Institute of Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelor's Thesis

Motion-compensated Frame Interpolation using Multiple Frames

Yaroslava Nalivayko

Course of Study:	Informatik
Examiner:	Prof. Dr. Andrés Bruhn
Supervisor:	Lukas Mehl, M.Sc.
Commenced:	2021-31-03
Completed:	2021-30-09

Abstract

Recent surge of interest towards increasing frame rate of existing videos to display the information with slow smooth motion has led to the demand for qualitative frame interpolation approaches that create spatially and temporally coherent intermediate frames. Traditional methods usually use only two adjacent frames to estimate the motion trajectories linearly, generally failing to overcome challenges like occlusion and non-linear motion.

In this thesis, we introduce basic concepts regarding motion-compensated frame interpolation including optical flow, warping, splatting and inpainting and present an extension of linear forward and forward forward warping [5] that uses arbitrary number of frames to approximate the motion trajectories in a polynomial way. We explore the difficulties of using optical flows over more distant frames and compare the use of sequentially warped optical flows with the direct variants.

The best performing approach of this work named quadratic forward forward warping approximates forward and backward motion trajectories as parabolas, using three frames for every direction and requiring four frames in total, and utilizes a modification designed specifically to reduce warping artifacts. It achieves better results than the linear approaches on the high frame rate version of Sintel dataset [20] proving that leveraging additional temporal information benefits frame interpolation. The use of more than three frames, however, is not beneficial and leads to decreasing performance.

Kurzfassung

Das aktuell gestiegene Interesse an der Erhöhung der Bildrate von existierenden Videos, um die Informationen mit langsamer glatter Bewegung anzuzeigen, hat zur Nachfrage nach qualitativen Bildinterpolationsansätzen geführt, die räumlich und zeitlich kohärente Zwischenbilder erzeugen. Herkömmliche Methoden verwenden meistens nur zwei benachbarte Frames, um die Bewegungsbahnen linear zu schätzen, und bewältigen in der Regel die Herausforderungen wie Okklusion und nichtlineare Bewegung nicht.

In dieser Arbeit stellen wir grundlegende Konzepte für die bewegungsbasierte Bildinterpolation einschließlich optischen Flusses, Warping, Splatting und Inpainting vor und präsentieren eine Erweiterung der linearen Forward und Forward Forward Warping [5], die eine beliebige Anzahl von Frames verwendet, um die Bewegungstrajektorien polynomisch anzunähern. Wir untersuchen die Schwierigkeiten bei der Verwendung optischer Flüsse über weiter entfernte Frames und vergleichen die Verwendung sequenziell gewarpten optischer Flüsse mit den direkten Varianten.

Der leistungsstärkste Ansatz dieser Arbeit heißt quadratisches Forward Forward Warping, approximiert Vorwärts- und Rückwärtsbewegungsbahnen als Parabeln, benutzt drei Frames für jede Richtung, erfordert vier Frames insgesamt und nutzt eine Modifikation, die speziell für die Reduktion der Warping-Artefakte entwickelt wurde. Dieser Ansatz erzielt bessere Ergebnisse als die linearen Methoden auf der Version vom Sintel Datensatz mit höherer Bildrate [20] und beweist, dass die Nutzung zusätzlicher zeitlicher Informationen für Bildinterpolation von Vorteil ist. Die Verwendung von mehr als drei Frames ist jedoch nicht vorteilhaft und führt zu einer abnehmenden Leistung.

Contents

1 Introduction

1.1 Motivation

Video frame interpolation is a classic problem in computer vision with various practical applications. While it is nowadays possible to create a qualitative high frame rate video without expensive professional cameras, the vast majority of existing videos has the standard frame rate of 24 frames-per-second. Retiming or slow-motion effect creation for such videos thus requires generation of intermediate frames that are spatially and temporarily coherent. The videos with higher frame rate positively influence human perception [25, 26], help to reduce judder, smooth motion blur and enhance visual quality for low response time devices, such as liquid crystal displays [27]. Intermediate frame generation can also be used for video compression by saving only a part of frames and interpolating the others [48, 51]. It can be used to generate training data to learn how to synthesize motion blur [7] or as a measure of how well the motion estimation methods can be applied for intermediate frames prediction to compare these methods [3].

The use of motion information has led to significant progress in the frame interpolation field. Usually, the motion between two consecutive frames is used to interpolate an intermediate frame between them. However, this requires a general assumption that the motion between two frames can be represented by a linear function. This thesis therefore introduces different ways of using three or more frames for frame interpolation and focuses on answering the following questions:

- In which ways can multiple frames be used for frame interpolation?
- Can leveraging additional temporal information increase the interpolant quality?
- What are the difficulties in using multiple frames and how can we overcome them?

1.2 Related Works

In the recent years, several frame interpolation methods were proposed in order to handle difficult scenarios caused by significant appearance changes or large motion and achieve better performance and efficiency. These methods can be roughly divided into three groups and will be briefly introduced in the following.

Phase-Based Frame Interpolation Instead of using estimated motion between frames, phase-based methods represent motion as the phase shift of individual pixels and generate intermediate frames by per-pixel phase modification [35]. The later phase-based approach, PhaseNet, is based on the advances in the deep learning and contains a neural network decoder that directly estimates the phase decomposition of the intermediate frame and helps to cope with larger motion, which was the key limitation of the earlier method [34]. The phase-based approaches compare favorably to other frame interpolation methods, do not require explicit motion estimation and are thus computationally very efficient, while being easy to implement and parallelize.

Kernel-Based Frame Interpolation Similarly to phase-based methods, kernel-based methods avoid using explicitly estimated motion. Instead, they calculate per-pixel spatially-adaptive convolutional kernels and convolve them with the input frame to predict future frames [14, 21, 49] or generate intermediate frames [36]. However, these kernels need to be large to handle large motion and this condition leads to drastic memory demand, thus an approach using separable kernels was introduced later to reduce the number of kernel parameters and memory consumption [37]. Kernel-based methods have achieved excellent performance, but still have difficulties with complex scenarios containing motion blur or light changes.

Motion-Based Frame Interpolation Motion-based methods are by far the most popular frame interpolation methods and usually leverage motion information in form of an optical flow, since motion estimation has achieved great progress in the last decades. However, estimating optical flow remains a challenging problem due to large motion, occlusion, brightness change and motion blur and many motion-based methods were specifically designed to account for the optical flow weaknesses [4, 42]. Most of motion-based methods focus on using two adjacent frames to create intermediate frames between them [22,24,39,47] and only a few approaches exploit temporal information from additional frames [12, 13, 46, 52]. The past few years have seen a rise in interest in using more than two frames for frame interpolation. The approach described in this work also belongs to the motion-based methods using multiple frames and is based on the research conducted in [5] where different ways of interpolation using optical flow were discussed and compared.

1.3 Outline

The following chapter provides relevant concepts required for understanding the basics of optical flow and frame interpolation. Chapter 3 introduces frame interpolation methods that use only two consecutive frames. Chapter 4 builds on the methods from the previous chapter and extends them for the case of multiple frames. Then in Chapter 5 the multi-frame techniques are evaluated and compared to methods that use two frames, and finally, the summary and outlook are given in the last chapter.

2 Background

This chapter contains the basic concepts required for foundational understanding of the research goals and solutions presented in the following chapters. Firstly, the reason for using motion compensation techniques in frame interpolation will be discussed, followed by the explanation of how it can be applied. The core technique needed for motion-compensated frame interpolation will be introduced along with its main problems.

2.1 Motion Estimation

The focus in this section lays on the motion. Firstly, the motivation for integrating motion into the frame interpolation process will be given. Afterwards, the representation of motion in the form of optical flow and a state-of-the-art method for motion estimation will be introduced.

2.1.1 Motivation

Motion is an important feature of image sequences that describes the position change of objects in the scene and helps to understand its dynamics. Motion information is widely used in computer vision for various purposes. It can help segmenting image sequences into independently moving regions [38] or recognizing an object by its characteristical motion features [10]. It can also be used for frame interpolation that is being researched in this work.

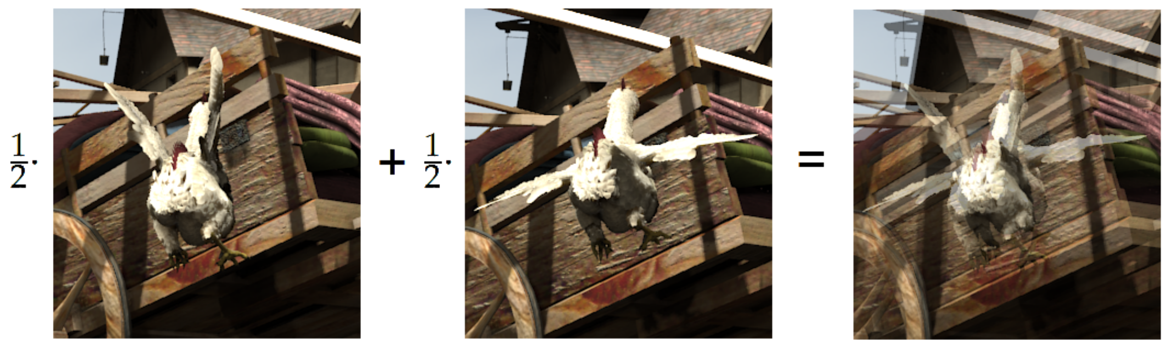


Fig. 2.1: A naive frame interpolation method without considering motion creates noticeable duplication artifacts.

The knowledge of motion between frames is crucial for creating high-quality interpolants. A naive approach without considering motion generates images with eye-catching duplication artifacts (see Fig. 2.1). The task of determining the intermediate position between two known points thus requires knowledge about the object motion which is typically used in computer vision in the form of the so-called *optical flow*.

2.1.2 Optical Flow

The concept of optical flow was first introduced by the American psychologist James J. Gibson as a description of visual stimuli animals percept while moving [16]. This term was later adapted in computer vision as the interframe per-pixel displacement field between two consecutive frames. It is an essential tool for many image processing problems and is used in various fields of application such as object detection and tracking [1], video object segmentation [11], robot navigation [41], and video compression [28]. Frame interpolation is one of the problems where significant progress was made due to the use of optical flow.

The importance of optical flow estimation has led to extensive research in this area over the last four decades. The first traditional approaches, which suggest handcraft feature evaluation, such as the Lucas-Kanade method [29] or the Horn-Schunck method [18], are currently effectively outperformed by newer approaches that use convolutional neural networks and are based on deep learning techniques [40]. However, the task of computing the optical flow remains unsolved because of various difficulties such as large object motion and occlusions caused by overlapping objects.

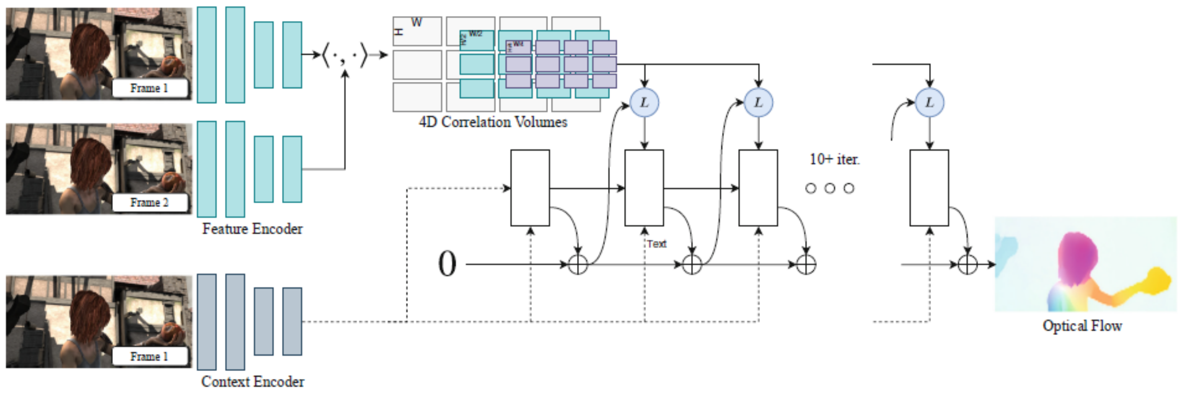


Fig. 2.2: Illustration of RAFT structure. Image source [43].

The optical flow estimation method chosen for this work is a deep network architecture called Recurrent All-Pairs Field Transforms (RAFT). It tries to combine the known concepts introduced by traditional optimization-based methods with the performance and learnability of deep learning methods. RAFT consists of three main parts that were designed to simulate corresponding processes in traditional approaches: (1) the feature encoder that extracts per-pixel features such as pixel intensity, edges, changes in brightness, etc.; (2) the correlation layer that computes visual similarity between pixel-wise features; (3) the update operator that mimics the steps of iterative optimization. The RAFT architecture is illustrated in more detail in Fig. 2.2.

Fig. 2.3 compares flow predictions by RAFT and two other approaches. Since RAFT approach showed an excellent performance on optical flow benchmarks, in the following it will be used to compute optical flow applied for the frame interpolation.

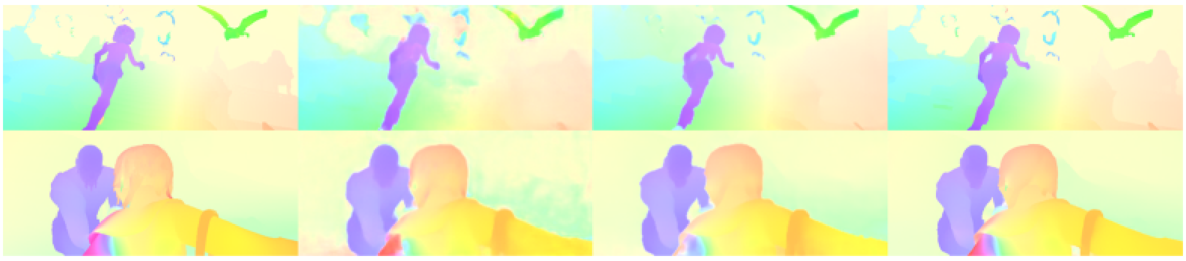
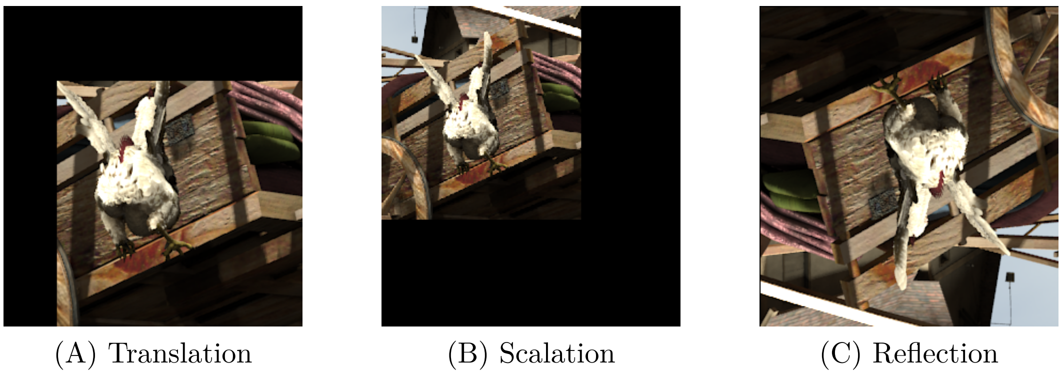


Fig. 2.3: Comparison of flow predictions on two sequences of the Sintel dataset [9]. *From left to right:* ground truth, VCN [50], IRR-PWC [19], RAFT. Image source [43].

2.2 Warping

Image warping is a common technique in image processing and can be used in various fields such as perspective transformation and removing or adding optical distortions. Warping is a specific type of transformation which maps the points without changing the colors. This can be based on any function from one image plane to a second plane $W : \mathbb{R}^2 \to \mathbb{R}^2$. The warping function W can be chosen to create different effects, e.g., translation, rotation, scaling, reflection, etc. (more details in [17]). Some effects are visualized in Fig. 2.4.



(A) Translation (B) Scalation (C) Reflection

Fig. 2.4: Visualization of different effects created by warping. Black areas indicate undefined pixels.

There are two approaches in creating a transformed image using warping. The more obvious one called *forward warping* iterates over all pixels $\vec{x} = (x, y)$ of the source image assigning them to some pixels of the target image according to the transformation W:

$$I_{target}(W(\vec{x})) = I_{source}(\vec{x}). \tag{2.1}$$

The other approach is applicable in the case of an injective warping function and is called *backward warping*. Unlike forward warping, it iterates over all pixels in the target image and assigns source values to them according to the inverse transformation W^{-1}:

$$I_{target}(\vec{x}) = I_{source}(W^{-1}(\vec{x})). \tag{2.2}$$

Although the backward warping may seem to be more promising because of the fact that the iteration goes over all target pixels leaving no undefined places, forward warping is claimed to have better results to a certain degree [5] and is thus used in this work.

Image warping is an important part for the frame interpolation along with the optical flow estimation. Together they build the basis for the core technique of motion-compensated frame interpolation which is introduced in the next section.

2.3 Frame Interpolation

Frame interpolation aims to create an intermediate spatially and temporarily coherent frame between two consecutive input frames denoted as I_0 and I_1. The goal of this section is to give a foundational explanation of warping using optical flow and introduce possible difficulties. The color information from I_0 will be moved according to the optical flow to reconstruct the frame I_1.

2.3.1 Optical Flow as Warping Function

This subsection addresses the use of optical flow introduced in the last section as the function for warping. The optical flow that represents the motion from frame I_0 to frame I_1 is denoted as $\vec{v}_{0\to1}$. Although the warping can be analogously done with the use of the opposite flow $\vec{v}_{1\to0}$, only the flow $\vec{v}_{0\to1}$ is going to be used in this chapter for the sake of consistency.

Since optical flow contains per-pixel displacements and pixel positions are usually integer, the following restriction on the warping function naturally appears:

$$W_{flow} : \mathbb{N}^2 \to \mathbb{R}^2. \tag{2.3}$$

The optical flow often contains sub-pixel displacements, thus the values are still real-valued.

Since the optical flow depicts the displacements, its use for the warping function strongly resembles translation. The distinctive feature of optical flow is that the displacement is not uniform for the whole image but pixelwise and is given for $\vec{x} = (x, y)$ as the optical flow value for this position $\vec{v}_{0\to1}(\vec{x})$:

$$W_{flow}(\vec{x}) = \vec{x} + \vec{v}_{0\to1}(\vec{x}). \tag{2.4}$$

2.3.2 Forward Warping

Forward warping is the core technique for motion-compensated frame interpolation and will be used as the basis for various interpolation methods in the following chapters. For the optical flow $\vec{v}_{0\to1}$ that describes motion from frame I_0 to frame I_1, the forward warping equation is given as follows:

$$I_1(\vec{x}_0 + \vec{v}_{0\to1}(\vec{x}_0)) = I_0(\vec{x}_0). \tag{2.5}$$

Fig. 2.5 visualizes the forward warping. The image I_0 serves as the source of color values that are warped from the positions $\vec{x}_0$ with the use of the optical flow $\vec{v}_{0\to1}$ to the target positions $\vec{x}_1$ of the image I_1. This way the color values are moved along the time axis according to the detected motion between frames.

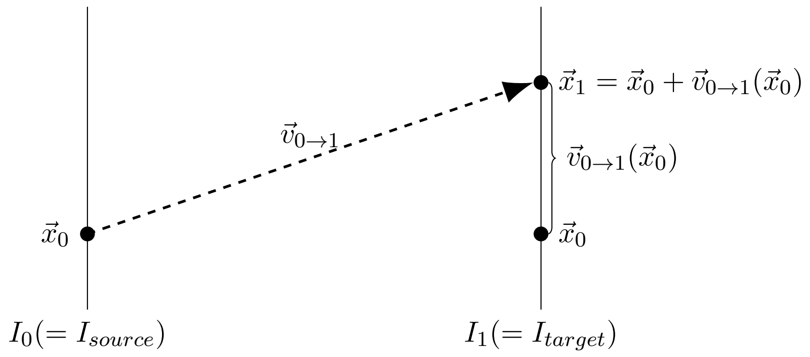

$\vec{x}_1 = \vec{x}_0 + \vec{v}_{0\to1}(\vec{x}_0)$
$\vec{v}_{0\to1}$
$\vec{v}_{0\to1}(\vec{x}_0)$
$\vec{x}_0$
$\vec{x}_0$
$I_0(= I_{source})$
$I_1(= I_{target})$

Fig. 2.5: Visualization of forward warping with the optical flow $\vec{v}_{0\to1}$. The images I_0,I_1 are schematically represented in 1D.

The real-valued nature of optical flow values leads to the fact that the target positions $\vec{x}_1 = \vec{x}_0 + \vec{v}_{0\to1}(\vec{x}_0)$ can lay between integer pixel locations. Thus, the warped color value can be *splatted* among the four nearest neighbors. The positions of these neighbors are always given as follows:

$$\vec{x}_{0,0} = \lfloor \vec{x}_1 \rfloor \qquad \vec{x}_{1,0} = \lfloor \vec{x}_1 \rfloor + (1,0)^T$$
$$\vec{x}_{0,1} = \lfloor \vec{x}_1 \rfloor + (0,1)^T \qquad \vec{x}_{1,1} = \lfloor \vec{x}_1 \rfloor + (1,1)^T$$

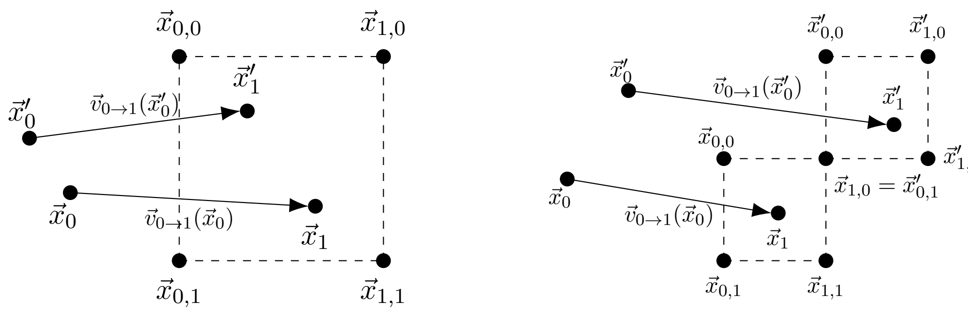
where $\lfloor \cdot \rfloor$ denotes rounding down.

Alternatively, assigning the warped color value only to the nearest neighbor pixel gives less blurred results, however, it also leads to more target pixel values being undefined. The strategy of distributing the warped color value to all four neighbors called *4N* has been proven to be more effective for frame interpolation [5] and is thus the splatting method used in this work.

Because of the real-valued nature of optical flow, the *4N* often leads to situations where a pixel is influenced by several warped color values and its resulting color value thus needs further considerations called collision handling. A pixel on the other hand can also have no warped color values influencing it and is left undefined. These problems will be discussed in the following subsections.

2.3.3 Collision Handling

The splatting in forward warping can lead to collisions in color values. They can appear when the warped target positions $\vec{x}_1$ and $\vec{x}'_1$ in I_1 of two source positions in I_0 have the same four neighbors (see Fig. 2.6 left). The other case of collisions is created by overlapping of the neighbors (see Fig. 2.6 right). The collision can naturally have more than two warped color values involved.



(A) Identical neighbours (B) Overlapping neighbours

Fig. 2.6: Left: Collision case with the same four neighbors for two different warped positions $\vec{x}_1$ and $\vec{x}'_1$. Right: Collision case with overlapping neighbors. The right top neighbor of $\vec{x}_1$ is the left bottom neighbor of $\vec{x}'_1$ and its color value is thus affected by color values of both $\vec{x}_0$ and $\vec{x}'_0$.

In the weighted variant of collision handling, the influence on the result pixel color value depends on its distance to the warped position, motivated by bilinear interpolation [23]. These distances are graphically illustrated in Fig. 2.7. The weights for the four neighbors are given as follows:

$$\begin{aligned} w_{0,0} &= (1-d_x)\cdot(1-d_y) & w_{1,0} &= d_x\cdot(1-d_y) & \begin{pmatrix} d_x \\ d_y \end{pmatrix} &= \vec{x}_1 - \lfloor \vec{x}_1 \rfloor. \\ w_{0,1} &= (1-d_x)\cdot d_y & w_{1,1} &= d_x\cdot d_y \end{aligned} \tag{2.6}$$

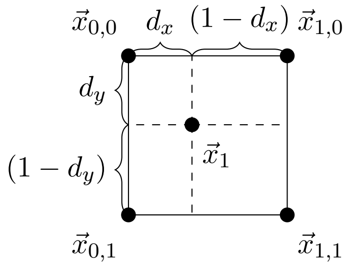


Fig. 2.7: Illustration of the distances d_x and d_y used in Equation (2.6) for weights calculation. Note that the weight of a pixel corresponds to the area of the opposite rectangle.

The result of summing up the weights from all warped positions that are influencing the pixel is not necessarily equal to 1. Thus, the next step after adding up the corresponding color values is the normalization of the result.

Algorithm 1 describes the splatting and collision handling process. Firstly, two arrays are initialized with zero values, the first one holds the information about summed up color values (array I_1), the other one is for keeping the summed up weights (array W). In a `for` loop over all pixels $\vec{x}_0$ of the image I_0 the warped positions $\vec{x}_1$ are determined. Then for every neighbor $\vec{x}_{i,j}$ of $\vec{x}_1$ the corresponding weight $w_{i,j}$ is calculated. The warped color values $I_0[\vec{x}_0]$ are weighted according to $w_{i,j}$ and added to the accumulated color value in $I_1[\vec{x}_{i,j}]$. The weight $w_{i,j}$ is summed up in $W[\vec{x}_{i,j}]$. Finally, the normalization takes place in another `for` loop. It iterates over all pixels $\vec{x}_1$ of the image I_1 and the color values $I_1[\vec{x}_1]$ are normalized due to the division by the according accumulated weight value $W[\vec{x}_1]$. The places where $W[\vec{x}_1]$ is equal to zero are marked as undefined and have to be filled in the next step called *inpainting*.

Algorithm 1: Forward warping with 4N splatting and weighted collision handling

```
Input: Image $I_0$ and optical flow $\vec{v}_{0\to 1}$
Output: Image $I_1$

Initialize an empty array $I_1$                    //summed up weighted color values
Initialize an empty array $W$                                 //summed up weights

foreach position $\vec{x}_0$ in $I_0$ do
    $\vec{x}_1 := \vec{x}_0 + \vec{v}_{0\to 1}(\vec{x}_0)$
    foreach $i \in \{0,1\}, j \in \{0,1\}$ do
        Calculate $\vec{x}_{i,j}$ and $w_{i,j}$ according to $\vec{x}_1$
        $I_1[\vec{x}_{i,j}] := I_1[\vec{x}_{i,j}] + I_0[\vec{x}_0] \cdot w_{i,j}$
        $W[\vec{x}_{i,j}] := W[\vec{x}_{i,j}] + w_{i,j}$
    end
end
foreach position $\vec{x}_1$ in $I_1$ do
    if $W[\vec{x}_1] > 0$ then
        $I_1[\vec{x}_1] := I_1[\vec{x}_1] / W[\vec{x}_1]$
    else
        Mark $\vec{x}_1$ as undefined
    end
end
```

2.4 Inpainting

As mentioned in the previous subsection, some pixels in the created result are left undefined leading to holes in the image. Although the color values could not be found during the warping process, they need to be determined with the help of defined pixels to obtain a complete image.

The process of closing such holes is called *inpainting*. It considers the information from the neighbors of undefined pixels and concludes from that which color value should be inserted in empty places (see Fig. 2.8).

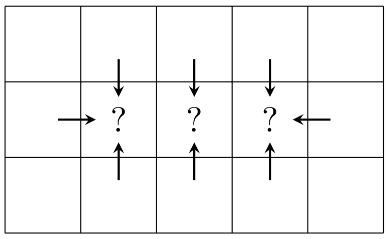


Fig. 2.8: Visualization of inpainting. The question mark denotes the pixels that are left undefined during the warping process and need to be filled with the help of color information from the neighboring pixels.

2.4.1 Diffusion

The inpainting process is inspired by the diffusion which describes the movement of substance from a region of higher concentration to the region of lower concentration. The spreading out substance in inpainting is represented by the color values. However, there is no quantitative difference in concentration between different positions, since the pixels in inpainting either have a color value or not.

Let f represent a continuous image with the domain Ω. This image is divided into region Ω_k which contains available color values and region Ω_{empty} of undefined pixels, $\Omega_k + \Omega_{empty} = \Omega$. The goal of inpainting is to close the holes and create a filled image u.

The main difference from the diffusion is that the region Ω_k should stay constant, i.e., the color information from Ω_k should be used to fill Ω_{empty} but should not leave Ω_k, which results in the following condition for u:

$$u(x,y) = f(x,y), \quad (x,y)^T \in \Omega_k. \tag{2.7}$$

The other condition concerns the behavior at the border of the image $\partial\Omega$ to model a closed medium and do not let color information leave the image area:

$$\vec{n}^T \nabla u(x,y) = 0, \quad (x,y)^T \in \partial\Omega, \tag{2.8}$$

where $\vec{n}$ denotes the normal vector and $\nabla u = (u_x, u_y)^T$ is the gradient of u.

The inpainting of the remaining pixels is described with the diffusion equation as follows:

$$\Delta u(x,y) = 0, \quad (x,y)^T \in \Omega_{empty}, \tag{2.9}$$

where $\Delta u = u_{xx} + u_{yy}$ corresponds to the Laplace operator. This equation models the diffusion process by verifying that the differences between color values in the holes are small enough, since the partial derivatives of the image u in x- and y-directions are close to zero. The regions of image f in inpainting are visualized in Fig. 2.9.

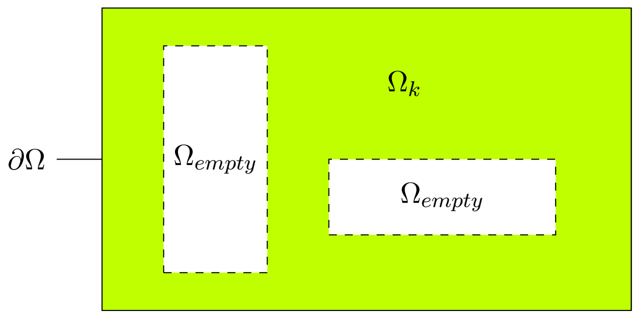


Fig. 2.9: The image regions in inpainting.

2.4.2 Discretisation

Diffusion Equation (2.9) contains two partial derivatives and is thus a partial differential equation (PDE) that needs to be solved numerically. This subsection focuses on the numerical approximation for this equation that leads to an equation system that can be iteratively solved with Jacobi method or Gauss-Seidel method.

Diffusion Equation (2.9) is discretized by central finite differences for every discrete position $(i, j) \in \Omega_{empty}$ as follows:

$$\begin{aligned} 0 &= u_{xx}(i,j) + u_{yy}(i,j) \\ &= \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_{empty}}(i,j)} \frac{u_{\tilde{i},\tilde{j}} - u_{i,j}}{h} + \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_k}(i,j)} \frac{f_{\tilde{i},\tilde{j}} - u_{i,j}}{h}. \end{aligned} \tag{2.10}$$

h denotes the step size and $N_4(i, j)$ is the set of available direct neighbors for the position (i, j). This set can have up to 4 elements depending on the location of (i, j) (corner, edge or in the middle) and whether the neighbor pixels are defined. The superscript of N_4 denotes which domain, Ω_k or Ω_{empty}, is used.

There are $|\Omega_{empty}|$ instances of the introduced equation which together build an equation system of the form $Ax = b$. x is a one-dimensional vector of the size Ω_{empty} which contains all variables $u_{i,j}$ to be found. b is a one-dimensional vector of the same size containing zeros. A is the equation matrix of the size $\Omega_{empty} \times \Omega_{empty}$ that contains the weights for neighbors of $u_{i,j}$ in the corresponding row. This equation system can grow very big depending on the number of undefined pixels in the image and is solved with relatively great expense.

Since A has up to 4 entries in every row and other elements are zeros, A is a sparse matrix and can be solved iteratively. For that, Equation (2.10) is reformulated for $h = 1$ by setting the sums apart as follows:

$$\begin{aligned} 0 &= \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_{empty}}(i,j)} (u_{\tilde{i},\tilde{j}} - u_{i,j}) + \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_k}(i,j)} (f_{\tilde{i},\tilde{j}} - u_{i,j}) \\ &= \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} - \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_{empty}}(i,j)} u_{i,j} + \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}} - \sum_{(\tilde{i},\tilde{j})\in N_4^{\Omega_k}(i,j)} u_{i,j} . \end{aligned} \tag{2.11}$$

Then the sums of $u_{i,j}$ are brought to the left side and merged since $\Omega_{empty} + \Omega_k = \Omega$:

$$\begin{aligned}
\sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_{empty}}(i,j)} u_{i,j} + \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} u_{i,j} &= \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}} \\
\sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega}(i,j)} u_{i,j} &= \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}} \qquad (2.12) \\
\left| N_4^{\Omega}(i,j) \right| \cdot u_{i,j} &= \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}} \ .
\end{aligned}$$

With a final division by $\left| N_4^{\Omega}(i,j) \right|$, Equation (2.11) results in an iterative Jacobi scheme, where the solution can be approximated step by step as follows:

$$u_{i,j}^{k+1} = \frac{\sum\limits_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum\limits_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}}}{|N_4^{\Omega}(i,j)|}. \qquad (2.13)$$

This iterative approach leads to the gradual inpainting from outside towards the center of the holes.

2.4.3 Acceleration

The number of iterations is crucial for the time required for frame interpolation. The Jacobi scheme introduced in the previous subsection fills the holes with a constant speed of one pixel per iteration and thus needs Ω_{empty} iterations which leads to a rather long inpainting in the case of large holes. We are generally interested in speeding up the inpainting process and thus switch from the Jacobi method to the Gauß-Seidel method.

Unlike the Jacobi method where only the values from the previous iteration are used, the Gauß-Seidel method uses the latest values as soon as they are updated and is thus expected to yield faster convergence. The new solution for the position (i, j) has the following form:

$$u_{i,j\ GS}^{(k+1)} = \frac{\sum\limits_{(\tilde{i},\tilde{j}) \in N_{4,-}^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum\limits_{(\tilde{i},\tilde{j}) \in N_{4,+}^{\Omega_{empty}}(i,j)} u_{\tilde{i},\tilde{j}} + \sum\limits_{(\tilde{i},\tilde{j}) \in N_4^{\Omega_k}(i,j)} f_{\tilde{i},\tilde{j}}}{|N_4^{\Omega}(i,j)|}. \qquad (2.14)$$

The sets $N_{4,-}$,$N_{4,+}$ contain not updated neighbor values from the previous iteration and updated values accordingly.

The last acceleration improvement for the inpainting process is made by converting the Gauß-Seidel equation to the successive over-relaxation (SOR) variant, which results in an even faster convergence. We use a fixed (not optimized) relaxation parameter $\omega = 1.95$ which results in the following equation:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + 1.95 \cdot (u_{i,j\ GS}^{(k+1)} - u_{i,j}^{(k)}). \qquad (2.15)$$

The use of SOR can lead to temporarily invalid color values and requires a sufficient number of iterations to ensure a converged result. The equations in this chapter are given for a single color channel and can be thus used during inpainting for every channel separately.

3 Frame Interpolation using Two Frames

This chapter addresses the task of intermediate frame interpolation with the use of techniques presented in Chapter 2. The frame interpolation methods that are being researched in this work are motion-compensated and the source for motion information is the optical flow. Therefore, the first step for all methods in this work is to estimate one or several optical flows needed for frame interpolation according to the method used. The next step is to generate a frame interpolant along the optical flow using the basic forward warping technique introduced in the previous chapter. This step can contain several substeps and multiple use of forward warping and is the main focus of this chapter. As explained in Chapter 2, the frame interpolant often has undefined places as a side effect of warping and has to undergo a final processing step named inpainting. The frame interpolation process is illustrated in Fig. 3.1.

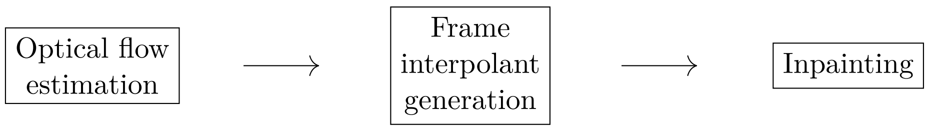


Fig. 3.1: Three main steps of motion-compensated frame interpolation.

Notation The frames in this chapter are notated as I_t. The parameter $t \in [0,1]$ indicates the relative temporal position of the frame. The vector $\vec{x}_t$ denotes all integer positions in the frame I_t. The optical flow $\vec{v}_{i \to j}$ contains the motion from frame I_i to frame I_j, the indices i, j are integer.

Problem Formulation Given are two adjacent frames I_0, I_1 and the target interpolation time step $t \in [0,1]$. To determine is the intermediate frame I_t with the use of optical flows created from frames I_0, I_1.

3.1 Naive Approach

Before starting with motion-compensated methods, the naive interpolation approach without considering motion is presented in this section to emphasis the advantages of using motion once again. The spatial information is not considered and only simple pixelwise interpolation takes place. Comparing to forward warping introduced in the previous chapter, this method acts as if the optical flows would contain only zero values (see Fig. 3.2). The intermediate frame is thus created as follows:

$$I_t(\vec{x}_t) = (1-t) \cdot I_0(\vec{x}_t) + t \cdot I_1(\vec{x}_t). \tag{3.1}$$

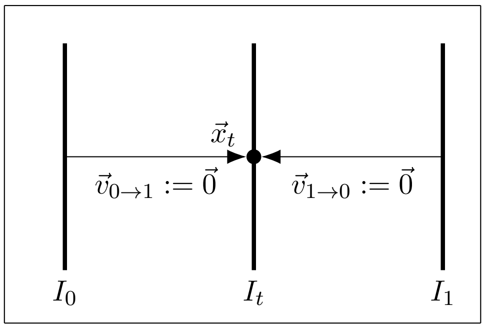


Fig. 3.2: Visualization of the naive approach when compared to forward warping. This method does not consider motion and acts as if the optical flows values were zero.

However, this simple method has obvious weaknesses. It completely ignores the image contents and the occurrence of motion between frames leads to severe artifacts, when objects from both sample frames appear in the interpolant twice (see Fig. 2.1). The following frame interpolation methods in this chapter are designed to reduce these duplication artifacts by using motion information provided by optical flow.

3.2 Assumption about Linear Motion

The central tool for frame interpolation in this chapter is the optical flow that contains pixelwise displacement information of the images. Since there are only two frames used for motion trajectory estimation, the objects in the intermediate frames follow a straight line (see Fig. 3.3). The main assumption of this chapter is that the linear function is sufficient for successful motion consideration.

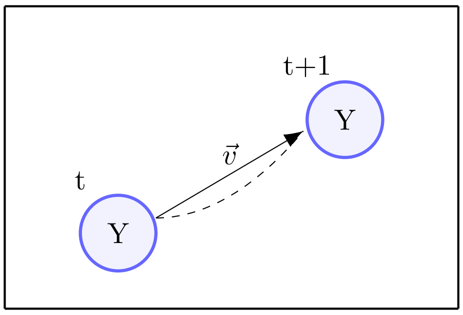


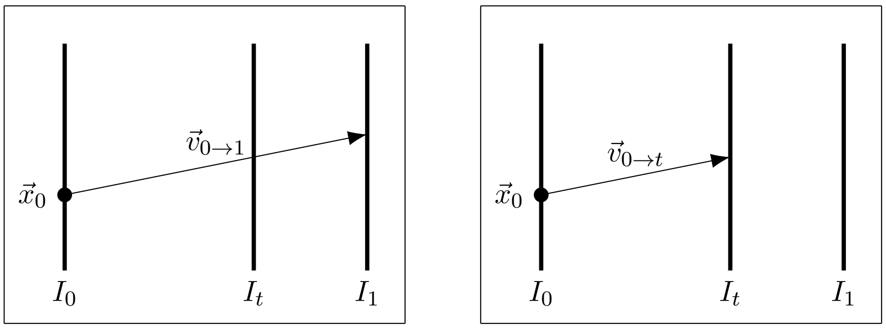
Fig. 3.3: In this chapter the motion between frames is assumed to be linear, according to the optical flow $\vec{v}$. The bigger is the temporal distance between frames t and $t+1$, the greater is the possibility for the actual motion (dashed) to differ from a linear trajectory.

While the assumption of linearity holds true for high frame rate slow motion sequences where the motion displacements are particularly small, this assumption may not suffice for larger motion. In Chapter 5 we are going to verify whether the linear function can approximate sequences with standard frame rate of 24 frames per second well enough.

3.3 Forward Warping

The considerations in the previous chapter concerned the warping process of image I_0 to reconstruct image I_1 using the optical flow $\vec{v}_{0\to 1}$ between them. The frame interpolation however focuses on the generation of intermediate frames I_t with $t \in [0, 1]$.

The core idea behind the forward warping lies in moving the information from the source image along the time axis determined by the optical flow. The intermediate frames can be created by moving the color information of I_0 partially until time step t with $\vec{v}_{0\to t}$ (see Fig. 3.4).



(A) Moving to full distance (B) Moving to partial distance

Fig. 3.4: Visualization of forward warping. `Left`: Information of I_0 is moved with $\vec{v}_{0\to 1}$ resulting in the warped reconstruction of frame I_1. `Right`: Information of I_0 is moved with $\vec{v}_{0\to t}$ resulting in an intermediate frame I_t.

The important question is how to estimate the flow $\vec{v}_{0\to t}$ when only the flow $\vec{v}_{0\to 1}$ is available. This is where the assumption about linear motion is applied. Since the optical flow $\vec{v}_{0\to 1}$ describes the whole displacement between images I_0 and I_1, this displacement has to be shortened in order to simulate the optical flow $\vec{v}_{0\to t}$. This is performed due to multiplication of the optical flow with t which leads to the displacement until the time step t and approximates $\vec{v}_{0\to t}$ as follows:

$$\vec{v}_{0\to t} \approx t \cdot \vec{v}_{0\to 1}. \tag{3.2}$$

This adjusted optical flow can be used for forward warping frame interpolation to obtain an intermediate frame I_t with given time step t as follows:

$$I_t(\vec{x}_0 + t \cdot \vec{v}_{0\to 1}(\vec{x}_0)) = I_0(\vec{x}_0). \tag{3.3}$$

The displacement of various objects in the scene can lead to two important problems. Firstly, there exists a possibility of overlapping when parts of different objects are moved to the same place in the image. This case requires collision handling, as described in Subsection 2.3.3. The other problem appears when an object is moved but nothing takes its place instead, resulting in the holes in the image. These undefined places have to be inpainted according to Subsection 2.4. Such holes often occur on the edges of images due to camera motion.

The forward warping frame interpolation consists of three main steps. Firstly, the optical flow $\vec{v}_{0\to1}$ is calculated based on images I_0 and I_1. Then the information of I_0 is forward warped using $\vec{v}_{0\to1}$ to create an intermediate interpolant I_t, splatting and collision handling are included in this step. Finally, the undefined places in the interpolant are closed during inpainting. These three steps correspond to the three steps of motion-compensated frame interpolation at the beginning of the chapter (see Fig. 3.1).

3.4 Forward Forward Warping

The forward warping introduced in the previous section uses color information from only one image and inevitably suffers from large amounts of undefined pixels after the warping step (see Fig. 3.5). Even the naive approach introduced in Section 3.1 uses color information from two images for frame interpolation, therefore this section focuses on advancing the method from the previous section by making use of both input frames I_0 and I_1.

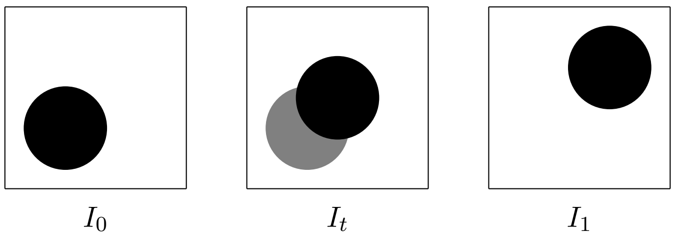


Fig. 3.5: `Left`: Source image I_0. `Right`: Target image I_1. `Middle`: Interpolant created with forward warping. The undefined pixels are colored in gray.

The use of two images is expected to enhance the performance by being able to give additional information needed for improving the closing of holes. While it is possible to use color information from both images with only a single optical flow, the separate optical flows independently calculated from frames I_0 and I_1 for both directions have been proven to have better performance [5]. Therefore, the input data for this section are the images I_0 and I_1 and the optical flows $\vec{v}_{0\to1}$ and $\vec{v}_{1\to0}$. The goal is as before to create an intermediate frame I_t for a given time step $t \in [0, 1]$.

The main idea of bidirectional frame interpolation methods, which use optical flows in both directions as opposed to the unidirectional methods like forward warping, is to create an interpolant for both directions and then fuse them together (see Fig. 3.6). The use of two interpolants leads to the change in notation. From now on in this chapter $I_t^{i\to j}$ denotes an interpolant created by using optical flow $\vec{v}_{i\to j}$ for the time step t. These intermediate interpolants are later used to create the final interpolant I_t.

The bidirectional frame interpolation method used in this work is called forward forward warping since forward warping is used there twice, once for every direction. Firstly, the

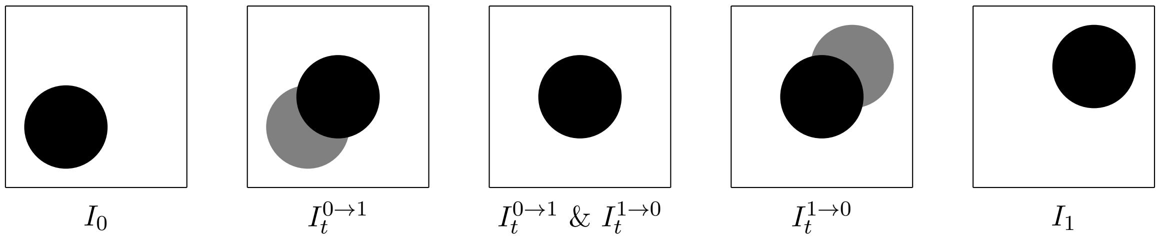


Fig. 3.6: Result of a bidirectional method. The leftmost and the rightmost images are I_0 and I_1 respectively. The second and second-to-last images are the interpolants created using forward warping for the respective optical flow direction. The undefined areas are colored in gray and are different for both interpolants. The image in the middle is created through the appropriate fusing of the interpolants.

optical flows $\vec{v}_{0\to 1}$ and $\vec{v}_{1\to 0}$ are calculated. Then the intermediate interpolants $I_t^{0\to 1}$ and $I_t^{1\to 0}$ are calculated for every direction with forward warping using the corresponding optical flow to simulate the flows $\vec{v}_{0\to t}$ and $\vec{v}_{1\to t}$ as follows:

$$I_t^{0\to 1}(\vec{x}_0 + t \cdot \vec{v}_{0\to 1}(\vec{x}_0)) = I_0(\vec{x}_0)$$
$$I_t^{1\to 0}(\vec{x}_1 + (1-t) \cdot \vec{v}_{1\to 0}(\vec{x}_1)) = I_1(\vec{x}_1)$$

The intermediate interpolants are illustrated in Fig. 3.7. In this example, the interpolant on the left is more reliable since its distance to the sample frame I_0 is smaller than the distance of $I_t^{1\to 0}$ to I_1.

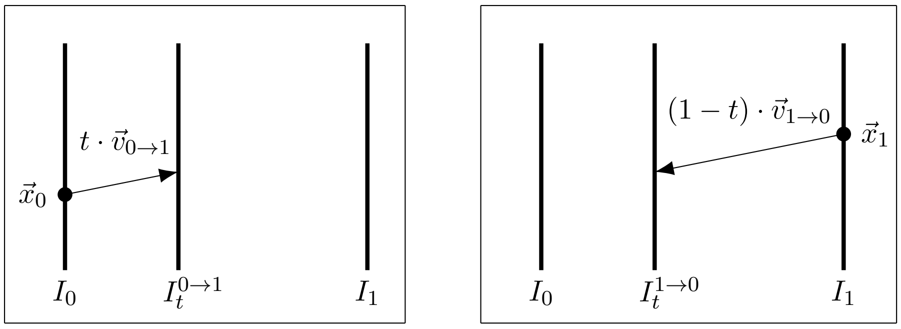


Fig. 3.7: Visualization of forward forward warping interpolants. `Left`: Interpolant created for the direction from I_0 to I_1. `Right`: Interpolant for the opposite direction.

In the next step the interpolants are fused together. The fusion is performed in such a way that as many undefined pixels as possible get a value. If both interpolants are undefined at some position, then this position remains undefined in the final interpolant and has to be inpainted. If only one of the interpolants at a position is defined, then this value is used in the final interpolant. If both interpolants are defined, then their values have to be weighted according to the distance to the source frame and summed up. The interpolant that lies temporarily nearer to its source frame is considered more reliable and is weighted stronger as follows:

$$I_t = (1-t) \cdot I_t^{0\to 1} + t \cdot I_t^{0\to 1}. \tag{3.4}$$

The interpolants fusing is visualized in Fig. 3.8. Finally, the inpainting step is performed after interpolants fusing.

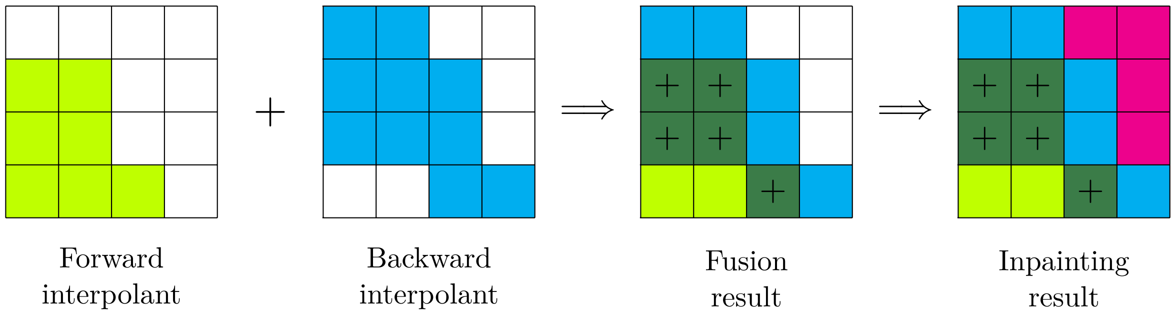


Fig. 3.8: Visualization of interpolants fusing followed with the inpainting step. Lime and blue colors denote the pixels acquired from the forward and backward interpolants respectively. Undefined pixels are marked in white color. The fused pixels which are defined in both interpolants are marked in dark green with + sign. The remaining undefined pixels are then inpainted and marked in pink.

3.5 Artifacts Reduction

This section describes some specific types of artifacts created during motion-compensated frame interpolation and ways to reduce them. It also addresses some artifacts that cannot be removed with the approaches introduced in this work, explaining its restrictions and emphasizing how challenging frame interpolation is.

3.5.1 Fusing Blur Reduction

A drawback of interpolants fusing as introduced in Equation (3.4) is blurring of the same objects that are warped to different positions in the interpolants (see Fig. 3.9). The blurring reaches its maximum at the time step 0.5 when both interpolants are equally weighted and the motion prediction is the most unreliable. The bigger the difference in the weights, the smaller the blurring gets.

To counteract the blurring, another approach for using interpolants can be applied. The core idea is to keep the difference between the weights as big as possible, i.e., to keep it one and use color information from only one interpolant for the pixels in the result. In this case one of the interpolants is chosen as the main interpolant and the other is used only in places where the main interpolant is undefined (see Fig. 3.10).

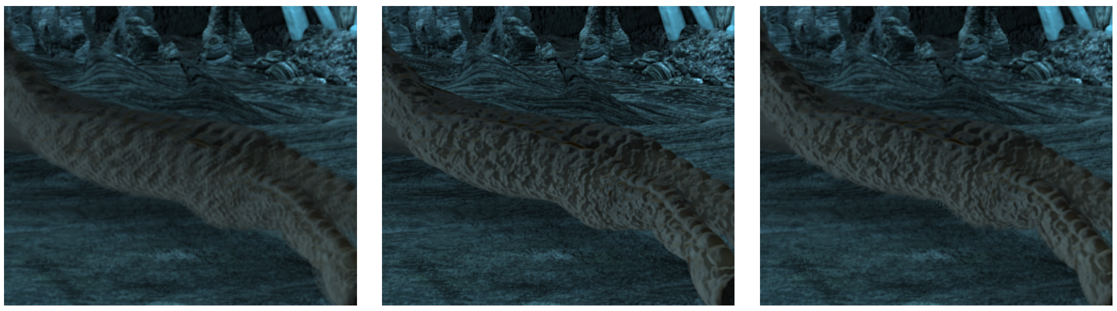


Fig. 3.9: `Left`: Cropped result of forward forward warping for the time step 0.5. Note the blur produced by interpolants fusing. `Middle`: Refence image for the same time step. `Right`: Result of the modification introduced in this subsection.

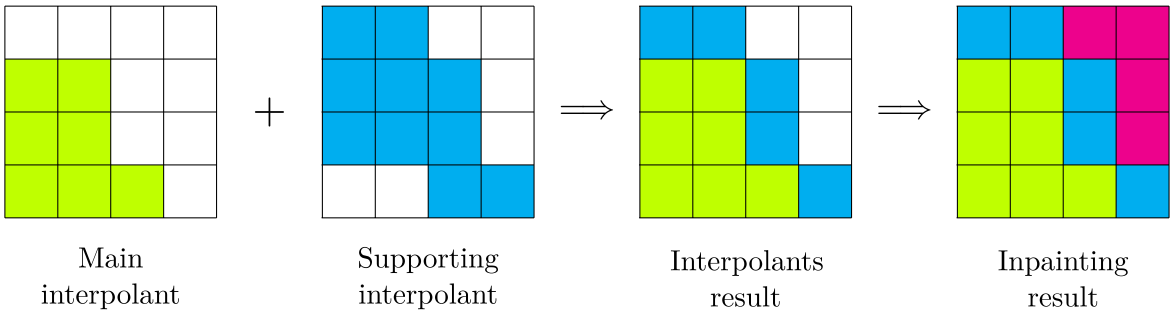


Fig. 3.10: Visualization of interpolation using a main interpolant followed with the inpainting step. Lime and blue colors denote the pixels acquired from the forward and backward interpolants respectively. Undefined pixels are marked in white color. The remaining undefined pixels are then inpainted and marked in pink.

3.5.2 Single Pixel Artifact Reduction

Another noticeable artifact is created during the warping process due to inaccuracies when some pixels are marked as defined in occluded regions that should be fully undefined, e.g. because they were covered by a moving object. This leads to standing out artifacts in form of single pixels after the fusing of interpolants (see Fig. 3.11).

To counteract these artifacts, the following modification is proposed: the image is traversed four times and in every iteration the number of undefined pixels among the eight neighbors is calculated for every pixel. If this number exceeds the threshold, this pixel is marked as undefined for the next iterations and the final result. The threshold corresponds to the number of the current iteration, starting with four and increasing up to seven. The result of this modification is illustrated in Fig. 3.12 and contains much less disturbing artifacts when compared to Fig. 3.11.

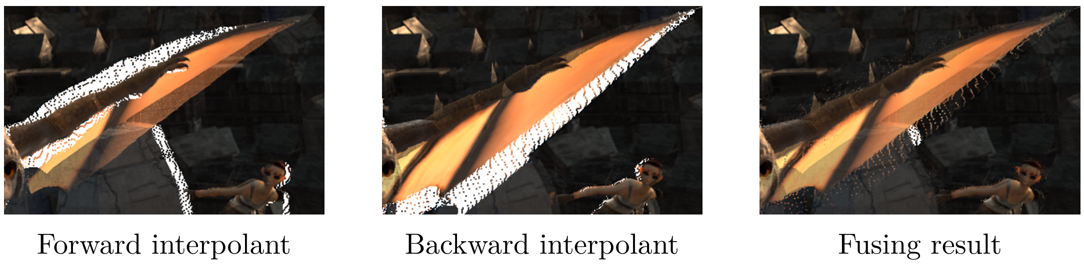


Fig. 3.11: Visualization of warping artifacts created when some pixels in undefined regions are falsely labeled as defined. Undefined pixels are marked in white.

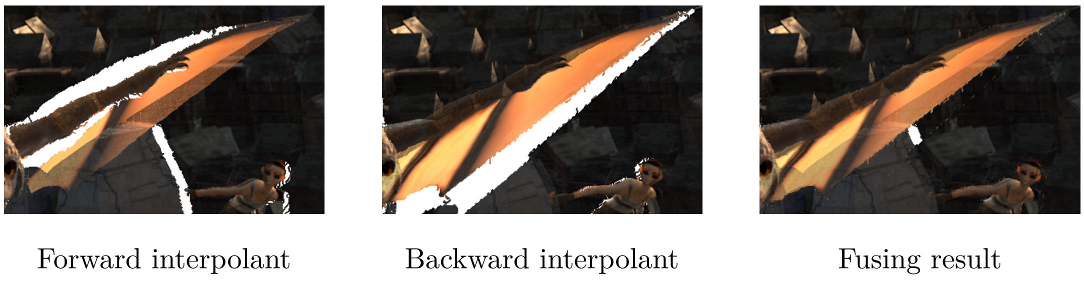


Fig. 3.12: Visualization of the interpolants after modification and their fusing result. Undefined pixels are marked in white.

3.5.3 Advanced Artifacts Outlook

Not all artifact types have effective countermeasures which are applicable based solely on the knowledge about motion. Two of them are presented in this subsection to emphasis the difficulty of frame interpolation.

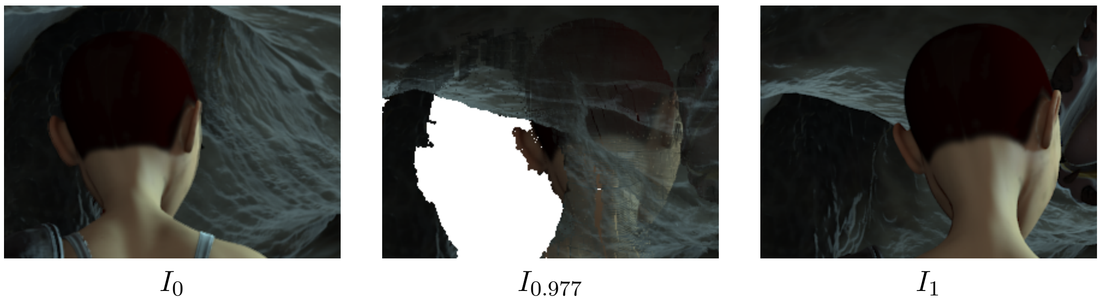


Fig. 3.13: Visualization of transparency artifacts. On the left and right are the sample frames I_0 and I_1. In the middle is the result of forward warping for the time step $t = 0.977$ with single pixel artifact reduction modification before inpainting.

Fig. 3.13 illustrates the transparency artifacts. They are caused by the way of collision handling described in Algorithm 1. As the dragon and girl move, parts of them occupy the same areas. Since there is no knowledge about what object is nearer to the observer, the values in these areas are simply summed up and normalized, resulting in transparent objects. The knowledge about distance to the objects is required in order to effectively reduce such artifacts.

Other sophisticated artifacts are illustrated in Fig. 3.14. As the wing changes its form between sample frames, the optical flow estimation and frame interpolation become extremely difficult leading to poor results. The case of object deformation is one of the most challenging issues in computer vision along with the luminance change.

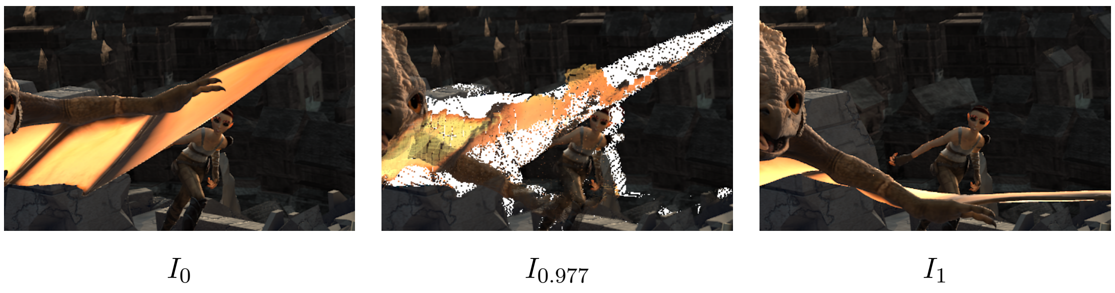


Fig. 3.14: Visualization of artifacts caused by object deformation. On the left and right are the sample frames I_0 and I_1. In the middle is the result of forward warping for the time step $t = 0.977$ before inpainting.

3.6 Summary

In this chapter, the frame interpolation methods using two frames were introduced with the general assumption that the motion between adjacent frames can be represented as a linear trajectory. Firstly, we took a look at the unidirectional forward warping that warps color information from one of two input frames, then this method was extended to bidirectional forward forward warping that uses both frames. Finally, several types of frame interpolation artifacts were described alongside with method modifications aiming to reduce some of them.

4 Frame Interpolation using Multiple Frames

Chapter 3 addressed the frame interpolation with a central assumption that the motion between frames can be approximated by a linear function. This chapter goes away from this assumption and introduces frame interpolation methods that approximate the motion as a polynomial function with a degree higher than one. Since optical flow plays as important role as in the previous chapter and a polynomial of degree n needs $n+1$ distinct data points to be determined, these polynomial methods are part of motion-compensated frame interpolation methods that use multiple frames.

Similar to the three main steps of motion-compensated frame interpolation from the previous chapter, the four main steps of the methods in this chapter are introduced in Fig. 4.1. It contains all three steps from Fig. 3.1 and additionally has a motion function estimation as the second step. The motion function that is used to describe motion between frames was not mentioned explicitly in the previous chapter since the object position in the frame interpolant was determined linearly. The methods in this chapter use more sophisticated polynomial motion functions that need to be explicitly calculated, therefore this step is added to the process to emphasis its importance.

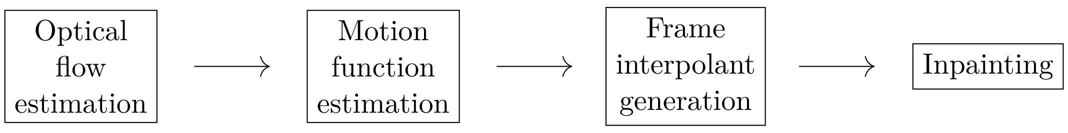


Fig. 4.1: Four main steps of motion-compensated frame interpolation with non-linear motion function.

The considerations about the polynomial motion functions are first explained on the case of the quadratic function that needs three frames for calculation, then they are generalized for a polynomial function of an arbitrary degree n. In contrast to the methods introduced in this chapter, the methods from Chapter 3 will be called linear forward warping and linear forward forward warping due to their linear motion functions. This chapter introduces both uni- and bidirectional polynomial frame interpolation methods.

Notation Since the restriction on having only two frames holds no more, some changes concerning notation have to be made. The input frames I_k in this chapter have integer indices $k \in \mathbb{Z}$ that order them on the time axis. The interpolation target frames I_t with $t \in (0,1)$ are also ordered temporarily and lay between the input frames I_0 and I_1. The vectors $\vec{x}_k$ and $\vec{x}_t$ denote all integer positions in the frames I_k and I_t respectively. As before, the optical flow $\vec{v}_{i\to j}$ contains the motion from frame I_i to frame I_j, the indices i, j are integer.

Problem Formulation Given are several adjacent frames $I_{-m}, I_{-m+1}, ..., I_0, I_1, ...I_{n-1}, I_n$ and the interpolation time step $t \in [0, 1]$. To determine is the intermediate frame I_t with the use of optical flows created from the frames I_k.

4.1 Quadratic Forward Warping

The forward warping method from the previous chapter used a linear motion function determined by two frames I_0 and I_1. The quadratic forward warping in this section on the other hand uses an additional frame I_{-1} and two optical flows $\vec{v}_{0\to1}$, $\vec{v}_{0\to-1}$ to better approximate the pixelwise motion functions not with a straight line but with a parabola (see Fig 4.2). The frame I_{-1} is the frame before I_0.

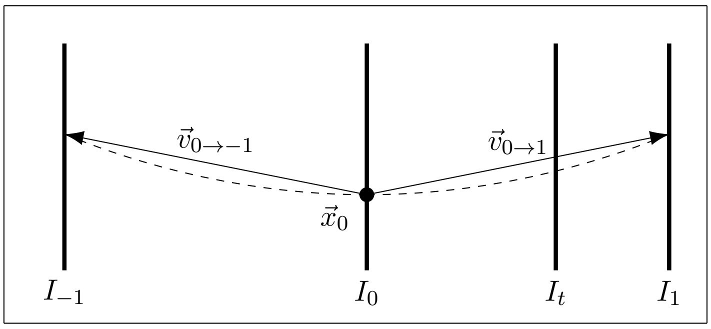


Fig. 4.2: Visualization of the quadratic forward warping. Three frames I_{-1}, I_0, I_1 are used for the motion function estimation at position $\vec{x}_0$ which results in a parabola marked with the dashed line. Note the differences between places where this parabola and $\vec{v}_{0\to1}$ used in the linear forward warping intersect the position of the frame I_t.

The only differences between the algorithms of quadratic forward warping and linear forward warping lay in the number of given frames, the number of optical flows to be estimated and in how the warping position is calculated. It is no longer performed linearly as $\vec{x}_0 + t \cdot \vec{v}_{0\to1}(\vec{x}_0)$, but with the use of a quadratic motion function $\vec{v}_{\vec{x}_0}(t)$ calculated for every pixel $\vec{x}_0$:

$$I_t(\vec{v}_{\vec{x}_0}(t)) = I_0(\vec{x}_0) \tag{4.1}$$

4.1.1 Quadratic Motion Function Calculation

This subsection concentrates on the problem of the quadratic motion estimation. As every pixel $\vec{x}_0$ in the image I_0 has its own optical flow values for both directions, it also has its own quadratic motion function $\vec{v}_{\vec{x}_0}(t)$ to be calculated. The quadratic motion function for $\vec{x}_0$ is determined using three positions: the position $\vec{x}_0$ and the corresponding positions from the previous and next frames calculated with the use of optical flows as $\vec{x}_0 + \vec{v}_{0\to-1}(\vec{x}_0)$ and $\vec{x}_0 + \vec{v}_{0\to1}(\vec{x}_0)$. The first value indicates from where the object in the pixel $\vec{x}_0$ came, the second value points to where it will move afterwards.

To simplify the calculation process, we assume that the motion functions can be determined for x- and y-axis independently, resulting in the following equations:

$$v_x(t) = a_x t^2 + b_x t + c_x$$
$$v_y(t) = a_y t^2 + b_y t + c_y.$$

These equations can be solved separately and their parameters a, b, c are derived from the next system of linear equations for every pixel $\vec{x}_0$ on the example for the x-axis:

$$\begin{cases} v_{x_0}(0) = x_0 \\ v_{x_0}(1) = x_0 + \Delta\ x_{0\to 1} \\ v_{x_0}(-1) = x_0 + \Delta x_{0\to -1} \end{cases}$$

with

$$\vec{x}_0 = (x_0, y_0),$$
$$\vec{v}_{0\to 1}(\vec{x}_0) = (\Delta x_{0\to 1}, \Delta y_{0\to 1})$$
$$\vec{v}_{0\to -1}(\vec{x}_0) = (\Delta x_{0\to -1}, \Delta y_{0\to -1}).$$

This leads to the following simple system of linear equations:

$$\begin{cases} c_{x_0} = x_0 \\ a_{x_0} + b_{x_0} + c_{x_0} = x_0 + \quad \Delta x_{0\to 1} \\ a_{x_0} - b_{x_0} + c_{x_0} = x_0 + \Delta x_{0\to -1} \end{cases}$$

Solving this system leads to the x-axis motion function to be determined as follows:

$$v_{x_0}(t) = \frac{x_{0\to 1} + x_{0\to -1}}{2} \cdot t^2 + \frac{x_{0\to 1} - x_{0\to -1}}{2} \cdot t + x_0. \tag{4.2}$$

The equation for y-axis is analogous. The equations for both axis combined lead to motion function for the pixel $\vec{x}_0 = (x_0, y_0)$ to be calculated as:

$$\vec{v}_{\vec{x}_0}(t) = (v_{x_0}(t), v_{y_0}(t)). \tag{4.3}$$

This motion function quadratically approximates the motion between frames I_{-1} and I_1 and is able to predict a target position for frame interpolation for $t \in [-1, 1]$.

4.2 Generalized Polynomial Forward Warping

Since a quadratic motion function potentially approximates the real motion better than a linear trajectory, the motion functions based on a polynomial of a higher degree also need to be taken into consideration to achieve an even better motion approximation (see Fig. 4.3). This section thus concerns the transition from quadratic forward warping to a generalized polynomial forward warping with a certain degree n. Exactly as in the previous section, the number of input frames, the number of optical flows and the way of calculating the warping position differ from those for the linear forward warping from the previous chapter. Every pixel $\vec{x}_0$ thus has its own polynomial motion function $\vec{v}_{n,\vec{x}_0}(t)$ and the interpolant is calculated as follows:

$$I_t(\vec{v}_{n,\vec{x}_0}(t)) = I_0(\vec{x}_0) \tag{4.4}$$

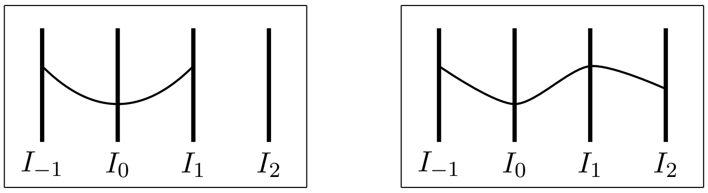


Fig. 4.3: `Left`: Quadratic motion function. `Right`: Cubic motion function.

4.2.1 Input Frames

The first question for the polynomial warping to be answered is which frames are used for interpolation. The polynomial of degree n needs $n+1$ data points, the parameter n in this chapter is set as $n \geq 2$ for the polynomials to be starting with quadratic functions. Since the result interpolant lays between I_0 and I_1, both these frames should be included in the input.

The motion function depends on which frames are used, and the input frames for polynomial motion in this section are distributed equally on both sides of I_0. If the number of frames is even, then the right side has one more frame. This distribution not only contributes to a more accurate motion function, but also considers the characteristics of optical flow as shown in the next subsection.

4.2.2 Polynomial Motion Function Calculation

The motion functions are calculated based on the optical flows. The optical flows are chosen according to the motion function type: in the previous chapter there was one optical flow $\vec{v}_{0\to1}$ based on both available frames I_0 and I_1 and in the previous section there were two optical flows $\vec{v}_{0\to1}$ and $\vec{v}_{0\to-1}$ based on pairs I_0, I_1 and I_0, I_{-1} respectively. The polynomial motion function with degree n needs $n+1$ frames, which leads to various opportunities for the optical flow set.

In this chapter the optical flows are calculated only between adjacent frames since motion estimation over bigger distances is less reliable. A position of an object from the start frame I_0 on other frames is determined with backward warping with optical flow and nearest neighbor strategy. The position estimation for more distant frames requires use of several backward warpings with sequential optical flows (see Fig. 4.4) and is calculated as follows:

$$\vec{x}_k = \vec{x}_0 + \sum_{i=0}^{k-1} \vec{v}_{i\to i+1}(\lfloor \vec{x}_i \rceil), \tag{4.5}$$

where $\lfloor \cdot \rceil$ denotes rounding to the next integer value. For negative k the index i in this equation is decreasing.

Since the optical flow values are determined only for integer coordinates, the intermediate real-valued positions have to be rounded to the nearest integer value according to the nearest neighbor strategy. Another strategy would be to involve all four neighbors of the intermediate warped position to calculate its optical flow value and is not further discussed in this work.

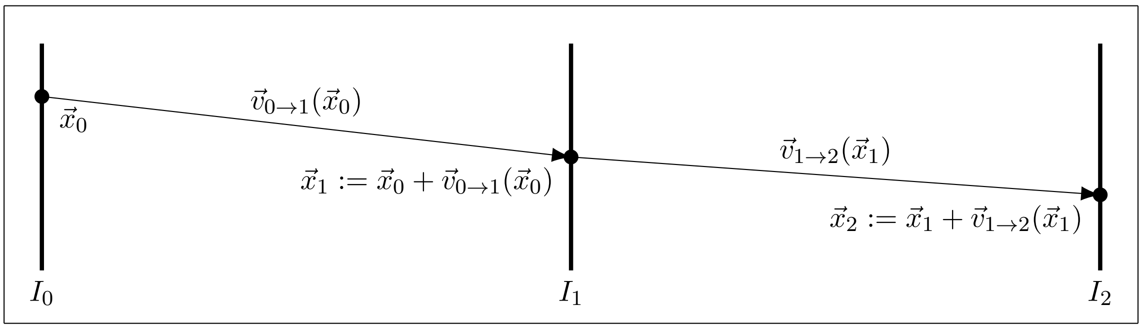

$\vec{x}_0$
$\vec{v}_{0\to 1}(\vec{x}_0)$
$\vec{x}_1 := \vec{x}_0 + \vec{v}_{0\to 1}(\vec{x}_0)$
$\vec{v}_{1\to 2}(\vec{x}_1)$
$\vec{x}_2 := \vec{x}_1 + \vec{v}_{1\to 2}(\vec{x}_1)$
I_0
I_1
I_2

Fig. 4.4: Visualization of sequential position calculation over several optical flow values.

The problem of sequential use of optical flows is that some intermediate positions can land beyond the valid value domain, making the calculation of subsequent positions impossible (see Fig. 4.5). This often happens for pixels on image edges where the objects go beyond the image. In this case there are not enough positions for polynomial motion function with initial degree n. Instead of completely giving up on the pixel $\vec{x}_0$, we use it with a motion function of a lower degree to minimize the number of pixels that need to be inpainted.

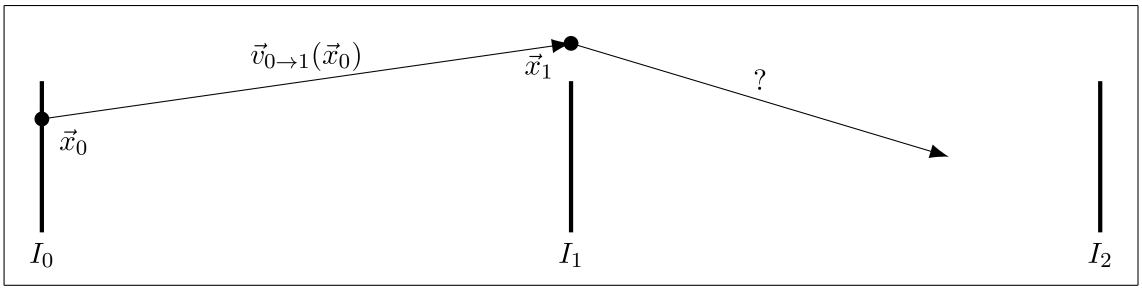


Fig. 4.5: Visualization of a position landing outside the valid value domain. Since $\vec{x}_1$ lays outside I_1, no further calculation with $\vec{v}_{1\to 2}(\vec{x}_1)$ is possible.

The system of linear equations for quadratic motion function from the previous chapter could be solved manually and was the only necessary system. Because of the possibility that positions during generalized polynomial motion function land outside the valid value set, more systems of linear equations need to be considered to cover all possible cases. For example, when using five input frames four cases need to be dealt with: the usual case, the case where position $\vec{x}_{-1}$ is invalid, the case where position $\vec{x}_1$ is invalid and the case where both these positions are invalid (the last case is illustrated in Fig. 4.6). Four systems of linear equations are needed to cover these cases: one for five available positions, two for four and one for three.

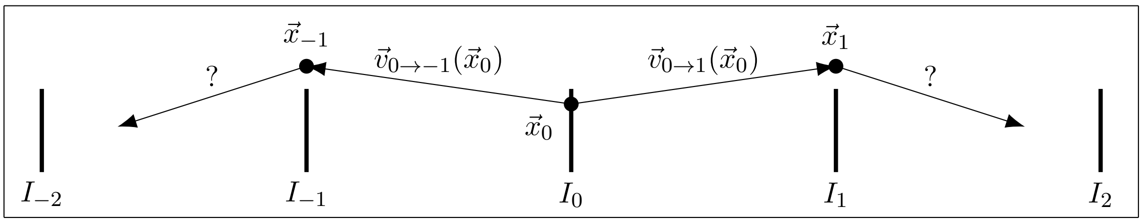


Fig. 4.6: Visualization of a polynomial motion function with five frames when positions $\vec{x}_{-1}$ and $\vec{x}_1$ land outside the valid value domain. Only a quadratic motion function can be calculated for the pixel $\vec{x}_0$ in this case.

While it is possible to automatically create and solve all necessary systems of linear equations for a given polynomial degree n, a better approach would be to use well-known and efficient methods designed specifically for this problem, such as polynomial and spline interpolation. Since polynomial interpolation that tries to approximate $n + 1$ data points with a polynomial of the degree at most n may diverge for equally spaced intervals [6] and is computationally expensive, we are going to use spline interpolation that will be introduced in the following subsection.

4.2.3 Spline Interpolation

When using the polynomial interpolation with polynomials of higher degrees, increasing the number of interpolation points may cause strong oscillation between the data points. Spline interpolation is commonly used to resolve this problem.

Spline interpolation is a special form of interpolation using piecewise polynomial functions called *splines*. Instead of building a single polynomial function of higher degree which is influenced by every data point, spline interpolation fits several splines of lower degree to small subsets of interpolation points. For example, in the case of eight input frames, seven splines are fitted between the pairs of frames.

Given the points $(x_i, y_i), i \in \{0, ..., n\}$, the goal is to find each polynomial $q_i(x)$ which interpolates from (x_i, y_i) to (x_{i+1}, y_{i+1}). The final function is built from $q_i(x)$ piecewise in the corresponding intervals. The spline interpolation in this work uses cubic polynomials as splines and is implemented using the *CubicSpline* method from the sub-package *interpolate* of the *SciPy* scientific computing library [45].

4.3 Generalized Polynomial Forward Forward Warping

The last step in this chapter is rather similar to the last step of the previous chapter. In this section the unidirectional forward warping will be extended to bidirectional forward forward warping using the previous considerations concerning the polynomial motion function calculation.

The forward forward warping is based on fusing two intermediate interpolants created with forward warping of I_0 or I_1 as source image respectively to get the final interpolant with less undefined pixels. Each interpolant has its own input frames and optical flows sets required for interpolation (see Fig. 4.7).

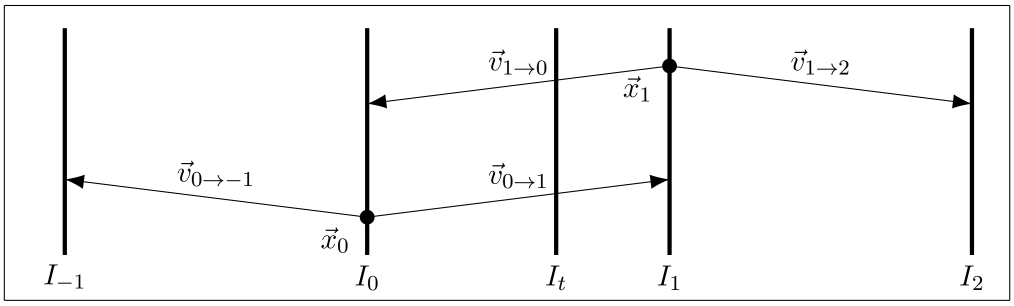


Fig. 4.7: Visualization of the quadratic forward forward warping. The interpolant that gets color information from the frame I_0 needs frames I_{-1}, I_0 and I_1 and optical flows $\vec{v}_{0\to -1}$ and $\vec{v}_{0\to 1}$. The interpolant that gets color information from the frame I_1 on the other hand needs frames I_0, I_1 and I_2 and optical flows $\vec{v}_{1\to 0}$ and $\vec{v}_{1\to 2}$. Four frames and four optical flows are required in total.

While the even number of frames in Section 4.1 always leads to the distribution where the right side of I_0 has one more frame, the distribution in this chapter depends on the interpolant direction. The forward interpolant created from I_0 has the same distribution as in Section 4.1. The backward interpolant created from I_1 has its input frames equally distributed on both sides of I_1, not I_0. In the case of an even number of frames, the left side of I_1 has one more frame. This minimizes the number of required frames.

4.4 Summary

In this chapter, we went away from the assumption about linear motion and took a look at the methods that use more than two frames to approximate the motion between frames. We started with quadratic warping that uses three frames and models the motion as a parabola and generalized the approach for an arbitrary number of frames for both forward and forward forward warping.

5 Results and Evaluation

This chapter focuses on the evaluation of frame interpolation methods introduced in the previous chapters. We are particularly interested in the advantages that the polynomial frame interpolation is expected to bring over the linear variant, and what its limitations are. Another aspect of interest is frame interpolation with several intermediate frames and their quality in relation to their temporal position.

Firstly, the datasets and quality assessment measures are introduced, then the experiments concerning interpolation methods using two frames introduced in Chapter 3 and multiple frames introduced in Chapter 4 are conducted and the usefulness of modifications proposed in Section 3.5 is tested. The results and observations are discussed in detail and finally the overall conclusion is given.

5.1 Test Data

The test results strongly depend on the quality and variety of the used test data. It is meaningful to use nontrivial test data that is as close as possible to the reality and contains challenges such as larger motion, occlusions and diversity.

In the computer vision field, several standardized datasets for the optical flow estimation assessment were developed which can also be used to compare frame interpolation results. The earlier datasets such as the famous Yosemite sequence [2] or Middlebury dataset [3] are rather small due to the complexity of ground truth optical flow estimation. In an attempt to create larger and more realistic datasets for such real-world applications as autonomous driving, two *KITTI* datasets from 2012 [15] and 2015 [33] were created by gathering ground truth data while driving with a Light Detection and Ranging (LIDAR) system that is used to estimate distances.

Another example of a popular standardized dataset is the *MPI Sintel* dataset from 2012 [9]. This dataset is based on the open-source computer-animated 3D short film Sintel. Its artificial nature allows to derive optical flows from the film source data with key features: long sequences, large motions, motion and defocus blur, specular reflections, and atmospheric effects. Alongside Kitti, Sintel serves as one of the most used benchmarks for optical flow estimation.

However, the MPI Sintel dataset is sampled at a rather low frame rate of 24 frames per second and is not suitable for evaluation of multiple intermediate frames. Therefore, the experiments in this work are conducted on the high frame rate version of the MPI dataset that is re-rendered at a multiple frame rate of the default MPI Sintel dataset [20]. As a result, the high frame rate version contains additional 41 intermediate frames between every frame pair. This dataset will be used for the frame interpolation estimation in this work.

5.2 Quality Assessment Measures

While checking whether two images are completely identical is a quite simple task, the problem of finding out their grade of similarity has proven to be challenging. Furthermore, the widely used techniques such as root mean squared error (RMSE), peak signal-to-noise ratio (PSNR) or structure similarity index measure (SSIM) are stated to be insufficient for evaluation of image interpolation performance [30,32]. Despite this fact, RMSE and SSIM are mainly used in this work for result quality assessment, since better methods are still to be developed.

The RMSE estimates the difference between the interpolant $I(\vec{x})$ and the reference image $I_{GT}(\vec{x})$ by calculating pixelwise differences with N being the number of pixels:

$$RMSE(I, I_{GT}) = \sqrt{\frac{1}{N}\sum_{\vec{x}} ||I(\vec{x}) - I_{GT}(\vec{x})||^2}. \tag{5.1}$$

The RMSE provides an easy way to measure differences between samples. However, such simple error metric does not necessarily correspond to the human visual perception. It is possible to create examples with clear visual differences, but the same RMSE value, as visualized in Fig 5.1.

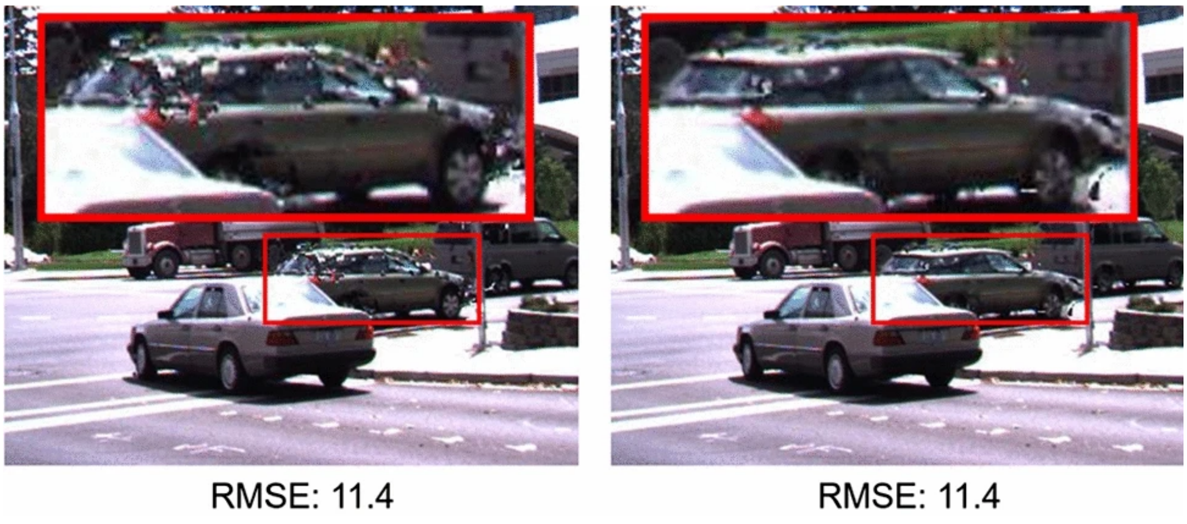


Fig. 5.1: Two frames interpolated with different methods. The RMSE values are equal, but the visual quality differs, in particular in the zoomed regions. Image source: [30]

The RMSE is not the only error metric that suffers from such problems. All metrics have their own weak points, that is why it is not wise to use a single technique for quality measurement, as confirmed in [30]. In this work, the SSIM is used in addition to the RMSE.

Unlike other common techniques, the SSIM does not try to estimate the absolute errors but to quantify differences in structural information. Since spatially close pixels have strong interdependencies, the SSIM evaluates squared areas of images called windows rather than single pixels, as it is the case for the RMSE. It has been recognized that the SSIM is sensitive to relative scaling, translations, and rotations [8].

The SSIM value of windows $\mathtt{x}$ in I and $\mathtt{y}$ in I_{GT} is a weighted combination of luminance l, contrast c and structure s calculated as:

$$SSIM(x,y) = [(l(x,y))^{\alpha} \cdot (c(x,y))^{\beta} \cdot (s(x,y))^{\gamma}]. \tag{5.2}$$

In this work luminance, contrast and structure are considered equally important, hence the weights are chosen as $\alpha = \beta = \gamma = 1$. The individual functions of comparison measurements are:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{{\mu_x}^2 + {\mu_y}^2 + c_1},\ c(x,y) = \frac{2\sigma_x\sigma_y + c_1}{{\sigma_x}^2 + {\sigma_y}^2 + c_1},\ s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{5.3}$$

with μ being the averages, σ being the variances, and c_i being the constants to stabilize the division with weak denominator according to the dynamic range of pixel values.

Since the test data are color images, the SSIM indices are calculated for every color channel separately and then averaged. The final SSIM value for I and I_{GT} with N windows is obtained as follows:

$$SSIM(I, I_{GT}) = \frac{1}{N} \sum_{x \in I, y \in I_{GT}} SSIM(x,y). \tag{5.4}$$

The SSIM value of identical images is 1 and the quality of an interpolant is determined by its similarity to the reference image I_{GT}, thus we want the SSIM values to be as near to 1 as possible. On the contrary, low RMSE values are desired to keep the differences small.

5.3 Experiments

The high frame rate version of MPI Sintel dataset consists of several independent scenes and we are going to use ten of them named *ambush_2*, *ambush_6*, *bamboo_2*, *cave_2*, *cave_4*, *market_2*, *market_6*, *mountain_1*, *temple_2*, and *temple_3*. Since the test data allows comparison for 41 intermediate frames, the frame interpolation methods introduced in previous chapters will also create 41 intermediate frames for every scene by setting the input time step t according to the frame position. The results are averaged over all scenes and all time steps if not stated otherwise.

In the first subsection the results of linear and quadratic frame interpolation are compared, interpolants of forward and forward forward warping are visually evaluated before and after inpainting and the interpolant quality is inspected over time for all intermediate frames. The second subsection focuses on methods using multiple frames and points out the weaknesses of sequentially warped optical flows introduced in Subsection 4.2.2. The third subsection compares methods using sequentially warped optical flow with methods using direct optical flows both in average and for every single scene separately. The last subsection briefly touches on the topic of consistency checking for direct optical flows.

5.3.1 Linear vs. Quadratic

This section focuses on comparing the results of the linear frame interpolation introduced in Chapter 3 with the quadratic frame interpolation from Chapter 4. Table 5.1 shows the results for linear and quadratic methods both before and after inpainting.

Method	Before inpainting	After inpainting	
	RMSE	RMSE	SSIM
Linear FW	22.912	28.039	0.639
Quadratic FW	**20.842**	**26.277**	**0.732**
Linear FFW	18.678	18.824	0.677
Quadratic FFW	**16.138**	**16.272**	**0.784**

Table 5.1: Results of the interpolation with linear and quadratic forward and forward forward warping methods.

The quadratic methods perform consistently better than the linear variants for both unidirectional forward warping and bidirectional forward forward warping, allowing to conclude that the use of an additional frame effectively helps to yield better results. Even the linear forward forward warping completely outperforms both forward warping variants, making clear that the use of color information from both sample frames is beneficial. Quadratic forward forward warping thus shows the best performance among these four methods.

As mentioned in the previous section, the SSIM index is calculated for the whole image by evaluating areas and not single pixels, thus it can not be used for quality assessment before inpainting, since the undefined pixels can not be excluded for estimation and would greatly influence the result. The RMSE values on the other hand can be calculated exclusively for defined pixels and show a drastic difference between uni- and bidirectional methods regarding results before and after inpainting. While forward forward warping has comparably few undefined pixels and its RMSE values are almost identical before and after inpainting, forward warping naturally has a large number of undefined pixels that need to be inpainted. The inpainting of bigger holes yields poor results and leads to a significantly larger RMSE value. Fig. 5.2 illustrates the result difference between uni- and bidirectional methods.

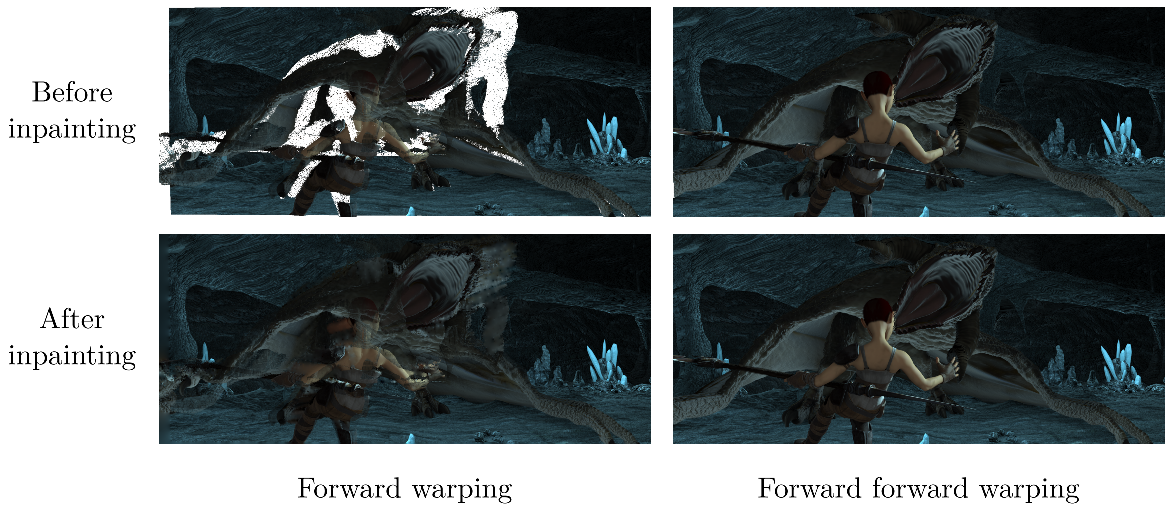


Fig. 5.2: Results of quadratic forward and forward forward warping for the scene *cave_4* and the time step $t = 0.977$ before and after inpainting. Undefined pixels are marked in white.

The previous table shows only the values averaged over all intermediate interpolants, the temporal behavior of results' quality is however also of interest. Fig. 5.3 visualizes how the RMSE and SSIM values change over time for forward and forward forward warping after inpainting for a single scene. While the values for forward warping monotonically get worse with a bigger time step t for both assessment measures, the trend of forward forward warping results resembles a parabola.

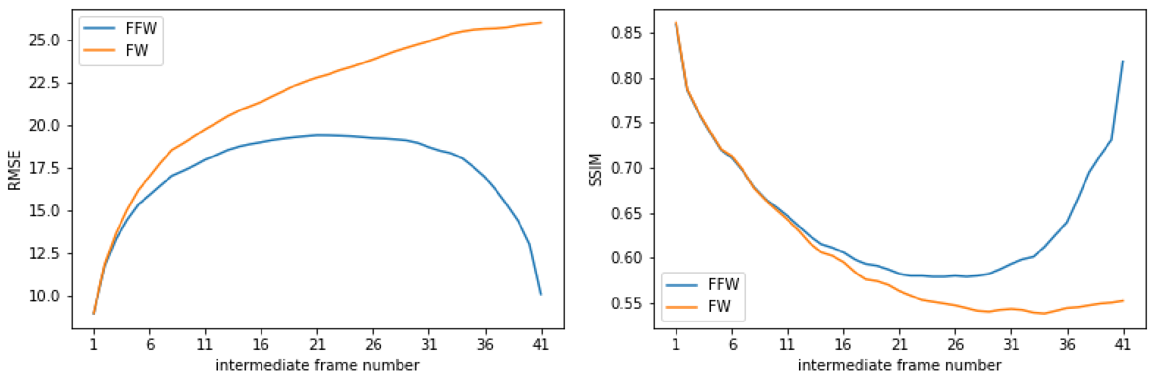


Fig. 5.3: Visualization of quality assessment measures' values changing over time, calculated for 41 intermediate frames after inpainting. The frame interpolation is performed with quadratic forward and forward forward warping for the scene *cave_4*.

The monotonical behavior of forward warping results can be explained by the fact that the number of undefined pixels gets larger with increasing distance to the sample frame. As these pixels need to be inpainted, the quality of interpolants steadily declines. The forward forward warping has two sample frames instead of one. Its results can be said to behave analogically to the forward warping, as the poorest quality is achieved approximately at the intermediate frame 21 where the distance to the sample frames is also the largest.

5.3.2 Frame Interpolation using Multiple Frames

As the use of one additional frame has proven useful, the next step is to evaluate the interpolation methods using more frames. Table 5.2 summarizes the results of forward and forward forward warping after inpainting performed for polynomials of different degrees starting with the quadratic methods. The results show that except for the case of degree four the performance slowly decreases with the increasing degree.

Polynomial degree	FW		FFW	
	RMSE	SSIM	RMSE	SSIM
2	**26.277**	**0.732**	**16.272**	**0.784**
3	26.615	0.730	17.037	0.771
4	26.450	**0.732**	16.853	0.764
5	26.651	0.717	17.275	0.753
6	26.709	0.715	17.378	0.747
7	26.793	0.707	17.540	0.741

Table 5.2: Results of the interpolation with forward and forward forward warping after inpainting using different polynomial degrees.

The reason for this behavior possibly lays in the way how the optical flow over longer distance is determined. Since direct motion estimation would be less reliable, we decided to sequentially use optical flows between adjacent frames to estimate it, as described in Subsection 4.2.2.

Fig. 5.4 visualizes an optical flow as described above in comparison to a direct optical flow, an optical flow between adjacent frames and an input frame. As expected, the optical flow $v_{0\to2}$ has a lower quality than the optical flow $v_{0\to1}$ over smaller distance, e.g., it looks more blurred and it failed to estimate motion for some fine details such as the girl's leg and ear. The sequentially warped optical flow, by contrast, has unexpected doubling artifacts such as the second girl, left paw or left horn. These artifacts are caused by the fact that the objects and the background move in the opposite directions [44]. These sharp doubling artifacts in the sequentially warped optical flows lead to whole regions in the interpolants to be moving wrongly and separately over time.

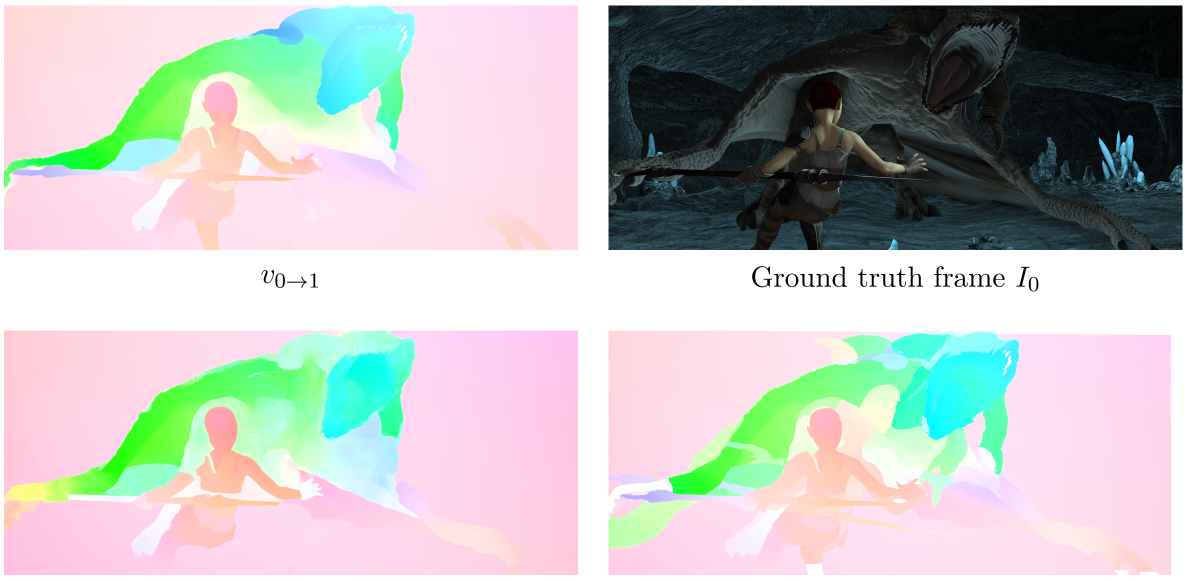


Fig. 5.4: `Below`: Direct and sequentially warped optical flows. `Above`: Optical flow between adjacent frames and the first of these frames.

5.3.3 Warped vs. Direct

Since the sequentially warped optical flows over more distant frames contain disturbing artifacts that lead to the performance decrease of frame interpolation, in this subsection we are going to compare their use against direct optical flows with the same value validity check. Table 5.3 shows the results of interpolating with direct optical flows compared with the previously used sequentially warped optical flows and, for the sake of completeness, also includes the results of the quadratic method.

The experiments show that the performance of interpolation with direct optical flows steadily declines with the increasing frames number, similar to the sequentially warped optical flow. The quadratic forward forward warping thus remains the best interpolation method in this work.

Unexpectedly, the interpolation with direct optical flows performed consistently worse in average than interpolation with sequentially warped optical flows, except for the case of degree three. Thus, we would like to also take a look at the results of every single scene. Table 5.4 contains the separate results for every scene for all polynomial degrees from one to seven and, where applicable, the warped (W) or direct (D) variant. The quadratic method performs better than all other methods for most scenes and interpolation with direct optical flows with degree four is the best for the remaining scenes while also performing mostly the best among the methods with degree higher than two.

The direct version performs better in `26` experiments with different scenes and degrees, the sequentially warped version performs better in `20` experiments, in four experiments both

Polynomial degree	Optical flow type	RMSE	SSIM
2	-	**16.272**	**0.784**
3	Warped	17.037	0.771
	Direct	16.900	0.772
4	Warped	16.853	0.764
	Direct	17.050	0.761
5	Warped	17.275	0.753
	Direct	17.492	0.746
6	Warped	17.378	0.747
	Direct	17.760	0.736
7	Warped	17.540	0.741
	Direct	17.968	0.727

Table 5.3: Results of forward forward warping with warped and direct optical flows after inpainting for different polynomial degrees.

methods have almost identical or controversial results. The SSIM differences were considered more important than the RMSE differences. Although direct version has more experiments where it has better results, its performance in the scene `ambush_2` is significantly worse and strongly influences the average results, leading to direct version being worse than sequentially warped version on average.

The distinctive feature of the scene `ambush_2` is that due to the large camera motion big objects leave the scene, making their motion estimation extremely difficult. While optical flows of adjacent frames have sufficient quality for these objects, optical flows of more distant frames are considerably worse. When an intermediate position in sequentially warped version lands outside the image domain, the following optical flow is simply not used. This validity test however is not as useful for the direct version and wrong optical flow values greatly influence the motion functions. Explicit optical flow values consistency check can potentially help to improve performance.

5.3.4 Direct Optical Flow with Consistency Check

In this last subsection we conduct experiments with direct optical flows and occlusion maps generated via forward-backward consistency checking [44] as follows:

$$\vec{v}_{i\to j}(\vec{x})\ \textit{consistent} \Leftrightarrow ||\vec{v}_{i\to j}(\vec{x}) + \vec{v}_{j\to i}(\vec{x} + \vec{v}_{i\to j}(\vec{x}))|| \leq 2, \tag{5.5}$$

where $||(a,b)|| = \sqrt{a^2+b^2}$ and i, j are different frame numbers. In this way we exclude from calculation the optical flow values where the target position $\vec{x} + \vec{v}_{i\to j}(\vec{x})$ lands outside the

Method	ambush_2		ambush_6		bamboo_2		cave_2		cave_4	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
1	27.362	0.485	21.056	0.313	10.346	0.952	15.217	0.549	19.018	0.544
2	**22.928**	**0.699**	**16.288**	**0.528**	**10.213**	**0.953**	8.961	0.840	**17.226**	0.646
3W	25.798	0.698	17.885	0.450	10.437	0.952	8.820	0.850	17.880	0.629
3D	26.086	0.663	17.306	0.471	10.348	**0.953**	8.782	0.850	17.637	0.643
4W	24.999	0.608	17.899	0.482	10.259	**0.953**	8.753	**0.851**	17.703	0.630
4D	28.179	0.532	17.398	0.480	10.258	**0.953**	**8.751**	**0.851**	17.565	**0.647**
5W	25.662	0.560	17.950	0.464	10.516	0.951	8.819	0.850	18.418	0.613
5D	29.891	0.492	18.424	0.471	10.353	0.952	8.768	**0.851**	18.077	0.631
6W	25.754	0.520	18.020	0.469	10.536	0.951	8.828	0.849	18.537	0.604
6D	31.281	0.427	18.691	0.469	10.356	**0.953**	8.769	**0.851**	18.096	0.627
7W	26.070	0.490	18.291	0.470	10.586	0.950	8.844	0.849	18.793	0.593
7D	31.281	0.427	18.691	0.469	10.356	**0.953**	8.769	**0.851**	18.096	0.627

Method	market_2		market_6		mountain_1		temple_2		temple_3	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
1	11.650	0.839	24.404	0.737	6.158	0.865	23.533	**0.861**	29.497	0.625
2	11.229	0.876	**18.511**	**0.834**	**5.703**	**0.870**	**22.992**	0.854	28.667	**0.741**
3W	12.013	0.871	19.433	0.829	5.889	0.869	23.727	0.849	28.484	0.714
3D	11.242	0.878	19.340	0.824	6.083	0.869	23.470	0.852	28.707	0.718
4W	11.938	0.872	19.423	0.827	5.858	0.869	23.387	0.846	28.708	0.708
4D	**11.202**	**0.879**	19.150	0.823	5.983	0.869	23.179	0.852	**28.334**	0.724
5W	12.500	0.868	20.191	0.821	5.926	0.869	23.732	0.841	29.038	0.690
5D	11.325	0.876	20.130	0.810	6.033	0.869	23.418	0.851	28.501	0.659
6W	12.722	0.866	20.251	0.820	5.966	0.868	23.725	0.838	29.437	0.680
6D	11.285	0.876	20.434	0.805	6.034	0.869	23.397	0.850	29.253	0.629
7W	12.978	0.864	20.461	0.818	5.989	0.868	23.810	0.834	29.577	0.671
7D	11.285	0.876	20.434	0.805	6.034	0.869	23.397	0.850	29.253	0.629

Table 5.4: Separate results for every scene of forward forward warping with warped and direct optical flows after inpainting for different polynomial degrees. For the completeness of comparison, linear and quadratic methods are also included, although they use optical flows only of adjacent frames. The best results for the scene are marked in bold, the best results for the scene among the methods with degree higher than two are underlined.

valid image domain or is covered by another object and thus has inconsistent opposite flow value.

Table 5.5 contains the results of consistency check application for the forward forward warping with direct optical flows and polynomial degree four, compared with the same method without consistency check and quadratic forward forward warping. The consistency check improves the performance of frame interpolation with degree four and it reaches the level of quadratic forward forward warping. However, since it is more computationally expensive due to additional optical flow estimation and consistency check, the quadratic forward forward warping remains the preferred frame interpolation method in this work.

Method	RMSE	SSIM
2	16.272	**0.784**
4D	17.050	0.761
4D with consistency check	**16.255**	0.779

Table 5.5: Results of forward forward warping for the degrees two and four with direct optical flows with and without consistency check.

5.4 Modifications

This section focuses on the experiments concerning the modifications of the basic approach consisting of single pixel artifact reduction modification and fusing blur reduction modification from Section 3.5. The test data is the same as in the previous section.

5.4.1 Single Pixel Artifact Reduction

The inaccuracies during the warping process often lead to single pixel artifacts, and in this subsection, we evaluate the advantages of using the modification proposed in Subsection 3.5.2 to reduce these artifacts. Table 5.6 shows the results of using single pixel artifact reduction modification. The modification leads to slightly better results for both uni- and bidirectional methods. The SSIM index for forward warping stays almost identical, possibly because the modification impact is rather small for a large number of undefined pixels.

This modification shows a performance increase for both warping variants and both quality measures, even though a rather small number of pixels is affected. Although the quantitative quality increase is comparably small, the advantages of using this modification are obvious for a subjective observer (compare Fig. 3.12 and Fig. 3.11). The objective quality measures generally have rather low correlations with the evaluations made by human observers [31]. Thus, this modification is beneficial despite small quantitative result improvement, since it reduces eye-catching single pixel artifacts.

Modification use	FW		FFW	
	RMSE	SSIM	RMSE	SSIM
-	26.277	0.732	16.272	0.784
+	**26.239**	**0.733**	**16.169**	**0.789**

Table 5.6: Results of frame interpolation with and without single pixel artifact reduction modification on the example of quadratic forward and forward forward warping after inpainting.

5.4.2 Fusing Blur Reduction

The fusing of interpolants can lead to notable blur due to the different positions of the same object in the interpolants. Table 5.7 shows the results of using fusing blur reduction modification introduced in Subsection 3.5.1. The decision to use one interpolant as main interpolant leads to worse results compared to fused interpolants.

Method	RMSE	SSIM
Quadratic FFW	**16.272**	**0.784**
Quadratic FFW with main forward interpolant	20.793	0.751
Qudratic FFW with main backward interpolant	20.479	0.751

Table 5.7: Results of frame interpolation with fused interpolants, main forward interpolant, and main backward interpolant on the example of quadratic forward forward warping after inpainting.

We are also interested in the temporal performance of the methods researched in this subsection, thus Fig. 5.5 visualizes how the assessment measures' values of the previous table change over time for a single scene. The results of using one of the interpolants as the main interpolant strongly resemble the results of forward warping, as they tend to get worse for bigger distance from the sample frame. Even using main forward interpolant for time steps smaller than 0.5 and then switching to the main backward interpolant for bigger time steps would yield worse results than fusing the interpolants for every time step.

This behavior is caused by the transparency artifacts that become more obvious when using only one interpolant, while fusing two interpolants effectively counteracts this problem by weighting the interpolant that is closer to its sample frame stronger. Fig. 5.6 illustrates this difference in the approaches. All methods have transparency artifacts in the interpolants for the time step $t = 0.5$. However, while for fused interpolants these artifacts are the worst for this time step in the middle between input frames and the quality gets better when nearing the frames, for main interpolant methods the transparency artifacts continue to get worse.

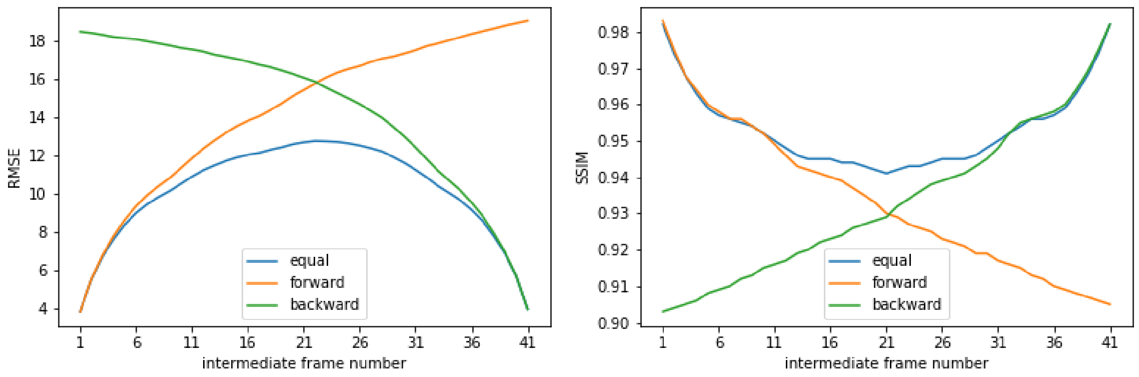


Fig. 5.5: Visualization of quality assessment measures' values changing over time, calculated for 41 intermediate frames after inpainting. The frame interpolation is performed with quadratic forward forward warping with fused interpolants, main forward interpolant, and main backward interpolant for the scene *bamboo_2*.

Fig. 5.7 visualizes the pixel origins for the three approaches to further point out their differences. The methods with a main interpolant acquire most pixel values from it, using the supporting interpolant only at places where the main interpolant is undefined. The method with fused interpolants mostly uses values obtained through fusion, leveraging information from a single interpolant only when the other is undefined. The areas that need to be inpainted afterwards are the same for all methods.

5.5 Summary

In this chapter, we examined frame interpolation methods using different polynomial degrees. According to the experiments, the methods that are leveraging information from additional frames perform better than the methods using only two frames. However, the performance deteriorates with increasing number of frames due to sequential warping of optical flows and using less precise optical flows over more distant frames. Even applying the consistency check for forward forward warping with direct flows with degree four did not outperform the quadratic forward forward warping.

Several modifications of interpolation methods were also tested. Although the fusing blur reduction modification did not prove useful, the single pixel artifact reduction modification slightly improves performance and, more importantly, removes eye-catching artifacts.

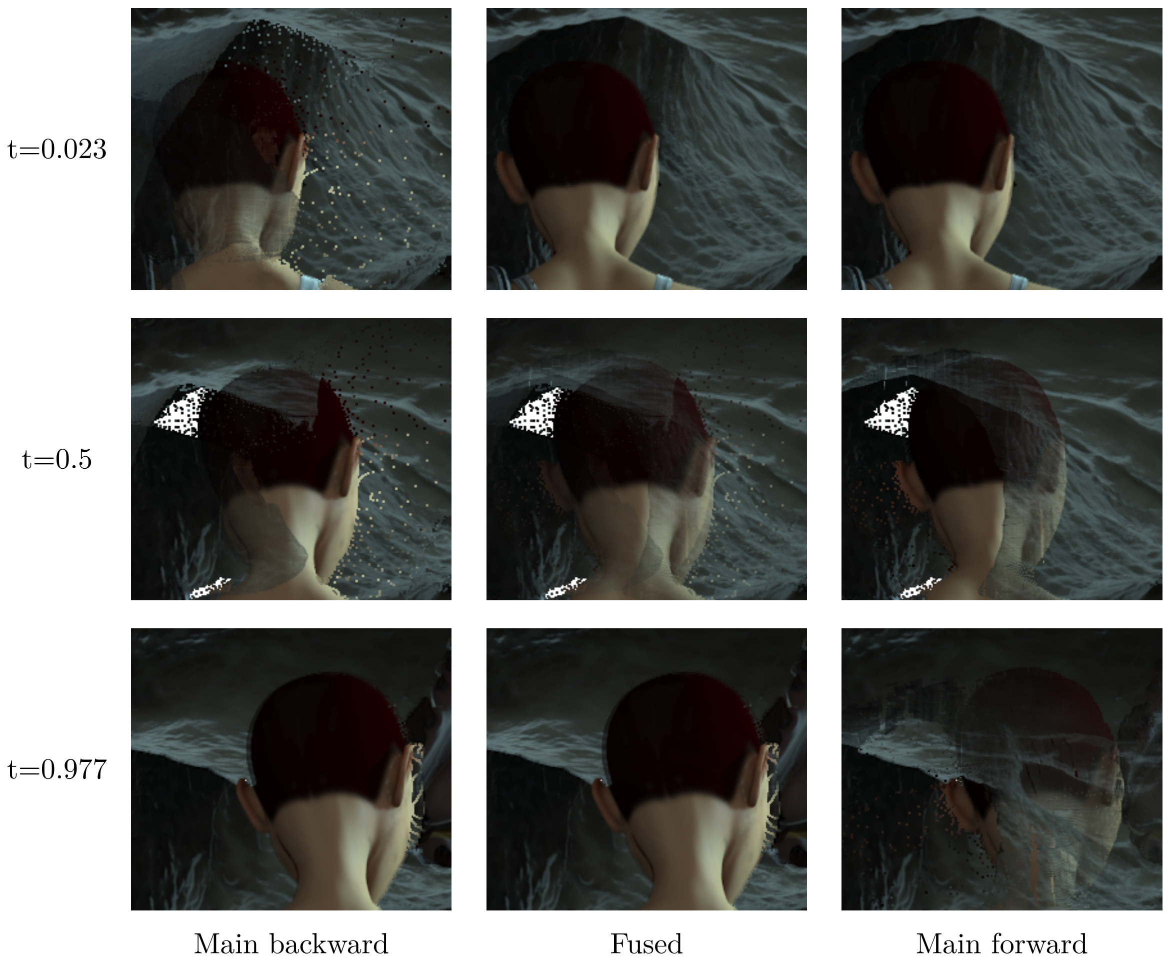


Fig. 5.6: Results of interpolation with quadratic forward forward warping with fused interpolants, main forward interpolant, and main backward interpolant for the scene *cave_4* and the time step $t = 0.977$ before inpainting. Undefined pixels are marked in white.

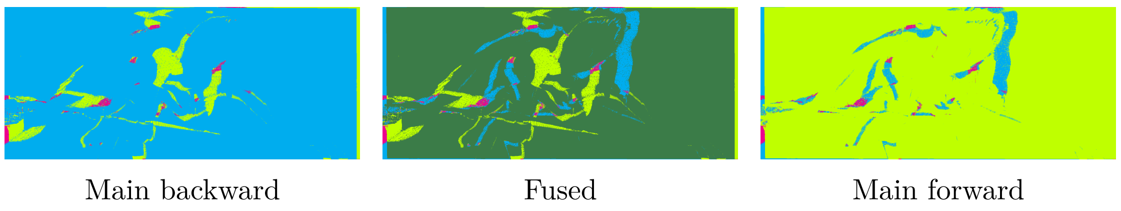


Fig. 5.7: Visualization of pixel origins for frame interpolation with fused interpolants, main forward interpolant, and main backward interpolant for the scene *cave_4*. Lime and blue colors denote the pixels directly acquired from forward and backward interpolants respectively. The pixel values obtained through fusion are marked in dark green color. Pink denotes the remaining inpainted pixels.

6 Conclusion

In this thesis, motion-compensated frame interpolation methods using multiple frames were introduced and evaluated in order to answer the three questions we initially raised. Firstly, we were interested in the different ways of using multiple frames. The extension of linear interpolation methods to the quadratic case was introduced and generalized for a polynomial of an arbitrary degree to approximate the motion between frames more precisely. Two different approaches using sequentially warped or direct optical flows along with a consistency check were presented to make use of more distant frames.

Then we conducted experiments using up to eight frames to find out whether leveraging additional temporal information helps to increase the interpolant quality. According to the previous chapter, the use of more than two frames indeed leads to a performance increase, confirming the main assumption of this work. The results show that approximating the object motion by using three frames for one direction improved the RMSE by 6% and the SSIM by 12% for the simple forward warping that interpolates the frame from a single direction. Both RMSE and SSIM were improved by 13% for the forward forward warping that interpolates from both directions, when compared to the linear methods using only two frames on the high frame rate version of MPI Sintel dataset [20].

The quadratic forward forward warping is the best method in this thesis since the use of more than three frames for motion trajectory estimation leads to a decrease in performance due to the difficulties in using multiple frames that were our another point of interest. The main problem of using additional frames is the need to acquire position information from more distant frames for motion function estimation. The idea of using sequentially warped positions resulted in eye-catching artifacts when whole regions in the interpolants move wrongly and separately. The attempt to use direct optical flows requires motion estimation over longer distance which results in optical flows of lower quality. Using such flows for position calculation with a simple value domain validity check led to severe artifacts at the boundaries. However, even the consistency check did not help to outperform the quadratic warping and the methods of degree higher than two have slow decrease of performance with the increasing degree.

Additionally, we proposed two modifications to reduce interpolation artifacts which can also be applied in other frame interpolation methods. Fusing blur reduction modification was introduced to counteract the blur caused by interpolants fusing in forward forward warping by using only one interpolant for color information and showed generally worse results than fusing. The second modification that aims to reduce eye-catching single pixel artifacts caused by inaccuracies during warping successfully increases the performance and applying it with quadratic forward forward warping yields the best results in this thesis.

6.1 Outlook

The multi-frame interpolation methods in this work leverage additional temporal information to track the object positions over several frames and approximate their motion with a polynomial of highest valid degree. However, we derive color information only from two frames like the linear methods. A potential approach would be to consider color information from the additional frames and polynomially track the luminance changes to consider them for frame interpolation.

The use of multiple frames for motion estimation can be researched further. More sophisticated policies concerning motion function calculation depending on the consistency of available optical flow values, occlusions and object segmentation could result in methods using more than three frames for motion trajectory estimation that outperform the quadratic warping. The sequentially warped optical flows may benefit from using four neighbors strategy instead of nearest neighbor strategy.

The extension to multi-frame case in this thesis was applied to basic frame interpolation methods forward and forward forward warping and brought significant performance increase. Given this success, the use of multiple frames can be also considered to increase quality of state-of-the-art machine learning-based frame interpolation methods that gained popularity due to recent deep convolutional neural networks achievements.

Bibliography

[1] Sepehr Aslani and Homayoun Mahdavi-Nasab. Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical and Computer Engineering*, 7(9):1252 – 1256, 2013.

[2] Ivar Austvoll. A study of the yosemite sequence used as a test sequence for estimation of optical flow. In *Scandinavian Conference on Image Analysis*, pages 659–668. Springer, 2005.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[4] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[5] Noah Berenguel Senn. Zwischenbildinterpolation mit optischem fluss. B.S. thesis, 2020.

[6] John P Boyd. *Chebyshev and Fourier spectral methods.* Courier Corporation, 2001.

[7] Tim Brooks and Jonathan T Barron. Learning to synthesize motion blur. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6840–6848, 2019.

[8] David R. Bull. Chapter 10 - measuring and managing picture quality. In David R. Bull, editor, *Communicating Pictures*, pages 317–360. Academic Press, Oxford, 2014.

[9] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[10] Claudette Cedras and Mubarak Shah. Motion-based recognition a survey. *Image and Vision Computing*, 13(2):129–155, 1995.

[11] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. All at once: Temporally adaptive multi-frame interpolation with

advanced motion modeling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 107–123. Springer, 2020.

[13] Yang-Ho Cho, Ho-Young Lee, and Du-Sik Park. Temporal frame interpolation based on multiframe feature trajectory. *IEEE transactions on circuits and systems for video technology*, 23(12):2105–2115, 2013.

[14] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29:64–72, 2016.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[16] James J Gibson. The perception of the visual world. 1950.

[17] Chris A Glasbey and Kantilal Vardichand Mardia. A review of image-warping methods. *Journal of applied statistics*, 25(2):155–171, 1998.

[18] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[19] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019.

[20] Joel Janai, Fatma Guney, Jonas Wulff, Michael J Black, and Andreas Geiger. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3597–3607, 2017.

[21] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016.

[22] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[23] Earl J Kirkland. Bilinear interpolation. In *Advanced Computing in Electron Microscopy*, pages 261–263. Springer, 2010.

[24] R. Krishnamurthy, J.W. Woods, and P. Moulin. Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(5):713–726, 1999.

[25] Yoshihiko Kuroki, Tomohiro Nishi, Seiji Kobayashi, Hideki Oyaizu, and Shinichi Yoshimura. A psychophysical study of improvements in motion-image quality by using high frame rates. *Journal of the Society for Information Display*, 15(1):61–68, 2007.

[26] Yoshihiko Kuroki, Haruo Takahashi, Masahiro Kusakabe, and Ken-ichi Yamakoshi. Effects of motion image stimuli with normal and high frame rates on eeg power spectra: comparison with continuous motion image stimuli. *Journal of the Society for Information Display*, 22(4):191–198, 2014.

[27] Yen-Lin Lee and Truong Nguyen. High frame rate motion compensated frame interpolation in high-definition video processing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 858–861. IEEE, 2010.

[28] Bohan Li, Jingning Han, Yaowu Xu, and Kenneth Rose. Optical flow based co-located reference frame for video compression. *IEEE Transactions on Image Processing*, 29:8303–8315, 2020.

[29] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981.

[30] Hui Men, Vlad Hosu, Hanhe Lin, Andrés Bruhn, and Dietmar Saupe. Subjective annotation for a frame interpolation benchmark using artefact amplification. *Quality and User Experience*, 5(1):8, Sep 2020.

[31] Hui Men, Vlad Hosu, Hanhe Lin, Andrés Bruhn, and Dietmar Saupe. Subjective annotation for a frame interpolation benchmark using artefact amplification. *Quality and User Experience*, 5(1):1–18, 2020.

[32] Hui Men, Vlad Hosu, Hanhe Lin, Andrés Bruhn, and Dietmar Saupe. Visual quality assessment for interpolated slow-motion videos based on a novel database. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2020.

[33] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[34] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 498–507, 2018.

[35] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1418, 2015.

[36] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[37] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[38] S. Peleg and H. Rom. Motion based segmentation. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, pages 109–113 vol.1, 1990.

[39] Lars Lau Rakêt, Lars Roholm, Andrés Bruhn, and Joachim Weickert. Motion compensated frame interpolation with a symmetric optical flow constraint. In *International Symposium on Visual Computing*, pages 447–457. Springer, 2012.

[40] Syed Tafseer Haider Shah and Xiang Xuezhi. Traditional and modern strategies for optical flow: an investigation. *SN Applied Sciences*, 3(3):289, Feb 2021.

[41] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):2, 2007.

[42] Chuanxin Tang, Ronggang Wang, Wenmin Wang, and Wen Gao. A new frame interpolation method with pixel-level motion vector field. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 350–353. IEEE, 2014.

[43] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[44] Patrick Tobien. Multi-frame approaches for learning optical flow predictions. Master's thesis, 2020.

[45] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[46] Demin Wang, Liang Zhang, and André Vincent. Motion-compensated frame rate up-conversion—part i: Fast multi-frame motion estimation. *IEEE Transactions on Broadcasting*, 56(2):133–141, 2010.

[47] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. Optical flow guided tv-l 1 video interpolation and restoration. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 273–286. Springer, 2011.

[48] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[49] Tianfan Xue, Jiajun Wu, Katherine L Bouman, and William T Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *arXiv preprint arXiv:1607.02586*, 2016.

[50] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32:794–805, 2019.

[51] Jiefu Zhai, Keman Yu, Jiang Li, and Shipeng Li. A low complexity motion compensated frame interpolation method. Institute of Electrical and Electronics Engineers, Inc., May 2005.

[52] Haoxian Zhang, Ronggang Wang, and Yang Zhao. Multi-frame pyramid refinement network for video frame interpolation. *IEEE Access*, 7:130610–130621, 2019.