset is 150×150 pixels. To avoid including a large number of samples with no or little motion, we estimate the optical flow between patches from the first and last frame in the triple-frame group [174] and compute the mean flow magnitude. We then sample $500,000$ triple-patch groups without replacement according to the flow magnitude: a patch group with larger motion is more likely to be chosen than the one with smaller motion. In this way, our training set includes samples with a wide range of motion while avoiding being dominated by patches with little motion. Since some videos consist of many shots, we compute the color histogram between patches to detect shot boundaries and remove the groups across the shot boundaries. Furthermore, samples with little texture are also not very useful to train our neural network. We therefore compute the entropy of patches in each sample and finally select the $250,000$ triple-patch groups with the largest entropy to form the training dataset. In this training dataset, about 10 percent of the pixels have an estimated flow magnitude of at least 20 pixels. The average magnitude of the largest five percent is approximately 25 pixels and the largest magnitude is 38 pixels.

We perform data augmentation on the fly during training. The receptive-field size required for the neural network is 79×79, which is smaller than the patch size in the training samples. Therefore, during the training, we randomly crop the receptive field patch from each training sample. We furthermore randomly flip the samples horizontally as well as vertically and randomly swap their temporal order. This forces

the optical flow within the samples to be distributed symmetrically so that the neural network is not biased towards a certain direction.

3.3.3 Implementation details

We used Torch [36] to implemented our neural network. Below we describe some important details.

3.3.3.1 Shift-and-stitch implementation

A straightforward way to apply our neural network to frame interpolation is to estimate the convolution kernel and synthesize the interpolated pixel one by one. This pixel-wise application of our neural network will unnecessarily perform redundant computations when passing two neighboring pairs of patches through the neural network to estimate the convolution kernels for two corresponding pixels. Our implementation employs the shift-and-stitch approach to address this problem to speed our system up [61, 105, 153].

Specifically, as our neural network is fully convolutional and does not require a fixed-size input, it can compute kernels for more than one output pixels at once by supplying a larger input than what is required to produce one kernel. This can mitigate the issue of redundant computations. The output pixels that are obtained in this way are however not adjacent and are instead sparsely distributed. We employ the shift-and-stitch [61, 105, 153] approach in which slightly shifted versions of the same input are used. This approach returns sparse results that can be combined to form the dense representation of the interpolated frame.

Considering a frame with size 1280×720, a pixel-wise implementation of our neural network would require 921,600 forward passes through our neural network. The shift-and-stitch implementation of our neural network only requires 64 forward passes for the 64 differently shifted versions of the input to cope with the downscaling by the

three down-convolutions. Compared to the pixel-wise implementation that takes 104 seconds per frame on an Nvidia Titan X, the shift-and-stitch implementation only takes 9 seconds.

3.3.3.2 Boundary handling

Due to the receptive field of the network as well as the size of the convolution kernel, we need to pad the input frames to synthesize boundary pixels for the interpolated frame. In our implementation, we adopt zero-padding. Our experiments show that this approach usually works well and does not introduce noticeable artifacts.

3.3.3.3 Hyper-parameter selection

The convolution kernel size and the receptive field size are two important hyper-parameters of our deep neural network. In theory, the convolution kernel, as shown in Figure 3.2, must be larger than the pixel motion between two frames in order to capture the motion (implicitly) to produce a good interpolation result. To make our neural network robust against large motion, we tend to choose a large kernel. On the other hand, a large kernel involves a large number of values to be estimated, which increases the complexity of our neural network. We choose to select a convolution kernel that is large enough to capture the largest motion in the training dataset, which is 38 pixels. Particularly, the convolution kernel size in our system is 41×82 that will be applied to two 41×41 patches as illustrated in Figure 3.1. We make this kernel a few pixels larger than 38 pixels to provide pixel support for re-sampling, which our method does not explicitly perform, but is captured in the kernel.

As discussed earlier, the receptive field is larger than the convolution kernel to handle the aperture problem well. However, a larger receptive field requires more computation and is less sensitive to the motion. We choose the receptive field using a

Input frame 1 Ours Meyer *et al.* DeepFlow2 FlowNetS MDP-Flow2 Brox *et al.*

Figure 3.4: Qualitative evaluation on blurry videos. Blurry regions are often challenging for optical flow estimation, resulting in noticeable artifacts (Columns 4-7). The phase-based method from Meyer *et al.* [117] can handle blurry regions better (3rd column). Our method tend to be more robust in regions with large motion, such as the right side of the hat in the bottom example.

validation dataset and find that 79×79 achieves a good balance.

3.4 Experiments

We compare our method to state-of-the-art video frame interpolation methods, including the recent phase-based interpolation method [117] and a few optical flow-based methods. The optical flow algorithms in our experiment include MDP-Flow2 [209], which currently produces the lowest interpolation error according to the Middlebury benchmark, the method from Brox *et al.* [19], as well as two recent deep learning based approaches, namely DeepFlow2 [200] and FlowNetS [44]. Following recent frame interpolation work [117], we use the interpolation method from the Middlebury benchmark [8] to synthesize the interpolated frame using the optical flow results. Alternatively, other advanced image-based rendering algorithms [225] can also be used.

For the two deep learning-based optical flow methods, we directly use the trained models from the author websites.

3.4.1 Comparisons

We evaluate our method quantitatively on the Middlebury optical flow benchmark [8]. As reported in Table 3.2, our method performs well on the four examples with real-world scenes. Among the over 100 methods reported in the Middlebury benchmark, our method achieves the best on Evergreen and Basketball, 2nd best on Dumptruck, and 3rd best on Backyard. Our method does not work as well on the other four examples that are either synthetic or of lab scenes, partially because we train our network on videos with real-world scenes. Qualitatively, we find that our method can often create results in challenging regions that are visually more appealing than state-of-the-art methods.

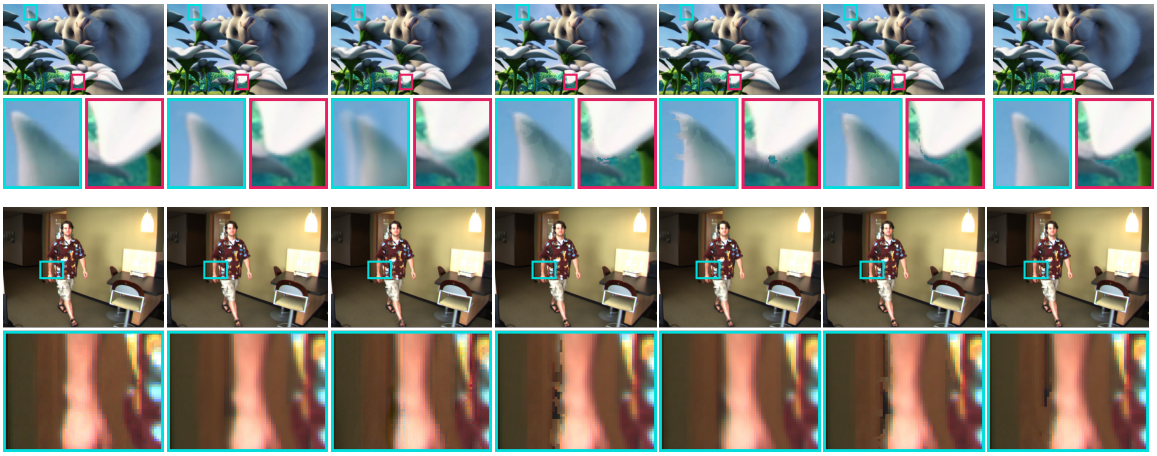
Blur. Figure 3.4 shows two examples where the input videos suffer from out-of-focus blur (top) and motion blur (bottom). Blurry regions are often challenging for optical flow estimation; thus these regions in the interpolated results suffer from noticeable artifacts. Both our method and the phase-based method from Meyer *et al.* [117] can handle blurry regions better while our method produces sharper images, especially in regions with large motion, such as the right side of the hat in the bottom example.

Abrupt brightness change. As shown in Figure 3.5, abrupt brightness change violates the brightness consistency assumption and compromises optical flow estimation, causing artifacts in frame interpolation. For this example, our method and the phase-based method generate more visually appealing interpolation results than flow-based methods.

Occlusion. One of the biggest challenges for optical flow estimation is occlusion. When optical flow is not reliable or unavailable in occluded regions, frame interpolation

Input frames Ours Meyer *et al.* DeepFlow2 FlowNetS MDP-Flow2 Brox *et al.*

Figure 3.5: Qualitative evaluation on video with abrupt brightness change. In general, our method and the phasebased method generate more visually appealing interpolation results than flow-based methods.



Input frame 1 Ours Meyer *et al.* DeepFlow2 FlowNetS MDP-Flow2 Brox *et al.*

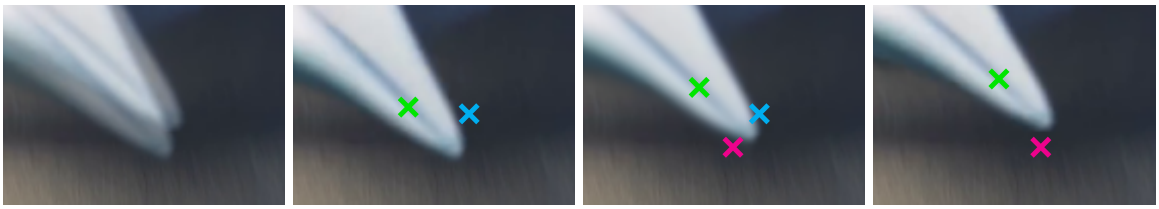
Figure 3.6: Qualitative evaluation with respect to occlusion. Occlusion is one of the biggest challenges for optical flow estimation. Our method adopts a learning approach to obtain proper convolution kernels that lead to visually appealing pixel synthesis results for occluded regions and preserve better object boundaries in the synthesis results.

	Mequ.	Schef.	Urban	Teddy	Backy.	Baske.	Dumpt	Everg.
Ours	3.57	4.34	5.00	6.91	**10.2**	**5.33**	**7.30**	**6.94**
DeepFlow2	2.99	3.88	**3.62**	5.38	11.0	5.83	7.60	7.82
FlowNetS	3.07	4.57	4.01	5.55	11.3	5.99	8.63	7.70
MDP-Flow2	**2.89**	**3.47**	3.66	**5.20**	**10.2**	6.13	7.36	7.75
Brox *et al.*	3.08	3.83	3.93	5.32	10.6	6.60	8.61	7.43

Table 3.2: Evaluation on the Middlebury testing set. We compared different methods in terms of the average interpolation error metric which measures the average absolute difference in per-pixel values between the predicted and the ground-truth target frames.

methods need to fill in holes, such as by interpolating flow from neighboring pixels [8]. Our method adopts a learning approach to obtain proper convolution kernels that lead to visually appealing pixel synthesis results for occluded regions, as shown in Figure 3.6.

To better understand how our method handles occlusion, we examine the convolution kernels of pixels in the occluded regions. As shown in Figure 3.1, a convolution kernel can be divided into two sub-kernels, each of which is used to convolve with one of the two input patches. For the ease of illustration, we compute the centroid of each sub-kernel and mark it using $\mathbf{x}$ in the corresponding input patch to indicate where the output pixel gets its color. Figure 3.7 shows an example where the white leaf moves up from Frame 1 to Frame 2. The occlusion can be seen in the left image that overlays two input frames. For this example, the pixel indicated by the green $\mathbf{x}$ is visible in both frames and our kernel shows that the color of this pixel is interpolated from both frames. In contrast, the pixel indicated by the red $\mathbf{x}$ is visible only in Frame 2. We find that the sum of all the coefficients in the sub-kernel for Frame 1 is almost zero, which indicates Frame 1 does not contribute to this pixel and this pixel gets its color only from Frame 2. Similarly, the pixel indicated by the cyan $\mathbf{x}$ is only visible



Overlay Frame 1 Ours Frame 2

Figure 3.7: Occlusion handling. For the illustration, we compute the centroid of each sub-kernel and mark it using **x** and only visualize it if the sum of kernel value is sufficiently larger than zero. The correspondence between a pixel and its convolution kernel is established by color. The pixel indicated by the green **x** is visible in both frames and our kernel shows that the color of this pixel is interpolated from both frames. On the other hand, the pixel indicated by the cyan **x** is only visible in Frame 1. Our kernel correctly accounts for this occlusion and gets its color from Frame 1 only.

in Frame 1. Our kernel correctly accounts for this occlusion and gets its color from Frame 1 only.

3.4.2 Edge-aware pixel interpolation

In the above, we discussed how our estimated convolution kernels appropriately handle occlusion for frame interpolation. We now examine how these kernels adapt to image features. In Figure 3.8, we sample three pixels in the interpolated image. We show their kernels at the bottom. The correspondence between a pixel and its convolution kernel is established by color. First, for all these kernels, only a very small number of kernel elements have non-zero values. (The use of the spatial softmax layer in our neural network already guarantees that the kernel element values are non-negative and sum up to one.) Furthermore, all these non-zero elements are spatially

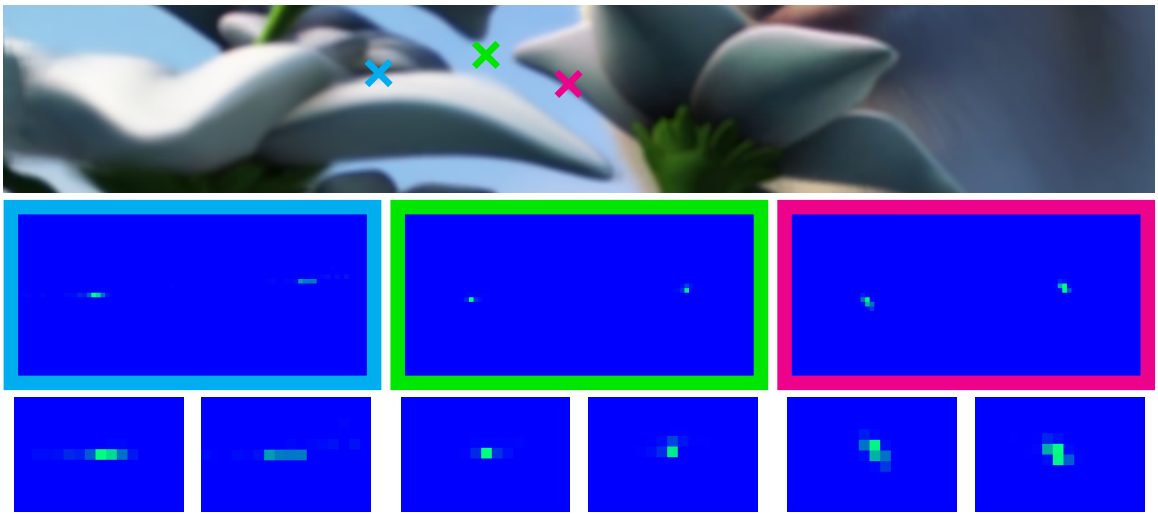


Figure 3.8: Convolution kernels. The third row provides magnified views into the non-zero regions in the kernels in the second row. While our neural network does not explicitly model the frame interpolation procedure, it is able to estimate convolution kernels that enable similar pixel interpolation to the flow-based interpolation methods. More importantly, our kernels are spatially adaptive and edge-aware, such as those for the pixels marked by the red and cyan **x**.

grouped together. This corresponds well with a typical flow-based interpolation method that finds corresponding pixels or their neighborhood in two frames and then interpolate. Second, for a pixel in a flat region such as the one indicated by the green **x**, its kernel only has two elements with significant values. Each of these two kernel elements corresponds to the relevant pixel in the corresponding input frame. This is also consistent with the flow-based interpolation methods although our neural network does not explicitly model the frame interpolation procedure. Third, more interestingly, for pixels along image edges, such as the ones indicated by the red and cyan **x**, the kernels are anisotropic and their orientations align well with the edge directions. This shows that our neural network learns to estimate convolution kernels that enable

Overlaid input | Long *et al.*

Direct | Ours

Figure 3.9: Comparison with direct synthesis. Direct prediction approaches, using both our network architecture and the archicture from Long *et al.* [104], produce blurry results. By allowing for the explicit motion reasoning in the inference process, our model can produce significantly sharper results.

edge-aware pixel interpolation, which is critical to produce sharp interpolation results.

3.4.3 Discussion

Our method is scalable to large images due to its pixel-wise nature. Furthermore, the shift-and-stitch implementation of our neural network allows us to both parallel processing multiple pixels and reduce the redundancy in computing the convolution kernels for these pixels. On a single Nvidia Titan X, this implementation takes about 2.8 seconds with 3.5 gigabytes of memory for a single 640×480 image, and 9.1 seconds with 4.7 gigabytes for 1280×720, and 21.6 seconds with 6.8 gigabytes for 1920×1080.

We experimented with a baseline neural network by modifying our network to directly synthesize pixels. We found that this baseline produces a blurry result for

an example from the Sintel benchmark [22], as shown in Figure 3.9. In the same figure, we furthermore show a comparison with the method from Long *et al.* [104] that performs video frame interpolation as an intermediate step for optical flow estimation. While their result is better than our baseline, it is still not as sharp as ours.

The amount of motion that our method can handle is necessarily limited by the convolution kernel size in our neural network, which is currently 41×82. As shown in Figure 3.10, our method can handle motion within 41 pixels well. However, any large motion beyond 41 pixels, cannot currently be handled by our system. Figure 3.11 shows a pair of stereo image from the KITTI benchmark [114]. When using our method to interpolate a middle frame between the left and right view, the car is blurred due to the large disparity (over 41 pixels), as shown in (c). After downscaling the input images to half of their original size, our method interpolates well, as shown in (d). In the future, we plan to address this issue by exploring multi-scale strategies, such as those used for optical flow estimation [141].

Unlike optical flow- or phased-based methods, our method is currently only able to interpolate a single frame between two given frames as our neural network is trained to interpolate the middle frame. While we can continue the synthesis recursively to also interpolate frames at $t = 0.25$ and $t = 0.75$ for example, our method is unable to interpolate a frame at an arbitrary time. It will be interesting to borrow from recent work for view synthesis [45, 79, 89, 176, 223] and extend our neural network such that it can take a variable as input to control the temporal step of the interpolation in order to interpolate an arbitrary number of frames like flow- or phase-based methods.

3.5 Discussion

This chapter presents a video frame interpolation method that combines the two steps of a frame interpolation algorithm, motion estimation and pixel interpolation,

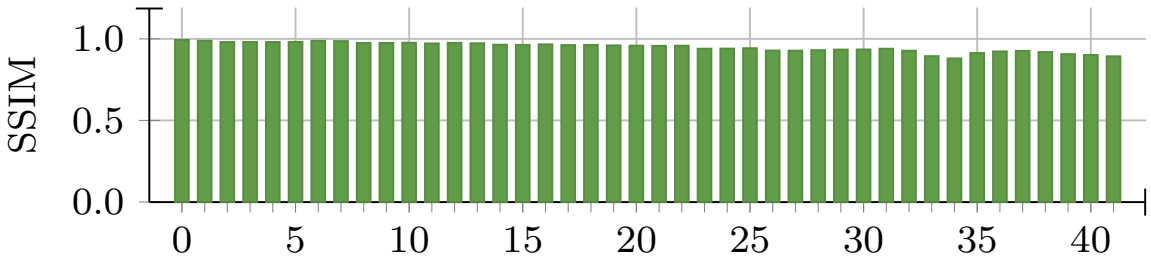


Figure 3.10: Interpolation quality of our method with respect to the flow magnitude (pixels).

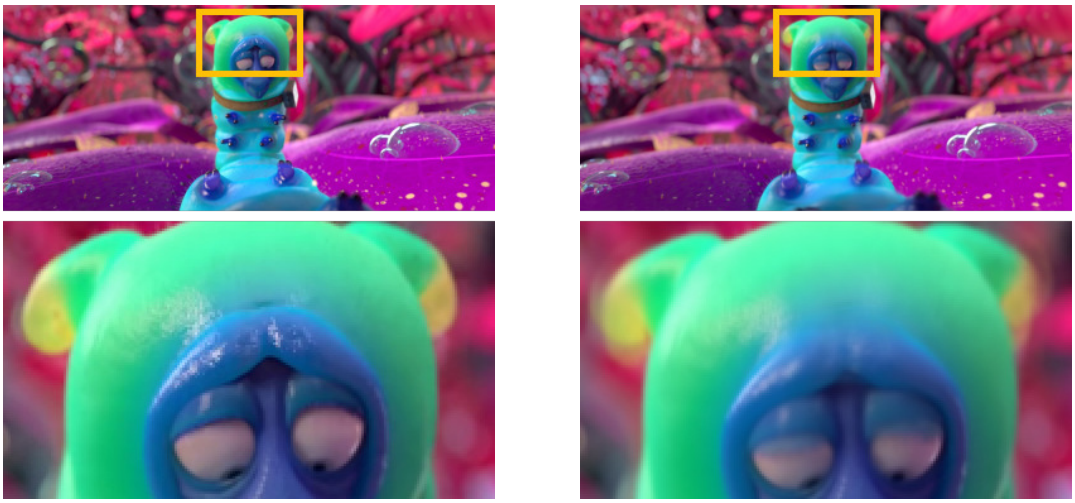
(a) Left view

(b) Right view

(c) Ours - full resolution

(d) Ours - half resolution

Figure 3.11: Interpolation of a stereo image. Our method fails to interpolate the left and right view in this stereo image due to the large disparity (over 41 pixels), as shown in (c). After downscaling the input images to half of their original size, our method interpolates well, as shown in (d).



(a) Ground truth

(b) AdaConv

(c) SepConv - $\mathcal{L}_1$

(d) SepConv - $\mathcal{L}_F$

Figure 3.12: Compared to the our original AdaConv approach that utilizes 2D kernels (b), our new separable convolution methods [128], especially the one with perceptual loss (d), incorporate 1D kernels that allow for full-frame interpolation and produce higher-quality results.

into a single step of local convolution with two input frames. The convolution kernel captures both the motion information and re-sampling coefficients for proper pixel interpolation. We developed a deep fully convolutional neural network that is able to estimate spatially-adaptive convolution kernels that allow for edge-aware pixel synthesis to produce sharp interpolation results. This neural network can be trained directly from widely available video data. Our experiments show that our method enables high-quality frame interpolation and handles challenging cases like occlusion, blur, and abrupt brightness change well.

However, our adaptive convolution formulation of the frame interpolation process suffers from a critical limitation of expensive computation in the inference process. As our method requires a $k \times k$ kernel to be predicted at each pixel, the model needs to output k^2 values for each pixel position. Due to such computational expense, we could only apply our method with a moderate-size kernel (up to 41×41), which limits the range of motion our method could handle. In a follow-up work [128], we extended our method to address that issue. The key idea is to leverage the separable convolution formulation instead of the original full convolution one. In this way, we only need to predict two kernels of size $k \times 1$ at each pixel, resulting in the $O(k)$ space complexity compared to the original $O(k^2)$ complexity. With this new formulation, we were able to train the model to process full video frames rather than local patches while using significantly larger kernels. That allows us to achieve significant improvements compared to this original, both quantitatively and qualitatively [128]. Fig. 3.12 shows an example comparing the frame interpolation quality of our new SepConv formulation with our original AdaConv formulation.

Our adaptive convolution frameworks for frame interpolation were the first to introduce the idea of incorporating motion estimation as a learnable yet explicit component into deep neural network architectures to support high-quality frame

prediction. This insight has soon been followed by contemporary works on deep learning-based video frame interpolation [10, 9]. With it, I hope to bring attention to the potential benefit of leveraging existing domain knowledge to design and incorporate inductive biases into modern neural-network based synthesis models. Our kernel-based formulation for motion estimation in deep neural network models has since become a standard paradigm for frame interpolation which has been adopted in many state-of-the-art methods in the field [30, 131, 186].

4 Motion-Adjustable Neural Implicit Video Representation

In recent years, Implicit Neural Representation (INR) has become a popular paradigm for modelling visual data. Originally developed for 3D data modelling [115, 121, 132, 164], INR has recently been shown successful in representing static images [163, 173]. In this chapter, I am interested in the question: is it possible to extend this novel representation to model video data? Moreover, is it possible to do it so that the video's temporal dynamics information can be manipulated?

It is important to note that contemporary image-based INR, with the use of Fourier-based positional encoding, can be viewed as a mapping from sinusoidal patterns with different frequencies to image content. Inspired by previous works in computer vision literature that explore the relation between the phase information in sinusoidal functions and their displacements, I hypothesize that it is possible to generate temporally varying content with a single image-based INR model by displacing its input sinusoidal patterns over time.

In this chapter, I will introduce a novel Implicit Neural Representation for videos. The proposed method incorporates a phase-varying positional encoding module into the conventional image-based INR model, and couple it with a phase-shift generation module that determines the phase-shift values at each frame. The model is trained end-to-end on a video to jointly determine the phase-shift values at each time and the mapping from the phase-shifted sinusoidal functions to the corresponding frame, enabling an implicit video representation. Experiments on a wide range of videos

suggest that such a model can learn to interpret phase-varying positional embeddings into the corresponding time-varying content. More importantly, the learned phase-shift vectors capture the video's meaningful temporal and motion information. In particular, manipulating the phase-shift vectors induces meaningful changes in the temporal dynamics of the resulting video, enabling non-trivial temporal and motion-editing effects such as temporal interpolation, motion magnification, motion smoothing, and video loop detection.

The content of this chapter was mainly adapted and slightly extended from the earlier version published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2022 [107]. The use of "*we*", "*our*", and "*ours*" throughout this chapter refer to the authors of the published paper (Long Mai and Feng Liu). My own contributions in this work are: the idea of modelling motion in video with the phase information in positional encoding, the design and implementation of the motion-adjustable neural implicit video representation model, the algorithms and implementation for video editing applications resulting from this novel representation, including motion filtering, motion-intensity adjustment, and video loop detection.

This chapter contains **video figures** that are best viewed using Adobe Reader. The video results in this chapter can also be viewed on our project website.

4.1 Introduction

Implicit neural representation (INR) has recently emerged as a powerful paradigm for representing visual data [121, 132, 163, 164, 173]. Notably, INR has recently been successfully adopted to represent 2D images for image processing and synthesis [5, 29, 48]. Image-based INR employs a coordinate-based multi-layer perceptron (MLP), typically along with Fourier-based positional encoding, to map 2D pixel coordinates to the corresponding color values. Existing works also studied video-based INR and

considered it as a natural extension of their image-based counterpart [113, 163]. Such an approach uses time as an additional input coordinate to the coordinate-MLP model, effectively treating a video as a 3D volume without explicitly modeling inherent temporal connection among video frames.

Alternatively, a video is often considered as a sequence of images evolving over time in computer vision research [137, 170]. This work explores a video-based INR from that perspective. We investigate if it is possible to leverage an image-based INR to generate temporally varying video content motivated by two observations. First, image-based INR, with the use of Fourier-based positional encoding [173], operates as a mapping from sinusoidal patterns of different frequencies to 2D image content. Varying the input sinusoids would necessarily cause the generated output to vary accordingly. Therefore, in principle a time-evolving image sequence can be generated from a single image-based INR by varying its sinusoidal functions over time. Second, displacements of sinusoidal functions can be modeled mathematically by the shifts in their phase angles. Time-varying sinusoids can therefore be achieved by assigning different phase shifts at different times.

We develop an implicit neural representation for videos based on these observations. We model the pixel generation process in a frame-wise manner with an image-based INR, and leverage the phase information in its positional encoding to generate temporally varying video content. Our model consists of two components, a frame generation module and a phase-shift generation module. Our frame generation module maps each pixel coordinate $\boldsymbol{c} = (x, y)$ to the color value $M_f(\boldsymbol{c})$ at the corresponding coordinates in the image plane. This frame generation module is a standard image-based INR model with a minimal yet important modification to its positional encoding (PE) operation. Different from a standard INR, each sinusoidal function in our PE is not static but to be shifted at each time t by a phase-shift vector $\boldsymbol{\phi}(t)$. The mapping $\boldsymbol{\phi}$ is

generated by the phase-shift generation module M_p, jointly trained end-to-end with M_f to fit the input video. After training, M_p can provide the per-frame phase-shift vector at each corresponding frame in the video. Those learned phase-shift vectors can be externally manipulated before entering the frame generation stage, potentially enabling new generated content with modified dynamics. That makes our neural implicit video representation motion-adjustable.

With the proposed neural implicit video representation, we center our study around two questions. *First, can the model learn to fit a video?* Compared to a standard INR approach where the spatial coordinate encodings are fixed across frames, the input coordinate encodings to our frame generation model constantly change from frame to frame, making it more challenging to memorize the pixel value at each location. *Second, does the learned phase space have any meaningful structures?* As the image content at each time is associated with a phase-shift vector, it is interesting to see whether manipulating the learned per-frame phase-shift sequence can result in meaningful changes in the generated video. Our experiments on diverse video content suggest positive answers. We found that the model can learn to interpret the learned phase-varying positional encoding into the corresponding time-varying video content. Interestingly, we found that the resulting phase space corresponds to meaningful information in the video. Manipulating the generated phase-shift vectors can enable different temporal-dynamics effects such as temporal interpolation, motion magnitude adjustment, motion filtering, and video loop extraction from the video as shown in Figure 4.1.

This work makes the following contributions.

- We introduce a motion-adjustable neural implicit video representation. Instead of treating the time dimension equally as the spatial dimensions, our representa-

Input Reconstruction Interpolation Frame t —— Interpolated —— Frame $t+1$

Original Motion Magnification Original Motion Smoothing

Figure 4.1: We extend a standard image-based implicit neural representation to a motion-adjustable neural implicit video representation by incorporating temporally varying phase-shift information into Fourier-based positional encoding. By changing the phase-shift values at inference time, our method can not only reconstruct video data but can also re-synthesize videos with modified motion properties. This chapter contains **video figures** that are best viewed using Adobe Reader. The video results in this chapter can also be viewed on our project website [1].

tion maps time to a driving signal to modulate the frame-generation process, effectively adapting regular image-based INR to generate temporally varying video content.

- We report the interesting finding that the phase information in Fourier-based positional encoding can be flexibly leveraged to capture temporal dynamics in a video. Our work adds to the growing literature on the use of Fourier-based positional encoding in INR, complementing prior works that study the roles of frequency information in Fourier-based positional embeddings.

- We experiment on a wide variety of real-world videos and demonstrate that our neural implicit video representation can not only represent a video but can also allow for modifying certain temporal-dynamics aspects of the video content,

enabling a motion-adjustable neural implicit video representation and supporting a range of video processing applications.

4.2 Related Work

Implicit Neural Representation has been shown a powerful approach to represent visual data, such as 3D data modelling [115, 132, 164, 121, 13, 211, 184, 15, 149, 95, 220, 151, 26] and image representation [163, 173, 110, 113, 172]. Image-based INR frameworks have been developed for numerous applications, including image compression [48], super-resolution [29], and image synthesis [5, 165]. In this paper, we focus on exploring a motion-adjustable neural implicit video representation. Different from the standard approach which extends image-based INR to fit a 3D video volume [163, 113], we leverage the phase information in the Fourier-based positional encoding to learn temporally varying video content with a regular 2D image-based INR.

Implicit Neural Representation for 3D Dynamic Scenes. Following the immense success of Neural Radiance Fields (NeRF) [121], many methods extend NeRF to model temporally varying 3D scenes from video data [57, 133, 138, 183, 207, 92, 56]. Existing works along this line typically treat video frames as the projection of a dynamic 3D scene onto the image plane. These methods explicitly model 3D scenes and per-frame camera poses. This paper works on a more relaxed setting without any 3D scene or camera information and focuses on adapting image-based INR model to capture the temporally evolving content in a video.

Fourier-Feature Based Positional Encoding. Positional encoding (PE) refers to the mechanism to represent position information by mapping low-dimensional input coordinates to higher-dimensional vectors, typically through a collection of sinusoidal functions. Initially made popular by Vaswani *et al.* through their Transformer paper [190], positional encoding has also proved critical for implicit neural representation

models [121]. Recent works have studied the importance of the frequency components in PE to the model's fitting quality [173, 163]. Our work adds to the growing literature on Fourier-feature-based positional encoding in INR, demonstrating that besides the frequency information, the phase information in Fourier-based PE can also be used to enable video modelling.

Phase-Based Motion Modelling. Our work is inspired by the rich literature on phase-based motion processing [192, 51, 59, 118, 116]. These works built on the connection between motion information in a video and its phase information extracted through frequency domain analysis [195] to enable various motion editing applications such as motion estimation [51, 59], motion magnification [192], and frame interpolation [118, 116]. In this work we explore the possibility of leveraging phase information embedded in the Fourier-based positional encoding to help implicit neural representation models learn temporal dynamics information in video data.

4.3 Method

4.3.1 Neural Implicit Image Representation

We first review image-based INR and motivate the use of phase shifts for generating temporally varying content. Image-based INR represents an image as a continuous function $f : \boldsymbol{c} \rightarrow \boldsymbol{v}$, where $\boldsymbol{c} = (x, y)$ are 2D coordinates on the normalized image plane, and $\boldsymbol{v} = (R, G, B)$ is the corresponding color value. The mapping function f is parameterized by the weights of a multi-layer perceptron (MLP) M_f. In practice, the input coordinates $\boldsymbol{c}$ are first mapped to higher-dimension vectors $\gamma(\boldsymbol{c})$ through a positional encoder module γ. M_f then maps the resulting positional encodings to the final color value $\boldsymbol{v}$ (Figure 4.4 left).

We adopt the widely used Fourier-based positional encoding scheme [95, 121, 190]

that forms the encoding by concatenating sinusoidal functions of c

$$\gamma(\boldsymbol{c}) = [\gamma_0(\boldsymbol{c}), ...\gamma_{N-1}(\boldsymbol{c})] \tag{4.1}$$

$$\gamma_i(\boldsymbol{c}) = [\sin(2^{i-i_0}\pi\boldsymbol{c}), \cos(2^{i-i_0}\pi\boldsymbol{c})] \tag{4.2}$$

where N denotes the number of frequencies. $\gamma_i(\boldsymbol{c})$ represents the encoding corresponding to the i-th frequency. The sin and cos functions are defined coordinate-wise. i_0 controls the lowest frequency component to use, which is typically set to 0 in most INR models. With the positional encoding incorporated, the resulting model can be viewed as mapping the sinusoidal patterns arranged in 2D planes to the corresponding image content.

4.3.2 Shifting Images with Pre-Trained Image-Based INR

Our key hypothesis in this work is that the displacement of the sinusoidal functions in positional encoding can be exploited to induce the image-based implicit neural representation (INR) model to generate varying outputs. In fact, special cases of image transformations such as global translation can be induced by phase-shifting even with a pre-trained image-based INR model.

As a preliminary test to motivate the use of our phase-varying positional encoding described later, here we train a standard image-based INR model on a *static image*. Recall that the positional encoding operation is defined by Equations 4.1 and 4.2

After training the model, we shift the phase of each sinusoidal function in the positional encoding in Equation 4.2 with a phase-shift term ϕ whose i-th component is defined as

$$\phi_i = [2^{i-i_0}\pi\delta_x, 2^{i-i0}\pi\delta_y] \tag{4.3}$$

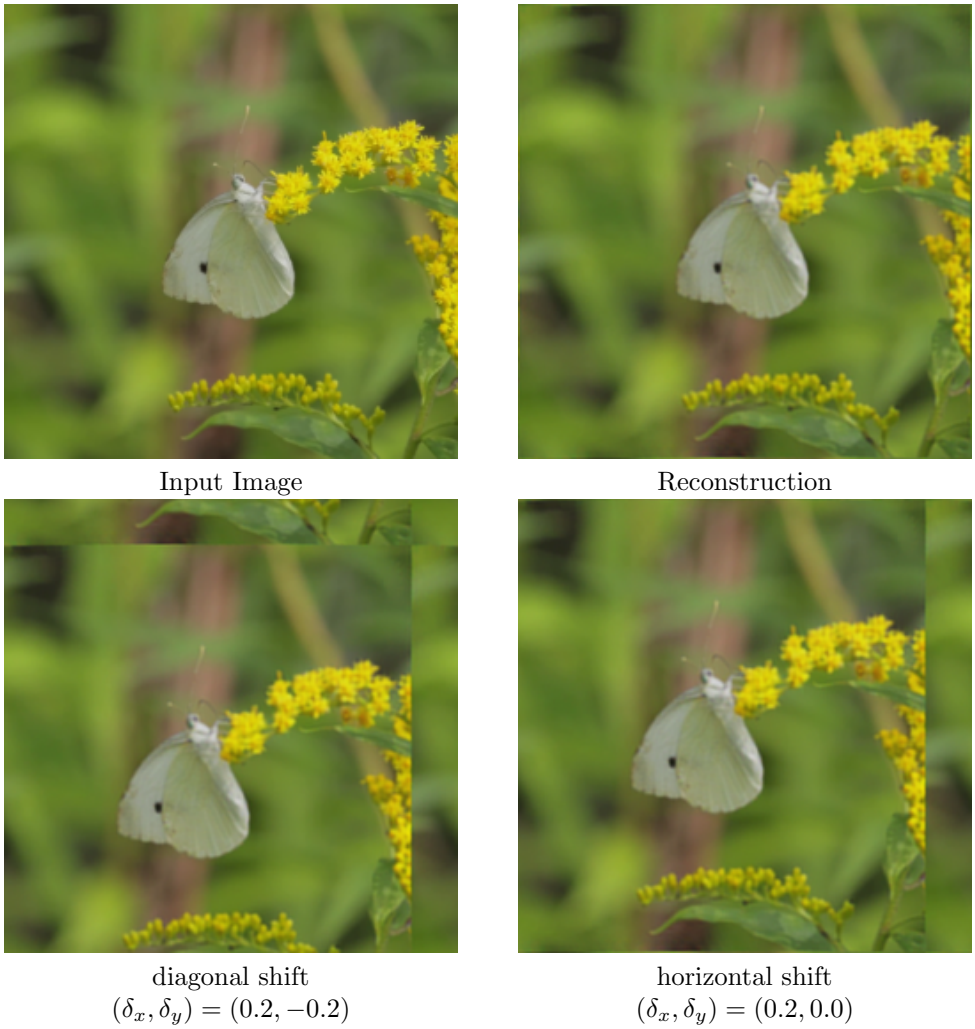
where δ_x and δ_y denote the desired (normalized) shifted amounts in each direction.

Equation 4.2 becomes

$$\gamma_i(\boldsymbol{c}) = [\sin(2^{i-i_0}\pi\boldsymbol{c} + \phi_i), \cos(2^{i-i_0}\pi\boldsymbol{c} + \phi_i)] \tag{4.4}$$

As shown in Figure 4.2, passing the modified positional embedding through the coordinate-MLP results in spatially shifted versions of the original image. Note that the top-bottom and left-right folding effects are generated by the model due to the repeating nature of the sinusoidal functions. This happens when i_0 is set to 0 in Equation 4.2. We repeat the experiment with $i_0 = 1$ and show the result in Figure 4.3. Note that the folding behavior disappears. Instead, the model tends to synthesize the unseen areas with content from the nearby regions. For example, the yellow color of the flower was extended to the right in Figure 4.2 (bottom right). This ability to avoid the folding bias motivates our use of $i_0 = 1$ in our experiments.

However, as to be expected the model also hallucinates noisy content in those areas since the model was not trained to interpret the part of the sinusoidal functions corresponding to those regions. We also note that the phase-shift values defined as in Equation 4.3 corresponds to all sinusoidal functions being shifted by the same amount and therefore can only model global translation. A pre-trained image-based INR model cannot interpret arbitrary per-channel phase shifts as it was only trained with the input sinusoids having a fixed phase relation. In our Neural Implicit Video Representation described below, we take one step further and explore training the INR model explicitly with phase-varying positional encoding to model complex transformations in real video data.



Input Image | Reconstruction

diagonal shift
$(\delta_x, \delta_y) = (0.2, -0.2)$

horizontal shift
$(\delta_x, \delta_y) = (0.2, 0.0)$

Figure 4.2: Phase-shift-induced image shifting with pre-trained image-based INR model ($i_0 = 0$).

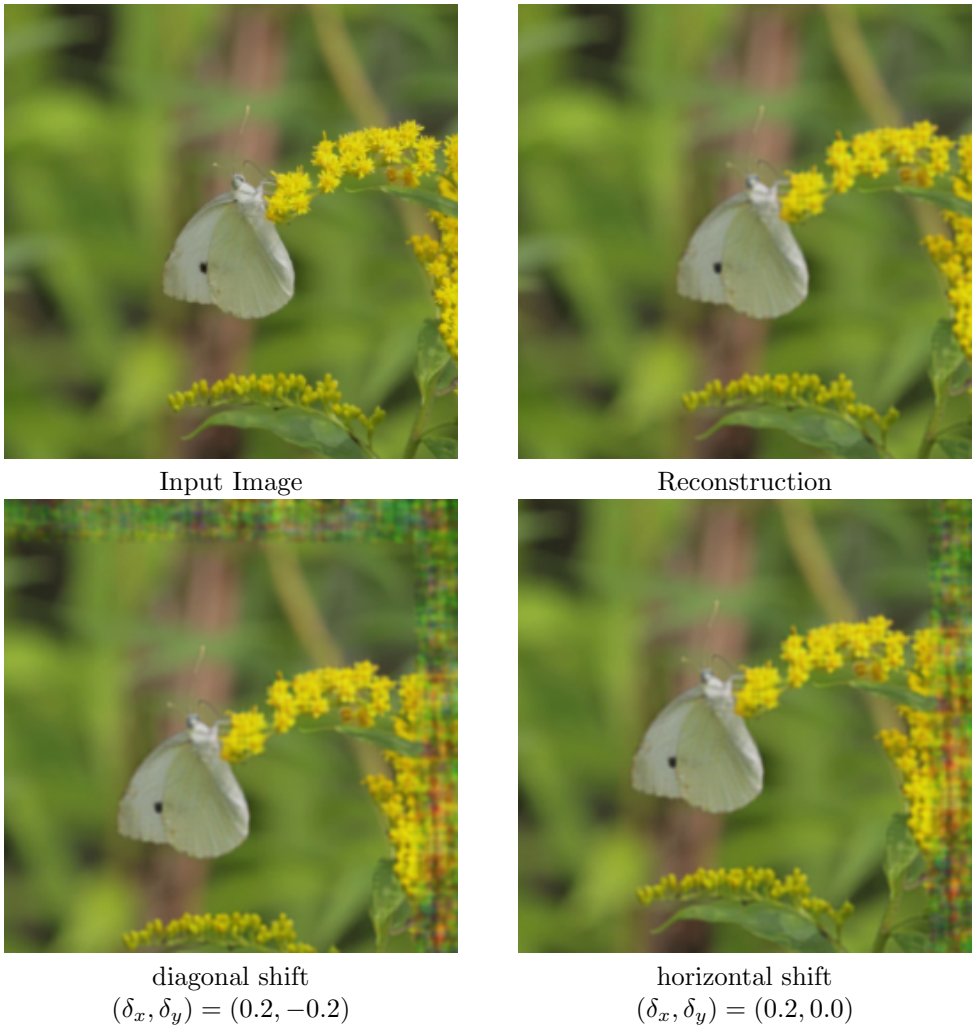


Figure 4.3: Phase-shift-induced image shifting with pre-trained image-based INR model ($i_0 = 1$).

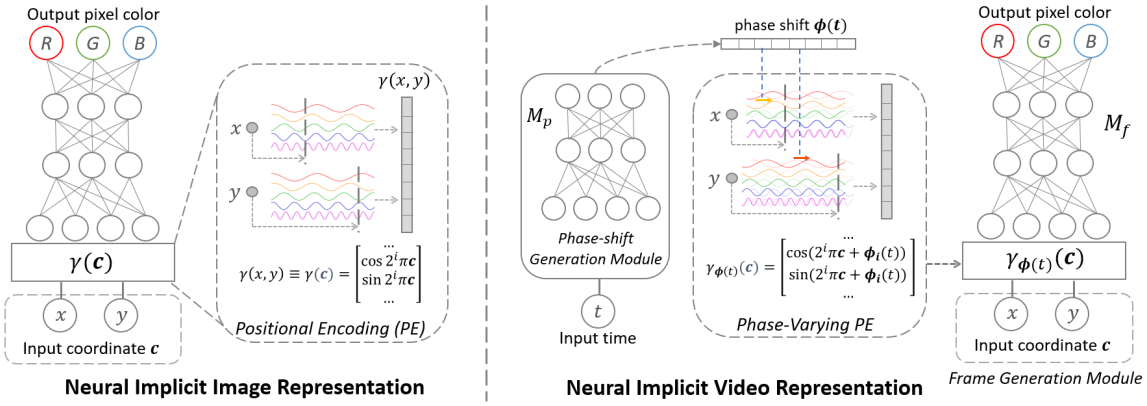


Figure 4.4: Motion-Adjustable Neural Implicit Video Representation. We extend image-based implicit neural representation (left) to model a video. Our method determines the phase-shift $\phi(t)$ at each time t using the phase-shift generation network M_p. The frame generation network M_f synthesizes the video frames corresponding to the positional embeddings with the phase shifted by $\phi(t)$. At inference time, $\phi(t)$ can be manipulated to generate new videos with modified dynamics.

4.3.3 Neural Implicit Video Representation

The displacement of the sinusoidal functions can be achieved by shifting their phase angles. Inspired by that fundamental mathematical relation, we jointly determine the phase-shift values at each time and the image-based INR model that map the phase-shifted positional encodings to the corresponding video frames as shown in Figure 4.4 right. This leads to our Neural Implicit Video Representation. Below we detail its two main components.

Frame Generation. The frame generation module M_f generates each 2D video frame. As in conventional image-based INR, M_f maps each 2D coordinate $\boldsymbol{c}$ to the corresponding pixel value using a coordinate-MLP with Fourier-based positional encoding. To make it generate different video content at different time, we modify its

positional encoding module to enable phase-varying positional encoding. Specifically, we incorporate an explicit phase-shift term into each sinusoidal function. As a result, the per-frequency positional embedding in Equation 4.2 is modified to

$$\gamma_{\phi(t)_i}(\boldsymbol{c}) = [\sin(2^{i-i_0}\pi \boldsymbol{c} + \phi_i(t)), \cos(2^{i-i_0}\pi \boldsymbol{c} + \phi_i(t))] \tag{4.5}$$

where $\phi_i(t)$ is a two-dimensional vector representing the i-th component of the phase shift at time t. With this minimal change, M_f can generate different values for the same (x, y) coordinate at different time, adapting an image-generation model for video generation.

Phase-Shift Generation. We parameterize the mapping from time t to phase shift $\phi(t)$ with a neural network M_p. As the mapping has a continuous nature, we implement M_p as a 1-D implicit neural representation. Specifically, the input t is first mapped to a positional embedding $\gamma(t)$ using the regular positional encoding procedure following the one-dimensional instantiation of Equation 4.1. The resulting positional embedding is then processed by an MLP to generate the output phase-shift vector $\phi(t)$.

Model Training. At each training iteration, we randomly sample one video frame V_i along with its frame index i, which is normalized to $[-1, 1]$ and passed through our model to generate the frame $\hat{V}_i$. The model is trained with the reconstruction-based loss function

$$L(\hat{V}_i, V_i) = ||\hat{V}_i - V_i||_1 + \lambda||\Phi(\hat{V}_i) - \Phi(V_i)||_2 \tag{4.6}$$

where $\Phi(.)$ denotes the feature maps extracted from the pre-trained VGG-19 network [162]. The loss function is composed of two loss terms: the conventional L_1 loss and the perceptual loss to encourage preserving better image details. $\lambda = 0.2$ is a weighing factor.

During training, we found it beneficial to update M_f and M_p in an asymmetric

manner. In particular, we update the parameters of both networks only on half the number of frames evenly sampled across the video. For the remaining frames, we only update the parameters of the phase-shift generation network M_p while freezing the parameters of M_f during back-propagation. In that way, M_f is prevented from overfitting to all the frames while still able to guide the update of M_p such that the predicted phase-shifts that can be correctly interpreted to generate the hold-out frames. We found such asymmetric training procedure critical for learning well-structured phase space.

4.4 Experiments

4.4.1 Implementation Details

We implement both M_f and M_p as MLPs with 3 hidden layers and 1024 neurons per hidden layer. Following [163], we use the sine activation function in all hidden layers. For M_f, the output layer has three neurons, corresponding to the RGB color values. Each neuron has a tanh activation function to constrain the output value to $[-1, 1]$. For M_p, the number of output neurons is equal to twice the number of frequency channels in the positional encoding module of M_f. The number of frequency channels N in positional encoding is determined by the number of samples L along each dimension of the input video as $N = [\log_2(L) + i_0]$ as done in [163]. L is taken to be the length of the video for the temporal dimension and the smaller side of the frame for the spatial dimension. We use $i_0 = 1$ in Eq. 4.5 and 4.2 for all experiments. We trained our model using the ADAM optimization algorithm [84] with learning rate 0.0001 for 6,000 passes over an input video. It takes about 18 hours to train on a video of 120 frames with resolution 256×452 on one NVIDIA 2080Ti GPU.

Method	PSNR	SSIM
Direct-VINR	31.98	0.897
Phase-NIVR	32.05	0.905

Table 4.1: Video fitting performance.

4.4.2 Learning to Fit Video Data

We examine whether incorporating the phase-varying positional encoding and the generated phase shifts hurts the ability of the model to fit the video data well. Compared to standard INR formulation, it is more challenging for our model to fit the coordinate-to-color mapping as the positional embeddings of the input spatial coordinates constantly change across frames. We test our neural implicit video representation (Phase-NIVR) on 25 videos from the WAIC-TSR dataset [135] that covers different content and motion types. For each video, we use the first 120 frames and resize them so that the small side is 128-pixel.

For comparison, we also train a direct extension of INR to video, named Direct-VINR, that incorporates t as an additional input coordinate. We use the same architecture and loss function as in our model to experiment with Direct-VINR. We train both models on each video in the dataset and compute the PNSR/SIIM reconstruction scores from their reconstructed videos. We report the video fitting qualities in terms of two quality metrics PSNR and SSIM in Table 4.1. The results indicate that our method performs comparably with Direct-VINR. This suggests that incorporating phase-varying positional encoding, while making the learning problem more challenging for the mapping network, does not prevent the model from fitting the videos.

Figure 4.5 shows 30-frame segments of some example reconstructed videos. Consistent with the numerical scores, we observe the reconstructed videos from two methods often have comparable visual quality (Figure 4.5 rows 1-3). In some cases, we observe that Direct-VINR tends to struggle in reconstructing object motions over relatively static and uniform background such as in the last example in Figure 4.5 when the man's legs pass through the uniform sky region. That could possibly be due to the model overfitting to the same background color that repeatedly appears at the same spatial location in most of the frames. We found that by explicitly removing such fix-position bias, our method tends to be more robust in such scenarios.

Figure 4.5: Video reconstruction examples. Our method can fit video content with comparable visual quality as Direct-VINR (rows 1-4) while tending to be robust in capturing object motion over uniform background such as the man's legs over the uniform sky regions (row 5).

4.4.3 Phase-based Motion Manipulation

The previous experiment shows the ability of our model to map per-frame phase information into the frame content. However, it is not clear whether the learned phase captures meaningful temporal dynamics structure or simply serves as an index for the model to memorize the frame content. In this section, we inspect how manipulating the generated phase-shift sequence $\phi(t)$ influence the change in the output frames.

4.4.3.1 Temporal Interpolation

We examine if interpolating two phase-shift vectors corresponds to a meaningful interpolation in the video domain. We sample five videos in the WAIC-TSR dataset that cover different scene types and have good reconstruction quality (PSNR ≥ 28.5) from the previous experiment. We re-train our model on 120 frames from each video sampled at half the original frame rate. For this test, we train the model on video frames resized to 256×452 so that more details can be observed. After training, we use M_p to generate the phase shift vectors at each time t and perform interpolation between each pair of consecutive phase-shift vectors to obtain the interpolated phase-shift sequence. We use spherical linear interpolation to account for the circular nature of phases [156]. The resulting phase-shift sequence is used in the frame generation module M_f to generate the final frames.

Figure 4.6 shows the interpolated video results. The videos were set to be played back at two frames per second in the figure for easier inspection. First, it can be observed that the interpolated frames have comparable visual quality as the original frames. This indicates that the model can indeed interpret the positional embedding from the interpolated phase-shift vectors into plausible video content rather than treating them as out-of-distribution samples. Second, the appearance of the frames

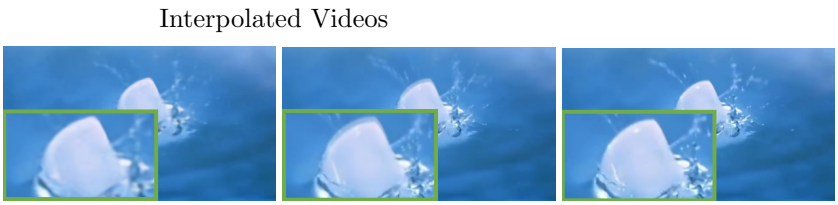


Figure 4.6: Temporal interpolation examples. The frame generation model can synthesize plausible interpolated frames with interpolated phase-shift vectors during inference time. The interpolation results often show plausible motion transition rather than copying nearby frames or taking frame-wise average (2nd row).

continuously changes, indicating that the model can associate the change in the phase-shift vectors to the change in the video domain rather than simply copying the content from the nearest frames. Finally, we inspect whether the interpolated frames are the results of the pixel-space average of the corresponding neighboring frames. We found that in general the interpolated frames are different from the frame-wise average results (Notice the ghosting around the ice cube in the averaging result in Figure 4.6 (bottom row)). We observe that when the motion is sufficiently small, the interpolated video does capture the interpolated motion. However, with a larger inter-frame motion, the model may not identify the corresponding large-moving regions across frames as part of a single motion. In those cases, interpolation tends to reduce to a blending operation, resulting in occasional ghosting artifacts as can be observed in the "running men" sequence (the third examples in Figure 4.6's top row).

4.4.3.2 Motion Filtering

The previous interpolation test suggests that the learned phase-shift vectors can be associated with the temporal states of the video content. We furthermore perform a simple experiment to test whether low-pass filtering the learned phase-shift sequence can smooth motion in the video. For this test, we collect videos that have some jiterring object motion on top of a longer-range motion trajectory such as a tunning fork vibrating while moving (Figure 4.1 (bottom-right)). After training our model on each video, we treat the generated phase-shift sequence as a multi-dimensional time series and apply a median filter with a temporal window-size of 7 to it. The filtered phase-shift sequence is used with the frame generation model M_f to synthesize the new video.

Figure 4.1 (bottom-right) and Figure 4.7 show two motion filtering results. More examples can be found in our supplementary video. We observe that filtering the learned phase-shift sequence leads to the resulting videos with reduced high-frequency jittering while the larger-scale motion is preserved. Note the overall up-down motion of the tuning fork in Figure 4.1 (bottom-right) is retained while its vibration is largely removed. Also, the base concrete platform in Figure 4.7 is stabilized while its overall motion direction is preserved.

4.4.3.3 Motion Magnitude Adjustment

Inspired by phase-based motion processing works [192, 195], we are curious if manipulating the phase-shift vectors in our framework can alter the motion magnitude in videos. Specifically, we test whether adjusting the magnitude of the difference between neighboring phase-shifts can result in motion magnitude change.

We test our method on different videos with object fluctuating in space. For phase-

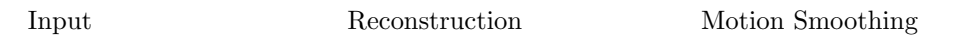


Figure 4.7: Motion filtering. Low-pass filtering the phase-shift sequence $\phi(t)$ at inference time can make the frame generation model to generate a new video with smoother object motion. 1st example: The concrete base becomes more stable while its larger-scale motion is preserved. 2nd example: The vibrating motion of the car-washing tool in the original video was significantly smoothened in the re-synthesized version. 3rd example: the hand-grip exhibit strong jiterring motion in the original video but remained relatively static after motion smoothing.

shift adjustment, we first scale the difference between each consecutive phase-shift vectors $\Delta\hat{\phi}(t) = \alpha(\phi(t+1) - \phi(t))$. We then fix the phase-shift vector at the first frame and re-compute the phase-shift sequence with the modified pair-wise difference $\Delta\hat{\phi}(t)$. The new videos are synthesized from the modified phase-shift sequence. Figure 4.8 and Figure 4.1 (bottom left) show two example videos with different scaling factor α values. When α is smaller than one, the resulting video shows reduced motion

Original ($\alpha = 1.0$) $\alpha = 0.5$ $\alpha = 1.5$

Figure 4.8: Motion magnitude adjustment. Scaling the phase-shift sequence $\phi(t)$ at inference time can alter the motion magnitude in the synthesized video. Varying the scaling factor allows for both motion minification and motion magnification.

magnitude, leading to the motion minification effect. The magnification effect was obtained with $\alpha > 1$. Note that only the motion magnitude was modified while the overall motion structure, including the direction of motion and different motion stages were preserved.

4.4.3.4 Video Loop Detection

Hypothesizing that the phase-shift vectors encode the states of dynamics, we investigate if we can detect loops in videos with repeated motion by analysing the phase-shift

sequence.

We adopt a simple approach to detect loops in a video from the learned phase-shift sequence. Let $\{\phi(k)\}_{k=0..N}$ represent the learned phase vectors from the video with N frames. We identify the looping point by determining the pair of frame indices $\hat{i}$ and $\hat{j}$ that minimize the cost function

$$\min_{i,j|j\geq i+\tau} \|\phi(i) - \phi(j)\| + \beta\|\Delta\phi(i) - \Delta\phi(j)\| \tag{4.7}$$

where $\|.\|$ denotes the L_1 distance, $\Delta\phi(i) = \phi(i) - \phi(i-1)$ represents the phase-difference vector, and β is a weighing factor to balance between phase matching and motion matching terms. τ determines the desired minimum length for the extracted loop. The idea is to determine the pair of as-similar-as-possible phase shifts that also have similar phase transition. After solving for $\hat{i}$ and $\hat{j}$, the sub-sequence $\{\phi(k)\}_{k=\hat{i}..\hat{j}}$ forms the candidate loop. The new video synthesized with this sequence would ideally transition from the $\hat{j}$-th frame back to the $\hat{i}$-th frame which has similar dynamics state, forming the illusion of looping.

In practice, we observe that perfect matching is only possible for simple mechanical motion where objects perfectly repeat themselves. For more organic motion such as human action, slight variations in object poses can cause a perceivable jump at the looping point. To address that problem, we further modify the phase-shift sequence with a simple phase blending process. We modify the first l phase-shifts by blending them with $\phi(\hat{j})$ using spherical linear interpolation with the blending weight $\alpha(n)$ of the n-th vector defined as $\alpha(n) = \frac{n}{l}$.

Figure 4.9 shows example loop extraction results from two potentially looping videos. Please check our supplementary video for more examples. In general, the loop points can be successfully detected by phase matching. This indicates that the similar phase-shift vectors reflect similar scene states reappearing at different times. We note

that phase blending helps improve the perceived looping noticeably. The wind-chime example (Figure 4.9 (bottom)) is particularly challenging to handle as the original video contains small camera motion. For that reason, no perfect loop point exists that can match both the background and the object motion, resulting in the noticeable temporal seam in the looping result. Surprisingly, with phase blending it is possible to achieve a seamless looping video. This indicates that manipulating in the phase-shift space can lead to plausible modification in the video domain.

Figure 4.9: Video loop detection. Potential repeat point in a video can be detected by simple phase-matching strategy in the learned phase-shift sequence $\phi(t)$. Applying phase blending improves the looping results especially for challenging scenarios, such as when both the wind chime and the background move due to subtle camera motion (3rd row).

4.5 Discussion

Our method demonstrates that the phase information in the positional encoding can be used to encode temporal dynamics information in videos. That begs the question: what does the learned phase-space look like? In this section, I visualize

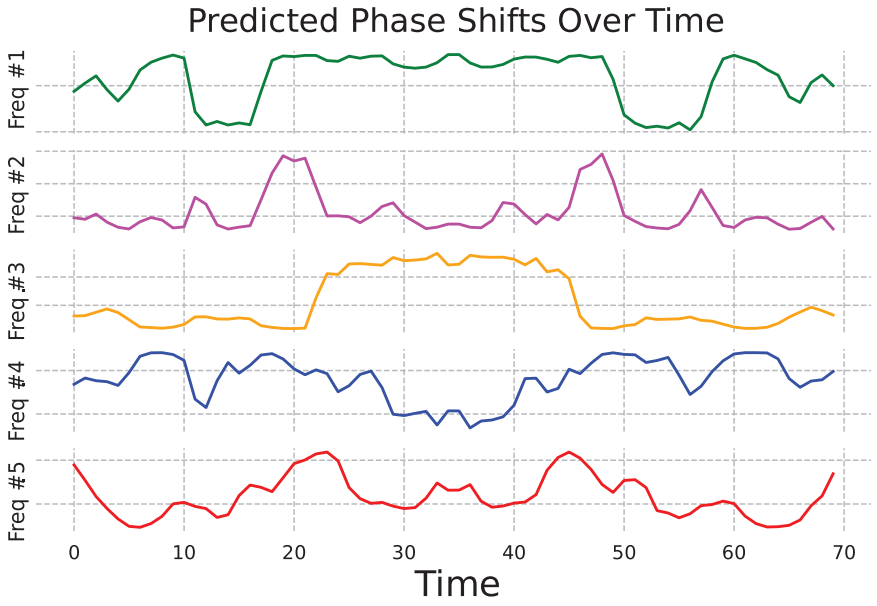
the learned phase-space to draw more insights into the relation between the phase information and the dynamics of the input video content. I will also discuss the role of the phase-generation module via experimenting with an alternative technique for phase prediction that follows a direct optimization approach.

4.5.1 Inspecting the Learned Phase Space

Our experiments suggest that the learned phase-shifts $\phi(t)$ can be associated with meaningful transition in videos. We visualize $\phi(t)$ as a function over time. Figure 4.10 shows such a visualization for our model trained on a video with structured and symmetric nature while containing some complex motion. We show the phase-shift visualization for 5 out of 14 phase-shift series (corresponding to 14 M_f's frequency channels).

Inspecting the visualized phase shifts, we can see that the phase-shift series evolve smoothly over time rather than forming a sequence of uncorrelated states. More interestingly, we found that the phase-shift series are well structured. The phase-shift plots contain highly symmetric patterns, reflecting the symmetric nature of the input video. We observe similar structured phase-shift patterns consistently in most videos that we experimented with. The transition in the phase-shift series often corresponds in meaningful ways to the transition in the visual domain.

Occasionally, we also observe a localized-control capability. For this example, we found that the fifth phase-shift series (the red curve in Figure 4.10) correlates with the movement of the hair-lock region. Keeping the fifth phase-shift evolving over the whole temporal range while freezing the phase-shift values of all other frequency channels at certain keyframes results in the re-synthesized scenes frozen at the selected keyframes while keeping the hair lock moving in similar ways (Figure 4.10 (bottom)). This localized-control behavior is interesting as it suggests the simple MLP networks can



Reconstructed Video | Freq 5 only (keyframe at $t = 0$) | Freq 5 only (keyframe at $t = 40$)

Figure 4.10: We visualize five channels of the learned phase-shift values $\phi(t)$ as a function of time (top). The structure of the phase-shift series reflects the symmetric nature of the video (bottom-left). In addition, the fifth phase-shift series (the red curve) correlates with the hair-lock movement even when other channels are frozen to one keyframe.

potentially learn non-trivial spatial-temporal relations from raw visual data without explicit motion, correspondence, or semantic supervision. However, we would like to note that our current model does not exhibit this localized-control ability on all videos. In general cases, one phase-shift series often correlates with more global motion, and the motion of one visual element is often influenced by multiple phase-shift channels. Explicitly encouraging such localized-control capability by incorporating a specialized

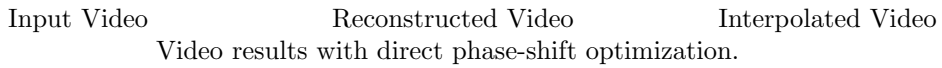
training strategy would be an interesting direction for future exploration.

4.5.2 Phase-Shift Learning with Direct Optimization

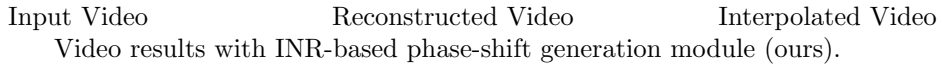
Our method parameterizes the mapping from t to $\phi(t)$ with a 1D coordinate-based mutilayer perceptron. This unifies the whole system under the neural implicit function paradigm. An alternative approach to realize the phase-shift generation module would be to optimize a sequence of per-frame phase-shift vectors. However, we found such direct optimization approach difficult to obtain well-structured phase space for two reasons. First, the phase shift value $\phi(t)$ is a function of the continuous time signal t, optimizing the per-frame phase shifts as a discrete sequence make it difficult to capture such nature of continuity. Second, phase naturally has circular nature. That makes the phase space highly non-Euclidean and challenging to optimize directly.

The top row of figure Figure 4.11 shows the video results synthesized by the model trained while replacing our INR-based phase-shift generation module with the one that uses direct optimization. For reference, we show the results generated by our method in the bottom row of Figure 4.11. In the figure, we show the video reconstruction results and the temporal interpolation results that is generated by re-synthesizing the video using the interpolated phase-shift sequence. The interpolated phase-shift sequence is obtained by interpolating each pair of consecutive original phase-shift vectors as done in our temporal-interpolation experiment in Section 4.4.3.1.

We optimize the per-frame phase-shift vectors jointly with the frame-generation module training, using the ADAM optimization algorithm [84]. We also experimented with other optimization algorithms such as SGD and LBFGS and observe similar results. From Figure 4.11, we observe that while the direct optimization approach can fit the video data reasonably well the resulting phase-shift space lacks the structure that reflects the inherent continuity in the video's temporal information. That leads



Video results with direct phase-shift optimization.



Video results with INR-based phase-shift generation module (ours).

Figure 4.11: Direct optimization for per-frame phase-shift sequence is challenging. Without stronger structural regularization, it is difficult for the optimized phase-shift sequence to reflect the inherent continuity in video data. The resulting model cannot generate plausible results from the interpolated phase-shift vectors (top row). Our method uses an implicit neural representation model to parameterize the mapping from the continuous input t to $\phi(t)$. This partly serves as an implicit continuity-aware regularization.

to implausible results when re-synthesizing the video with interpolated phase-shift sequence. The model seems to use the phase-shift vector simply to index and memorize the mapping from the resulting shifted sinusoidal patterns to the RGB content at each individual frame, without capturing the relation across frames. More advanced optimization algorithms and explicit regularization would be needed to encourage modelling temporal relation between neighboring phase-shift vectors. We conjecture that our method, by parameterizing the phase-shift optimization process as neural network training, imposes an implicit continuity-aware regularization since the model is trained to map the continuous signal t to the continuous output $\phi(t)$. We note that

other ways of neural network based parameterization exists, for example by using more sophisticated sequential models such as recurrent neural networks (RNNs) [72, 32] or transformer architectures [190]. Exploring those models in the context of our problem would be an interesting direction for future work.

4.5.3 Limitation and Future Work

Our study demonstrates the surprising effectiveness of using phase-varying positional encoding in image-based INR to capturing temporal dynamics. However, our method has several limitations.

First, while our model can fit a video, the reconstruction quality is not perfect. Our reconstructed videos are often slightly more blurry and sometime noisy compared to input videos, as can be seen from the video results. We built our model upon the most standard Fourier-based positional encoding scheme which uses a pre-defined set of frequency components without per-example tuning. Incorporating more advanced frequency selection principles [163, 173] or employing local implicit function models [29, 110, 113] are promising directions to improve the visual quality.

Second, as our framework requires example-specific training, it takes many hours to process one video. Extending our method to multiple-video settings with hyper-networks models [164] or meta-learning [172] can be fruitful directions to explore in future work.

Finally, while our motion-adjustable neural implicit video representation shows promising results for various motion editing tasks, we believe that incorporating application-dependent domain knowledge for those tasks will improve our method to generate better results.

Another interesting direction for future research is to extend the proposed method to make it a generative model. While capable of synthesizing video content with

modification in temporal dynamics content, the current model is not a full-fledged generative model. While we can slightly manipulate the temporal dynamics information by adjusting the learned phase space in certain ways, we cannot freely sample arbitrary phase values to generate arbitrary variations from the original videos as demonstrated in [66]. Exploring our model with GAN-based training objectives could be a promising direction to pursue.

5 Conclusion

Deep neural networks have transformed every aspects of computer graphics and vision research. For visual content synthesis, imagery with impressive levels of realism can now achieved with deep generative models [82, 139, 147]. Synthesizing video content, arguably due to the the lack of effective motion-aware inductive biases, has been shown much more challenging for deep neural network to master.

Before the deep learning era, computer graphics and vision research already established a vast literature on video synthesis. Traditional graphics- and vision-based methods exploited the rich source of domain knowledge and insights reflecting the researchers' understanding on the nature of visual data. However, in those more traditional methods, such insights were often incorporated as heuristics. That often limits their robustness and flexibility, leading to fragile systems. The key motivation of my research in this dissertation is that such insights are can still be highly relevant in this era of modern deep learning approaches. In particular, they can be used as effective inductive biases for deep learning models.

Toward this idea of using domain knowledge as inductive bias for video synthesis, I started my exploration with the problem of video frame interpolation. The existing literature in this domain from traditional graphics and vision research allowed me to stand on the giants' shoulders. In particular, I was inspired by the key insight that video frame interpolation can be modelled as fundamentally composing of two processes: motion estimation and pixel synthesis. Leveraging that insight as an

inductive bias, I proposed the adaptive convolution formulation for CNN-based video frame interpolation. This formulation effectively combines both processes into a unified convolution process. Thanks to the inductive bias that explicitly encourages motion estimation and sampling-based synthesis, the proposed method was able to synthesize results with much higher quality compared to previous deep learning based techniques for video synthesis which follow the direct prediction approach. Compared to the conventional techniques, the proposed method enables end-to-end learning from data. Therefore, the model can make use of the incorporated domain knowledge in a flexible and data-adaptable manner rather than relying on heuristics or rigid hand-engineered components. This makes the proposed method more robust against the challenging scenarios where heuristics-based approaches often have difficulties handling such as blurriness, occlusion, and the scenarios where motion cannot be faithfully represented by optical flow (e.g., lighting changes or transparent movement) [117].

More recently, I explored applying this line of research to enable video manipulation with the implicit neural representation (INR). Simply extending the standard image-based INR model to video by considering time as an additional input dimension, while suitable for video fitting, does not enable intuitive temporal dynamics manipulation. By viewing contemporary INR models as learned (non-linear) inverse Fourier transform processes, I can take inspiration from the important insight about the relation between the phase information in the frequency domain and the motion information in the image domain. This important insight was developed and applied for a long time in many computer vision works [54, 117, 136, 194]. However, it has been relatively under-explored in the age of deep learning based approaches. My research described in Chapter 4 showed that incorporating such phase-based manipulation insight as a simple modification into the standard image-based INR model makes it possible to not only represent the video data well but also manipulate the video to achieve

different temporal dynamics effects.

Since the publication of our first paper on video frame interpolation in 2017, the idea of incorporating explicit motion modelling into deep learning based video synthesis systems has been greatly adopted. Motion estimation modules have become integral components in not only video frame interpolation models [9, 10, 126] but also other video processing models such as video super-resolution [108, 197], video-to-video translation [198], and video generation [144, 158]. Most of these works, however, often consider motion estimation simply as a computational trick and heuristic constraints. For that reason, they tend to be limited to using only optical flow as motion models.

In this dissertation, I hope to push forward the view of leveraging the motion modelling insights from traditional methods as useful inductive biases for deep model learning. That view was what inspired us to incorporate not only the motion estimation module into video frame interpolation, but also combine it with the pixel synthesis process into the overall adaptive convolution formulation for frame synthesis. It was also from that view that connect the idea of phase-based motion modelling from older computer vision literature with the modern implicit neural representation to adapt it to video synthesis and manipulation tasks.

Future Work

Toward Systematic Approaches to Discovering Knowledge-Based Inductive Biases for Motion Modelling

I hope our works have demonstrated the value of this research direction of leveraging existing domain knowledge as inductive learning biases. Nevertheless, my work in this dissertation have only investigated this research direction in a rather ad-hoc manner. The incorporated insights were determined by carefully reviewing the large existing literature, and the incorporation of those insights as inductive biases in model design

was done on a per-problem basis.

More fundamental methods to discover and incorporate such motion modelling insights in more systematic ways will be an interesting direction for future research. Meta-learning techniques that can analyze the results from existing conventional method and automatically discover motion-related inductive biases from them to train problem-specific models can be a fruitful direction to explore toward this goal.

From Video Editing to Video Generation: Dynamical Systems Modelling as Inductive Bias

My research described in this dissertation only focused on the video synthesis settings that synthesize video content from a provided source content. This is a the most common setting in modern video editing workflows. However, the future of content creation will likely also have creators generating content from scratch. In such an un-conditional, or weakly-conditional synthesis setting, full visual content will be generated from a completely random noise or simplified input signal from different modality such as text or audio.

Impressive results for image generation have recently been demonstrated [82] Compared to the image generation part, the video generation results still pretty much lack behind. State of the art methods can only generate very short videos, with low-resolution, and often with severe distortions.

On the challenge that hampers the current generative models to properly model video data, one key hypothesis I plan to investigate is that those existing models often lack the inductive bias that encourage temporal continuity and structural variation at multiple scales. Our visual world is not a set of disorganized, randomly sampled data. It has very rich structures with content organized along many different scales. For example, at the smaller time scale, we have content corresponding to consecutive

frames in a video. At larger time scale, perhaps it can arrange different videos of the same subject while gradually varying the background scenes. And at an even larger time scale, the changes from one video to the next can be more profound but may still exhibit some continuation, such as transitioning from one type of animal to another animal, then to the animal in a different environment, and then from one type of environment to another. In the current frameworks, such structural relation has not been encouraged. The model simply map stochastic noise, to equally stochastic sampled data.

One idea I want to explore to address this challenge is to leverage dynamical-systems modelling to encourage structural inductive bias in model learning. There are many appealing properties of dynamical system modelling that can benefit the modelling of video data, such as the built-in notion of temporal continuity and multi-scale structured variation. Dynamical systems are well-known for their ability to generate very rich dynamical patterns, and with good design, they can achieve that in a controllable manner.

What I plan to explore as a first step is to replace the stochastic noise input with the patterns generated from a simple dynamical system. I plan to start with the simplest system possible: a set of independent harmonic oscillators with different frequencies. Interestingly, this would lead to the same formulation as the popular positional encoding that has been popular in neural implicit representations and transformer architectures. As the system of oscillators naturally evolve over time, it would be interesting to investigate whether the models trained this way can be induced to capture the notion of temporal continuity. It is also interesting to investigate the possibility of using multiple temporal scales in the input signals to induce the multiple scales of variation in output content.

Bibliography

[1] Motion-adjustable neural implicit video representation – project website. https://mai-t-long.com/Phase_NIVR/index.html x, 53

[2] This person does not exist. https://thispersondoesnotexist.com/ 17

[3] Agarwala, A., Zheng, K.C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D., Szeliski, R.: Panoramic video textures. In: ACM Transactions on Graphics. p. 821–827 (2005) 14

[4] Almuslamani, H.A.I., Nassar, I.A., Mahdi, O.R.: The effect of educational videos on increasing student classroom participation: Action research. International Journal of Higher Education (2020) 1

[5] Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., Korzhenkov, D.: Image generators with conditionally-independent pixel synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 9, 50, 54

[6] Arias, P., Morel, J.M.: Video denoising via empirical bayesian estimation of space-time patches. Journal of Mathematical Imaging and Vision 60(1), 70–93 (jan 2018) 2

[7] Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. ACM Transactions on Graphics 26(3) (jul 2007) 3

[8] Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision 92(1), 1–31 (2011) 6, 8, 14, 22, 25, 37, 38, 40

[9] Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 48, 82

[10] Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018) 48, 82

[11] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics 28(3) (jul 2009) 3, 4, 15

[12] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: IEEE International Conference on Computer Vision (2021) 20

[13] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: IEEE International Conference on Computer Vision (October 2021) 54

[14] Battaglia, P., Hamrick, J.B.C., Bapst, V., Sanchez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G.E., Vaswani, A., Allen, K., Nash, C., Langston, V.J., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. arXiv (2018) 19

[15] Bemana, M., Myszkowski, K., Seidel, H.P., Ritschel, T.: X-fields: Implicit neural view-, light- and time-image interpolation. ACM Transactions on Graphics 39(6) (nov 2020) 54

[16] Bonneel, N., Tompkin, J., Sunkavalli, K., Sun, D., Paris, S., Pfister, H.: Blind video temporal consistency. ACM Transactions on Graphics (2015) 18

[17] Brame, C.J.: Effective educational videos: Principles and guidelines for maximizing student learning from video content. CBE—Life Sciences Education 15(4), es6 (2016) 1

[18] Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019) 17

[19] Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision. vol. 3024, pp. 25–36 (2004) 14, 37

[20] Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: ACM Transactions on Graphics. p. 425–432 (2001) 12

[21] Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with BM3D? In: IEEE Conference on Computer Vision and Pattern Recognition (2012) 25

[22] Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: European Conference on Computer Vision (2012) 44

[23] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics 32(6), 1309–1332 (dec 2016) 13

[24] Challa, S., Morelande, M., Mušicki, D., Evans, R.: Fundamentals of Object Tracking. Cambridge University Press (2011) 6, 13

[25] Chan, E., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 20

[26] Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 54

[27] Chen, D., Liao, J., Yuan, L., Yu, N., Hua, G.: Coherent online video style transfer. In: IEEE International Conference on Computer Vision (2017) 14

[28] Chen, X., Zhang, Y., Wang, Y., Shu, H., Xu, C., Xu, C.: Optical flow distillation: Towards efficient and stable video style transfer. In: European Conference on Computer Vision (2020) 14

[29] Chen, Y., Liu, S., Wang, X.: Learning continuous image representation with local implicit image function. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 9, 50, 54, 78

[30] Cheng, X., Chen, Z.: Multiple video frame interpolation via enhanced deformable separable convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021) 48

[31] Cho, D., Jung, Y., Rameau, F., Kim, D., Woo, S., Kweon, I.S.: Video retargeting: Trade-off between content preservation and spatio-temporal consistency. In: ACM Internation Conference on Multimedia. p. 882–889 (2019) 2

[32] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (2014) 78

[33] Cinganotto, L., Cuccurullo, D.: The role of videos in teaching and learning content in a foreign language. Journal of E-Learning and Knowledge Society 11, 49–62 (01 2015) 1

[34] Clark, A., Donahue, J., Simonyan, K.: Adversarial video generation on complex datasets. arXiv: Computer Vision and Pattern Recognition (2019) 18

[35] Cohen, M., Szeliski, R.: Lumigraph, pp. 462–467 (01 2014) 12

[36] Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop (2011) 35

[37] Debevec, P.E., Yu, Y., Borshukov, G.: Efficient view-dependent image-based rendering with projective texture-mapping. In: Eurographics Workshop. pp. 105–116 (1998) 13

[38] Deng, Y., Yang, J., Xiang, J., Tong, X.: Gram: Generative radiance manifolds for 3d-aware image generation. In: IEEE Conference on Computer Vision and Pattern Recognition (2022) 20

[39] Denton, E.L., Birodkar, v.: Unsupervised learning of disentangled representations from video. In: Conference on Neural Information and Processing Systems (2017) 18, 21

[40] Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. In: Conference on Neural Information Processing Systems (2021) 17

[41] Didyk, P., Sitthi-amorn, P., Freeman, W.T., Durand, F., Matusik, W.: Joint view expansion and filtering for automultiscopic 3D displays. ACM Transactions on Graphics 32(6), 221:1–221:8 (2013) 25

[42] Dong, C., Deng, Y., Loy, C.C., Tang, X.: Compression artifacts reduction by a deep convolutional network. In: IEEE International Conference on Computer Vision (2015) 25

[43] Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 38(2), 295–307 (2016) 25

[44] Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with

convolutional networks. In: IEEE International Conference on Computer Vision (2015) 25, 29, 37

[45] Dosovitskiy, A., Springenberg, J.T., Brox, T.: Learning to generate chairs with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 25, 44

[46] Drulea, M., Nedevschi, S.: Total variation regularization of local-global optical flow. IEEE Conference on Intelligent Transportation Systems (2011) 14

[47] Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. arXiv/1610.07629 (2016) 25

[48] Dupont, E., Golinski, A., Alizadeh, M., Teh, Y.W., Doucet, A.: COIN: COmpression with implicit neural representations. In: International Conference on Learning Representations Wokshops (2021) 9, 50, 54

[49] Epstein, W., Rogers, S.: Perception of Space and Motion. Academic Press (1995) 5, 13, 15

[50] Finn, C., Goodfellow, I.J., Levine, S.: Unsupervised learning for physical interaction through video prediction. In: Conference on Neural Information and Processing Systems (2016) 27

[51] Fleet, D.J., Jepson, A.D.: Computation of component image velocity from local phase information. International Journal of Computer Vision 5(1), 77–104 (1990) 10, 55

[52] Fleet, D.J., Jepson, A.D.: Computation of component image velocity from local phase information. International Journal of Computer Vision 5(1), 77–104 (sep 1990) 15

[53] Flynn, J., Neulander, I., Philbin, J., Snavely, N.: DeepStereo: Learning to predict new views from the world's imagery. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 25, 26, 29

[54] Freeman, W.T., Adelson, E.H., Heeger, D.J.: Motion without movement. ACM Transactions on Graphics p. 27–30 (1991) 15, 81

[55] Gadot, D., Wolf, L.: PatchBatch: A batch augmented loss for optical flow. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 25, 29

[56] Gafni, G., Thies, J., Zollhofer, M., Niessner, M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 54

[57] Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic view synthesis from dynamic monocular video. In: IEEE International Conference on Computer Vision (2021) 54

[58] Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2016) 4, 25

[59] Gautama, T., Van Hulle, M.A.: A phase-based approach to the estimation of the optical flow field using spatial filtering. IEEE Transactions on Neural Networks 13(5), 1127–1136 (sep 2002) 10, 15, 55

[60] Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2014) 25

[61] Giusti, A., Ciresan, D.C., Masci, J., Gambardella, L.M., Schmidhuber, J.: Fast image scanning with deep max-pooling convolutional neural networks. In: IEEE International Conference on Image Processing (2013) 31, 35

[62] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics (2010) 33

[63] Goold, A.: The Video Editing Handbook: For Beginners. John Goold (2021) 1

[64] Gu, J., Liu, L., Wang, P., Theobalt, C.: Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In: International Conference on Learning Representations (2022) 20

[65] Güney, F., Geiger, A.: Deep discrete flow. In: Asian Conference on Computer Vision. vol. 10114, pp. 207–224 (2016) 25, 29

[66] Haim, N., Feinstein, B., Granot, N., Shocher, A., Bagon, S., Dekel, T., Irani, M.: Diverse video generation from a single video. CoRR (2022) 15, 79

[67] Han, J., Kopp, T., Xu, Y.: An estimation-theoretic approach to video denoising. In: IEEE International Conference on Image Processing. pp. 4273–4277 (2015) 2

[68] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2003) 6

[69] Hassaballah, M., Awad, A.: Deep Learning in Computer Vision: Principles and Applications. CRC Press (2020) 17

[70] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 25

[71] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Conference on Neural Information Processing Systems (2020) 17

[72] Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (11 1997) 78

[73] Horry, Y., Anjyo, K.I., Arai, K.: Tour into the picture: Using a spidery mesh interface to make animation from a single image. In: ACM Transactions on Graphics. p. 225–232 (1997) 13

[74] Huang, H., Wang, H., Luo, W., Ma, L., Jiang, W., Zhu, X., Li, Z., Liu, W.: Real-time neural style transfer for videos. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 14, 19

[75] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. vol. 37 (2015) vii, 30, 31

[76] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 17

[77] Jia, X., Brabandere, B.D., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Conference on Neural Information and Processing Systems (2016) 27

[78] Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (2016) 25

[79] Kalantari, N.K., Wang, T., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Transactions on Graphics 35(6), 193:1–193:10 (2016) 25, 26, 29, 44

[80] Kang, S.B., Li, Y., Tong, X., Shum, H.: Image-based rendering. Foundations and Trends in Computer Graphics and Vision 2(3) (2006) 24

[81] Karayev, S., Trentacoste, M., Han, H., Agarwala, A., Darrell, T., Hertzmann, A., Winnemoeller, H.: Recognizing image style. In: British Machine Vision Conference (2014) 25

[82] Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. In: Conference on Neural Information and Processing Systems (2021) 17, 80, 83

[83] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014) 33

[84] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015) 62, 76

[85] Kokaram, A., Kelly, D., Denman, H., Crawford, A.: Measuring noise correlation for improved video denoising. In: IEEE International Conference on Image Processing (2012) 2

[86] Kostereliogl u, I.: Student views on learning environments enriched by video clips. Universal Journal of Educational Research 4, 359–369 (2016) 1

[87] Krähenbühl, P., Koltun, V.: Efficient nonlocal regularization for optical flow. In: European Conference on Computer Vision (2012) 14

[88] Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Conference on Neural Information and Processing Systems (2012) 25

[89] Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. In: Conference on Neural Information and Processing Systems (2015) 25, 26, 44

[90] Levoy, M., Hanrahan, P.: Light field rendering. In: ACM Transactions on Graphics. p. 31–42 (1996) 12

[91] Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 25

[92] Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 54

[93] Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Transactions on Graphics 20(3), 127–150 (jul 2001) 15

[94] Liao, Z., Joshi, N., Hoppe, H.: Automated video looping with progressive dynamism. ACM Transactions on Graphics 32(4) (jul 2013) 14

[95] Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: IEEE International Conference on Computer Vision (2021) 54, 55

[96] Lin, S.S., Yeh, I.C., Lin, C.H., Lee, T.Y.: Patch-based image warping for content-aware retargeting. IEEE Transactions on Multimedia 15(2), 359–368 (2013) 15

[97] Liu, C., Freeman, W.T.: A high-quality video denoising algorithm based on reliable motion estimation. In: European Conference on Computer Vision. p. 706–719 (2010) 2, 19

[98] Liu, F., Gleicher, M.: Video retargeting: Automating pan and scan. In: ACM Internation Conference on Multimedia. p. 241–250 (2006) 2

[99] Liu, F., Gleicher, M., Jin, H., Agarwala, A.: Content-preserving warps for 3d video stabilization. ACM Transactions on Graphics 28(3) (jul 2009) 2, 14

[100] fu Liu, G., Gao, P., chun Li, Y., ping Zhang, Z.: Research on the influence of social media short video marketing on consumer brand attitude. International Conference on Social Science and Higher Education (2019) 1

[101] Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: European Conference on Computer Vision (September 2018) 4

[102] Liu, S., Yuan, L., Tan, P., Sun, J.: Bundled camera paths for video stabilization. ACM Transactions on Graphics 32(4) (jul 2013) 2, 14

[103] Liu, X., Shi, S., Teixeira, T., Wedel, M.: Video content marketing: The making of clips. Journal of Marketing 82 (04 2018) 1

[104] Long, G., Kneip, L., Alvarez, J.M., Li, H., Zhang, X., Yu, Q.: Learning image matching by simply watching video. In: European Conference on Computer Vision (2016) ix, 21, 26, 32, 43, 44

[105] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 31, 35

[106] Mahajan, D., Huang, F., Matusik, W., Ramamoorthi, R., Belhumeur, P.N.: Moving gradients: A path-based method for plausible image interpolation. ACM Transactions on Graphics 28(3), 42:1–42:11 (2009) 25

[107] Mai, L., Liu, F.: Motion-adjustable neural implicit video representation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 10738–10747 (June 2022) 50

[108] Makansi, O., Ilg, E., Brox, T.: End-to-end learning of video super-resolution with motion compensation. In: German Conference on Pattern Recognition (2017) 14, 19, 82

[109] Marr, D., Ullman, S.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. MIT Press (2010) 5

[110] Martel, J.N.P., Lindell, D.B., Lin, C.Z., Chan, E.R., Monteiro, M., Wetzstein, G.: Acorn: Adaptive coordinate networks for neural scene representation. ACM Transactions on Graphics 40(4) (2021) 54, 78

[111] Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: International Conference on Learning Representations (2016) 32

[112] Matsushita, Y., Ofek, E., Tang, X., Shum, H.Y.: Full-frame video stabilization. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 50–57 (2005) 2

[113] Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representa-

tions. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 9, 51, 54, 78

[114] Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 44

[115] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (2019) 49, 54

[116] Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: Phasenet for video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 6, 55

[117] Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) viii, 8, 16, 22, 25, 37, 38, 81

[118] Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 6, 10, 55

[119] Michaeli, T., Irani, M.: Nonparametric blind super-resolution. In: IEEE International Conference on Computer Vision. pp. 945–952 (2013) 15

[120] Michaeli, T., Irani, M.: Blind deblurring using internal patch recurrence. In: European Conference on Computer Vision (2014) 15

[121] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (2020) 5, 49, 50, 54, 55

[122] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (2020) 20

[123] Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR abs/1411.1784 (2014) 17

[124] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics 41(4), 102:1–102:15 (Jul 2022) 20

[125] Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: Hologan: Unsupervised learning of 3d representations from natural images. In: IEEEInternational Conference on Computer Vision (2019) 5, 19

[126] Niklaus, S., Liu, F.: Context-aware synthesis for video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2018) 82

[127] Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: IEEE Conference on Computer Vision and Pattern Recognition (July 2017) 22

[128] Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: IEEE International Conference on Computer Vision (2017) ix, 46, 47

[129] Ong, E.P., Spann, M.: Robust optical flow computation based on least-median-of-squares regression. International Journal of Computer Vision 31(1), 51–82 (feb 1999) 6

[130] Owens, J., Millerson, G.: Video Production Handbook. Routledge, 6th edn. (2017) 1

[131] Parihar, A.S., Varshney, D., Pandya, K., Aggarwal, A.: A comprehensive survey on video frame interpolation techniques. The Visual Computer 38(1), 295–319 (jan 2022) 48

[132] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 49, 50, 54

[133] Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. IEEE International Conference on Computer Vision (2021) 54

[134] Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 4, 17

[135] Pollak Zuckerman, L., Naor, E., Pisha, G., Bagon, S., Irani, M.: Across scales and across dimensions: Temporal super-resolution using deep internal learning. In: European Conference on Computer Vision (2020) 63

[136] Prashnani, E., Noorkami, M., Vaquero, D., Sen, P.: A phase-based approach for animating images using video examples. Computer Graphics Forum 36(6), 303–311 (sep 2017) 16, 81

[137] Prince, S.: Computer Vision: Models Learning and Inference. Cambridge University Press (2012) 51

[138] Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NeRF: Neural Radiance Fields for Dynamic Scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (2020) 54

[139] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with CLIP latents. CoRR (2022) 18, 80

[140] Ranftl, R., Bredies, K., Pock, T.: Non-local total generalized variation for optical flow estimation. In: European Conference on Computer Vision (2014) 14

[141] Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. arXiv/1611.00850 (2016) 44

[142] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv/1412.6604 (2014) 32

[143] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: International Conference on Machine Learning (2016) 17

[144] Ren, J., Chai, M., Woodford, O.J., Olszewski, K., Tulyakov, S.: Flow guided transformable bottleneck networks for motion retargeting. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 82

[145] Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Epicflow: Edge-preserving interpolation of correspondences for optical flow. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2015) 6

[146] Rubinstein, M., Gutierrez, D., Sorkine, O., Shamir, A.: A comparative study of image retargeting. ACM Transactions on Graphics 29(6) (dec 2010) 3

[147] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding. CoRR abs/2205.11487 (2022) 18, 80

[148] Saito, M., Matsumoto, E., Saito, S.: Temporal generative adversarial nets with singular value clipping. In: IEEE International Conference on Computer Vision (2017) 4, 18

[149] Saito, S., Simon, T., Saragih, J., Joo, H.: Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In: IEEE Conference on Computer Vision and Pattern Recognition (2020) 54

[150] Sanakoyeu, A., Kotovenko, D., Lang, S., Ommer, B.: A style-aware content loss for real-time hd style transfer. In: European Conference on Computer Vision (2018) 4

[151] Schwarz, K., Liao, Y., Niemeyer, M., Geiger, A.: Graf: Generative radiance fields for 3d-aware image synthesis. In: Conference on Neural Information Processing Systems (2020) 54

[152] Schödl, A., Szeliski, R., Salesin, D., Essa, I.: Video textures. ACM Transactions on Graphics 2000 (July 2000) 14

[153] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: OverFeat: Integrated recognition, localization and detection using convolutional networks. In: International Conference on Learning Representations (2013) 25, 31, 35

[154] Shade, J., Gortler, S., He, L.w., Szeliski, R.: Layered depth images. In: ACM Transactions on Graphics. p. 231–242 (1998) 13

[155] Shahar, O., Faktor, A., Irani, M.: Space-time super-resolution from a single video. In: IEEE Conference on Computer Vision and Pattern Recognition (2011) 15

[156] Shoemake, K.: Animating rotation with quaternion curves. In: ACM Transactions on Graphics. p. 245–254 (1985) 66

[157] Shum, H., Chan, S., Kang, S.: Image-Based Rendering. Springer (2008) 12, 13

[158] Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., Sebe, N.: First order motion model for image animation. In: Conference on Neural Information Processing Systems (December 2019) 82

[159] Simoncelli, E.: Local analysis of visual motion. The Visual Neurosciences (1992) 15

[160] Simoncelli, E., Freeman, W.: The steerable pyramid: a flexible architecture for multi-scale derivative computation. In: IEEE International Conference on Image Processing (1995) 15

[161] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv/1409.1556 (2014) 25

[162] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015) 61

[163] Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Conference

on Neural Information and Processing Systems (2020) 9, 49, 50, 51, 54, 55, 62, 78

[164] Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Conference on Neural Information and Processing Systems (2019) 49, 50, 54, 78

[165] Skorokhodov, I., Ignatyev, S., Elhoseiny, M.: Adversarial generation of continuous images. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 9, 54

[166] Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Conference on Neural Information Processing Systems (2019) 17

[167] Sun, J., Cao, W., Xu, Z., Ponce, J.: Learning a convolutional neural network for non-uniform motion blur removal. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 25

[168] Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 25

[169] Svoboda, P., Hradis, M., Barina, D., Zemcík, P.: Compression artifacts removal using convolutional neural networks. arXiv/1605.00366 (2016) 25

[170] Szeliski, R.: Computer Vision: Algorithms and Applications. Springer London (2010) 6, 24, 51

[171] Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: Closing the gap to human-level performance in face verification. In: IEEE Conference on Computer Vision and Pattern Recognition (2014) 25

[172] Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P.P., Barron, J.T., Ng, R.: Learned initializations for optimizing coordinate-based neural representations. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 54, 78

[173] Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Conference on Neural Information and Processing Systems (2020) 49, 50, 51, 54, 55, 78

[174] Tao, M.W., Bai, J., Kohli, P., Paris, S.: SimpleFlow: A non-iterative, sublinear optical flow algorithm. Computer Graphics Forum 31(2), 345–353 (2012) 34

[175] Tassano, M., Delon, J., Veit, T.: Fastdvdnet: Towards real-time deep video denoising without flow estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2020) 2

[176] Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3D models from single images with a convolutional network. In: European Conference on Computer Vision (2016) 25, 26, 44

[177] Teed, Z., Deng, J.: RAFT: recurrent all-pairs field transforms for optical flow. In: European Conference on Computer Vision (2020) 6

[178] Tekalp, A.M.: Digital Video Processing. Prentice-Hall, Inc. (1995) 6, 13

[179] Teney, D., Hebert, M.: Learning to extract motion from videos in convolutional neural networks. arXiv/1601.07532 (2016) 25, 29

[180] Thompson, R., Bowen, C.: Grammar of the Edit. Bitacora de retórica, Focal Press (2009) 1

[181] Thompson, W., Fleming, R., Creem-Regehr, S., Stefanucci, J.: Visual Perception from a Computer Graphics Perspective. Taylor & Francis (2011) 5, 13, 15

[182] Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., Paluri, M.: Deep End2End Voxel2Voxel prediction. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (2016) 25, 29

[183] Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In: IEEE International Conference on Computer Vision (2021) 54

[184] Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d representation and rendering. In: IEEE Conference on Computer Vision and Pattern Recognition (October 2021) 54

[185] Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2018) 4, 18

[186] Tulyakov, S., Gehrig, D., Georgoulis, S., Erbach, J., Gehrig, M., Li, Y., Scaramuzza, D.: TimeLens: Event-based video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2021) 48

[187] Ugail, H.: Deep Learning in Visual Computing Explanations and Examples. CRC Press (2022) 17

[188] Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.S.: Texture networks: Feed-forward synthesis of textures and stylized images. In: International Conference on Machine Learning (2016) 25

[189] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 19

[190] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Conference on Neural Information Processing Systems (2017) 54, 55, 78

[191] Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Conference on Neural Information and Processing Systems (2016) 4, 18

[192] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. ACM Transactions on Graphics 32(4) (2013) 6, 10, 55, 68

[193] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. ACM Transactions on Graphics 32(4), 80:1–80:10 (2013) 16, 25

[194] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. ACM Transactions on Graphics 32(4) (jul 2013) 81

[195] Wadhwa, N., Wu, H.Y., Davis, A., Rubinstein, M., Shih, E., Mysore, G.J., Chen, J.G., Buyukozturk, O., Guttag, J.V., Freeman, W.T., Durand, F.: Eulerian

video magnification and analysis. Commununication of the ACM 60(1), 87–95 (dec 2016) 10, 55, 68

[196] Wadhwa, N., Wu, H.Y., Davis, A., Rubinstein, M., Shih, E., Mysore, G.J., Chen, J.G., Buyukozturk, O., Guttag, J.V., Freeman, W.T., Durand, F.: Eulerian video magnification and analysis. Commununication of the ACM 60(1), 87–95 (dec 2016) 16

[197] Wang, L., Guo, Y., Liu, L., Lin, Z., Deng, X., An, W.: Deep video super-resolution using hr optical flow estimation. IEEE Transactions on Image Processing 29, 4323–4336 (2020) 14, 82

[198] Wang, T.C., Liu, M.Y., Zhu, J.Y., Liu, G., Tao, A., Kautz, J., Catanzaro, B.: Video-to-video synthesis. In: Conference on Neural Information and Processing Systems. vol. 31 (2018) 4, 82

[199] Watanabe, T.: High-level Motion Processing: Computational, Neurobiological, and Psychophysical Perspectives. CogNet (1998) 5, 13

[200] Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: Large displacement optical flow with deep matching. In: IEEE International Conference on Computer Vision (2013) 6, 25, 29, 37

[201] Werlberger, M., Pock, T., Unger, M., Bischof, H.: Optical flow guided TV-L 1 video interpolation and restoration. In: Energy Minimization Methods in Computer Vision and Pattern Recognition. vol. 6819, pp. 273–286 (2011) 6, 14, 25

[202] Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: IEEE Conference on Computer Vision and Pattern Recognition (2004) 15

[203] Wolf, L., Guttmann, M., Cohen-Or, D.: Non-homogeneous content-driven video-retargeting. pp. 1–6 (2007) 2

[204] Wu, H., Rao, K.: Digital Video Image Quality and Perceptual Coding. CRC Press (2017) 6, 13

[205] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition (2015) 25

[206] Xia, X., Zhang, M., Xue, T., Sun, Z., Fang, H., Kulis, B., Chen, J.: Joint bilateral learning for real-time universal photorealistic style transfer. In: European Conference on Computer Vision (2020) 4

[207] Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 54

[208] Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: Conference on Neural Information and Processing Systems (2012) 25

[209] Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(9), 1744–1757 (2012) 37

[210] Xu, L., Ren, J.S.J., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: Conference on Neural Information and Processing Systems (2014) 25

[211] Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In: Conference on Neural Information and Processing Systems (2019) 54

[212] Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision 127(8), 1106–1125 (aug 2019) 14

[213] Xue, T., Wu, J., Bouman, K.L., Freeman, B.: Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In: Conference on Neural Information and Processing Systems (2016) 27

[214] Yang, J., Reed, S.E., Yang, M., Lee, H.: Weakly-supervised disentangling with recurrent transformations for 3D view synthesis. In: Conference on Neural Information and Processing Systems (2015) 25, 26

[215] Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys 38(4) (dec 2006) 6, 13

[216] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Conference on Neural Information and Processing Systems (2014) 25

[217] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 4

[218] Yu, Z., Li, H., Wang, Z., Hu, Z., Chen, C.W.: Multi-level video frame interpolation: Exploiting the interaction among different levels. IEEE Transactions on Circuits and Systems for Video Technology 23(7), 1235–1248 (2013) 6, 14, 25

[219] Zhang, C., Chen, T.: A survey on image-based rendering - representation, sampling and compression. Signal Processing: Image Communication 19(1), 1–28 (2004) 24

[220] Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv:2010.07492 (2020) 54

[221] Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European Conference on Computer Vision (2016) 25

[222] Zhou, B., Lapedriza, À., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Conference on Neural Information and Processing Systems (2014) 25

[223] Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: European Conference on Computer Vision (2016) 25, 26, 29, 44

[224] Zhu, J., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: European Conference on Computer Vision (2016) 25

[225] Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S.A.J., Szeliski, R.: High-quality video view interpolation using a layered representation. ACM Transactions on Graphics 23(3), 600–608 (2004) 37

[226] Zontak, M., Irani, M.: Internal statistics of a single natural image. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 977–984 (2011) 15

[227] Zontak, M., Mosseri, I., Irani, M.: Separating signal from noise using patch recurrence across scales. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1195–1202 (2013) 15