Institute of Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelor's Thesis

Analysing the Quality of Interframe Interpolation with Optical Flow

Katarina Baričová

Course of Study:	Informatik
Examiner:	Prof. Dr. Andrés Bruhn
Supervisor:	Jenny Schmalfuß, M. Sc.
Commenced:	7 January, 2021
Completed:	7 July, 2021
CR-Classification:	I.4.8

Abstract

The generation of an intermediate frame between two consecutive frames in a video is an ongoing challenge in computer vision. This concept is known as motion interpolation, or interframe interpolation, and usually works by following the path of an optical flow. To obtain accurate interpolated images, a good optical flow estimation is needed. Typically, it is assumed that the better a flow estimation is, the better the resulting interpolated images are. To test how the underlying optical flow influences the interpolation results, we look at two different optical flow estimations, RAFT [1] and the baseline version of ProFlow [2], and perform various interpolation experiments with them.

In the experiments, we first examine the accuracy of the underlying flows estimated with RAFT and the ProFlow baseline by comparing their EPE and flow visualizations. Following this, we conduct several interpolation experiments with these flows by using different interpolation methods. Here, the focus lies on two methods called forward warping and forward forward warping, which are based on the concept of image warping. For a wide variety of datasets, we use three different benchmarks, Middlebury [3], KITTI [4], and Sintel [5]. In an oracle experiment, we then compare how the interpolation performs when ground truth flows are used and investigate different approaches to change these results.

From these experiments, we find that a RAFT flow does not always produce better interpolation results than a ProFlow baseline flow, despite being more accurate. Furthermore, the oracle experiments show that using a more accurate flow estimation often creates more doubling artifacts in the interpolated image and thus a better flow does not always result in a better interpolation quality.

Kurzfassung

Die Erzeugung eines Zwischenbildes zwischen zwei aufeinanderfolgenden Bildern in einem Video ist eine andauernde Herausforderung in der Computer Vision. Dieses Konzept ist als Motion Interpolation oder Interframe Interpolation bekannt und funktioniert in der Regel, indem es dem Pfad eines optischen Flusses folgt. Um akkurate interpolierte Bilder zu erhalten ist eine gute Schätzung des optischen Flusses erforderlich. Normalerweise wird davon ausgegangen, dass die interpolierten Bilder umso besser sind, je besser die Schätzung des Flusses ist. Um zu testen, wie der zugrundeliegende optische Fluss die Interpolationsergebnisse beeinflusst, betrachten wir zwei verschiedene optische Flussschätzungen, RAFT [1] und die Baseline Version von ProFlow [2] und führen mit ihnen verschiedene Interpolationsexperimente durch.

In den Experimenten untersuchen wir zunächst die Genauigkeit der zugrundeliegenden Flüsse, die mit RAFT und der ProFlow Baseline geschätzt wurden, indem wir deren EPE und Flussvisualisierungen vergleichen. Anschließend führen wir mehrere Interpolationsexperimente mit diesen Flüssen durch, indem wir verschiedene Interpolationsmethoden verwenden. Der Fokus liegt hierbei auf zwei Methoden, dem sogenannten Forward Warping und dem Forward Forward Warping, die auf dem Konzept des Image Warpings basieren. Für eine Vielzahl von Datensätzen verwenden wir drei verschiedene Benchmarks, Middlebury [3], KITTI [4] und Sintel [5]. In einem Orakel-Experiment vergleichen wir schließlich, wie sich die Interpolation verhält, wenn Ground Truth Flüsse verwendet werden und untersuchen verschiedene Ansätze, um diese Ergebnisse zu verändern.

Aus diesen Experimenten geht hervor, dass ein RAFT Fluss nicht immer bessere Interpolationsergebnisse als ein ProFlow Baseline Fluss liefert, obwohl er genauer ist. Des Weiteren zeigen die Orakel-Experimente, dass die Verwendung einer genaueren Flussschätzung oftmals mehr Doppelungsartefakte im interpolierten Bild erzeugt und somit nicht immer zu einer besseren Interpolationsqualität führt.

Contents

1 Introduction

Creating new frames in a low frame rate video can be of high importance in many fields [6, 7, 8, 9]. This can be achieved with an interframe interpolation [10]. Interframe interpolation is able to generate an intermediate frame between two consecutive frames in a frame sequence. Figure 1.1 shows this process, where a new image is synthesized between image I_0 and image I_1. The scalar value $t \in (0, 1)$ indicates the temporal distance from image I_0 to the interpolated image I_t. Analogously, the temporal distance from the interpolated image I_t to image I_1 is $1 - t$.

Using interframe interpolation to increase the frame rate in videos and thus providing the perception of a more fluid motion can be beneficial in many areas. Besides using generated intermediate frames on television and computer screens for a smoother visual appearance of low frame rate films, it is also used in interframe video compression [6]. To decrease the needed data rate in video compression, only a series of anchor frames get encoded. The remaining intermediate frames are then reconstructed with an interpolation. This allows higher compression rates than an intraframe compression, where every single frame gets encoded. Additionally, generating a higher frame rate can also be beneficial to obtain sharp slow-motion effects [7] of videos with a lower frame rate. Furthermore, interpolation is also used in virtual reality [8] where multiple cameras are used in a ring position. To combine the different views of adjacent cameras, interpolation is used to generate the needed intermediate frames. Not only entertainment media benefit from this concept, but certain areas of the medical field can also take advantage of interpolation. For example, a capsule endoscopy [9] uses a low frame rate to last a whole traversing of the human body. This results in poor visual effects and can lead to a harder diagnosis. Providing a higher frame rate via frame interpolation allows for a more realistic 3D view of the digestive system and thus can result in a better diagnosis.

However, creating interpolated frames that are of good quality and do not require an immense amount of computational power is an ongoing challenge. The easiest way of generating an intermediate frame between frames I_0 and I_1 are methods that ignore the motion taking place in the given frames, such as frame repetition and frame averaging [11]. In frame repetition, the previous frame gets repeated and thus the interpolated frame corresponds to frame I_0. Frame averaging generates the intermediate frame by overlaying frame I_0 and frame I_1 on top of each other. While these methods are easy and might work for static scenes, they generally produce blurring effects around moving objects and are thus not optimal for videos with larger motion. This shows the need for more advanced approaches, which take the motion of objects

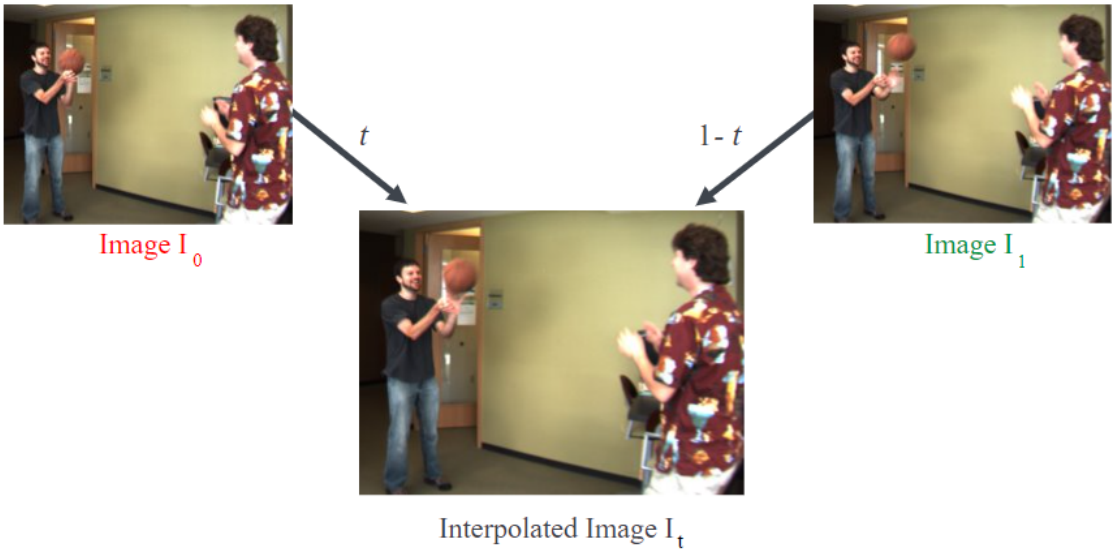


Figure 1.1: Concept of interframe interpolation shown for the scene *'Basketball'* of Middlebury benchmark [3]. Intepolated image I_t gets generated between image I_0 and image I_1 at time t.

in a series of frames into consideration. Such an approach is known as motion compensated interpolation. For this, the motion that takes place between a frame and its consecutive frame has to be correctly estimated. This motion estimation is done with an *optical flow*. Optical flow describes the motion of objects between two consecutive frames. In this thesis, we look at two methods for estimating such optical flow, called RAFT [1] and the baseline version of ProFlow [2]. The idea is to take these flow estimations and perform an interpolation along their trajectory. A key aspect of this is the concept of image warping. In image warping, positions of an image get mapped to new positions with a mapping function, resulting in a warped image. Such image warping can use the optical flow as the mapping function. This concept is then used as an interpolation method in various ways. Two of these methods that make use of this image warping concept are the forward warping and forward forward warping. These serve as the two main methods used throughout this thesis.

There are many challenges to interframe interpolation. For example, performing a warping can lead to collisions, caused by multiple pixels being mapped to one pixel. To deal with this, we look at different strategies for such collisions. Furthermore, while multiple pixels can be mapped to one pixel, the other problem arises when a pixel has no pixel that gets mapped to it. This creates holes in the warped image that have to be filled with a so-called inpainting method. For this, we look at one inpainting approach based on diffusion. A further problem that is discussed in this thesis, are occluded areas in frames. To detect such occluded areas

we use the forward-backward consistency check to generate occlusion maps. To evaluate the interpolation results, we look at three different metrics, the SSIM, RMSE, and VMAF.

The main goal of this thesis is to analyze how the quality of an optical flow estimation influences the interpolation results. Intuitively, one would expect that using a more accurate optical flow estimation results in a more accurate interpolated image. To test this, we use the two previously mentioned optical flow estimations RAFT and ProFlow baseline and perform various interpolation experiments with them. Here, we first look at the main differences between the flow estimations and examine how these differences play a role in the interpolation results. In his work [12], N. B. Senn already compared how different interpolation strategies perform when using the ProFlow baseline flow. For this, a variety of scenes were used from the Middlebury benchmark [3]. To build on this, we perform these same interpolation methods on the same scenes, using a RAFT flow. Here, as mentioned above, the focus lies on the forward warping and forward forward warping method. We then compare the results generated with a RAFT flow and a ProFlow baseline flow. To obtain more data, we use additional benchmarks, KITTI [4], and Sintel [5]. Furthermore, we then use the ground truth flows provided with the Sintel benchmark to conduct an oracle experiment and investigate further how the accuracy of an optical flow influences the resulting interpolation.

Structure

The main part of this thesis consists of two parts, a theory part, and an experiment part. In the first part, we introduce all the theoretical background that is needed for the experiments. Here, we first discuss the notion of optical flow in Chapter 2. This chapter consists of background information, a brief overview of possible approaches, and possible challenges. Additionally, we also introduce the two flow estimations RAFT [1] and ProFlow baseline [2], as well as the endpoint error metric (EPE). In Chapter 3, we then look at how interpolation works, discuss the concept of image warping, and introduce different interpolation methods based on this concept, with the two main methods being forward warping and forward forward warping. Furthermore, the concept of splatting and inpainting and their possible strategies are discussed as well. A method for generating occlusion maps called the forward-backward consistency check is also introduced in this chapter. Lastly, we introduce three error metrics, RMSE, SSIM, and VMAF, which are used for the evaluation.

The second part of this thesis contains the conducted experiments, shown in Chapter 4. Here, we first compare the flow estimations of RAFT and ProFlow baseline, then we perform several interpolation experiments, and examine the results. Finally, we perform an oracle experiment with ground truth flows and investigate the showcased results further. In a final summary of the experiments, we then gather all observed results.

2 Optical Flow

Interframe interpolation uses optical flow to estimate motion between consecutive frames to then warp one or both frames to generate an interpolated frame, making optical flow one central notion of this thesis. This chapter gives details on what optical flow is and where it is used, discussed in Section 2.1, what methods exist for estimating optical flow, Section 2.2, as well as what makes its estimation challenging, Section 2.3. A central question of this thesis is whether the quality of an optical flow has an influence on the results of an interpolation. For this, we introduce two separate methods for estimating optical flow, the ProFlow Baseline in Section 2.4, and RAFT in Section 2.5. Lastly, the endpoint-error (EPE) metric to determine the quality of an optical flow is explained in Section 2.6. This metric is later used to determine which optical flow is more accurate.

2.1 Background

Estimating the motion of objects in a moving scene is a fundamental problem in computer vision. Such estimation is known as an optical flow. This term goes back to as late as the 1950s where American psychologist J. J. Gibson described in his work [13] the notion of optical flow as a technique used by animals for obstacle avoidance to navigate through the world. Different scientific fields made use of this notion to estimate the apparent motion of objects in a visual scene. A high-level definition describes an optical flow as a vector field for two consecutive images in a sequence [14]. This vector field consists of a displacement vector $\overrightarrow{v}(x, y)$ for each position (x, y) which indicates its displacement from the first frame to the consecutive frame. Each displacement vector is defined as the motion of position (x, y) in terms of a horizontal displacement u and a vertical displacement v. This is described as follows:

$$\overrightarrow{v}(x, y) = \begin{pmatrix} u \\ v \end{pmatrix}. \tag{2.1}$$

Figure 2.1 shows the position (x, y) in image I_0 and its displacement vector $\overrightarrow{v}(x, y) = (u, v)^T$. Position (x, y) in image I_0 thus corresponds to the position $(x + u, y + v)$ in the consecutive image I_1.

Such flow fields can either be sparse and only define vectors for important points of the frame, or they can be dense when a displacement vector is defined for every single pixel.

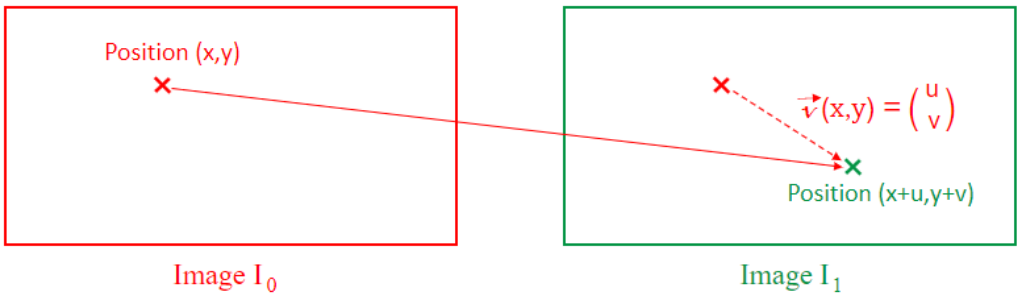


Figure 2.1: Displacement vector $\overrightarrow{v}$ for the position (x, y) in image I_0 and its corresponding position $(x + u, y + v)$ in image I_1.

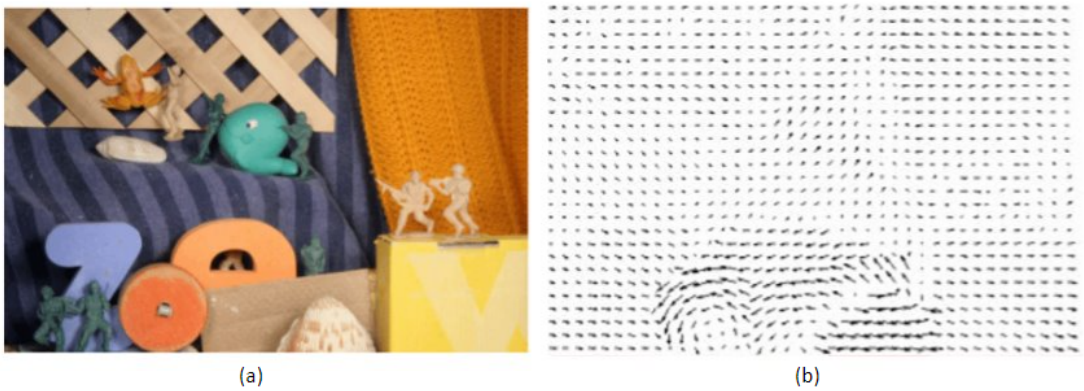


Figure 2.2: Sparse optical flow field (b) for the *Army* scene (a) from the Middlebury benchmark [3]. Image source for (b): [15].

Figure 2.2 (b) shows such a sparse vector field for the scene Army of the Middlebury benchmark [3]. In the scene, the objects at the bottom center of the image move the most from the first frame to the second. Thus, in this area, one can see the vectors with the largest length. Using optical flow for interpolation of images requires a dense flow field. However, visualizing them in the same way as in Figure 2.2 would not be optimal. For this, there is a second option of visualizing optical flow, namely using a color code, shown in Figure 2.3 (c). Here, the different colors indicate the direction of that vector. Additionally, the richer the color is, the larger the displacement of that area. Applying this color code visualization to the optical flow of the aforementioned scene can be seen in Figure 2.3 (b).

While this thesis focuses on using optical flow for interframe interpolation, many different areas benefit and make use of an optical flow as well, such as video compression [17, 18], video annotation [19], and video segmentation [20]. Robotics research uses optical flow

Figure 2.3: Optical flow field (a) visualized with a color code (c) for the scene *Army* (a) from the Middlebury benchmark [3]. Image source for (c): [16].

for robot navigation and localisation [21, 22, 23]. Video surveillance utilizes optical flow to recognize actions [24, 25] or faces [26] as well as for tracking moving objects in traffic [27, 28]. In meteorology, optical flow is applied for weather forecasting [29] and even in biology, optical flow can be applied in many different areas such as the measurement of plant growth [30] or even analyzing the swarm of bees [31]. Another prominent field that uses optical flow is fluid mechanics, to measure the velocity of fluids [32]. In recent years even medical imaging was able to benefit from Optical Flow for example by using it to analyze and diagnose different forms of cancer [33, 34].

2.2 Methods for Optical Flow Estimation

Over the years there have been various approaches for generating optical flow fields. The most prominent being differential methods. These differential methods consist of two subclasses, global and local techniques. Both are based on a data term that expresses a brightness constancy assumption. Local techniques are based on the assumption that the flow is constant within a small neighborhood of the considered pixel. A well-known example of this technique is the Lucas–Kanade method [35], introduced by B. D. Lucas and T. Kanade. This method minimizes a local energy and since it consists of more equations than unknowns, these equations are solved with a least squares method. While this method is easy to implement and robust under noise [36], it only generates sparse flow fields, which may not be optimal in many cases. For the global techniques, the most prominent method is the one introduced by B. Horn and B. Schunk [37]. This method works as a variational approach, which computes an optical flow field by minimizing a suitable global energy functional, consisting of not only the data term but an additional smoothness term. This smoothness term penalizes deviations from smoothness. While this method produces dense flow fields, it is not as robust to noise [36].

In recent years a different approach emerged from deep learning, providing even more promising results than the classical approaches. End-to-end convolutional neural networks (CNN) are trained on datasets to recognize patterns and estimate optical flow for a pair of input frames. FlowNet [38] and its successor FlowNet2 [39] paved the way for such CNN based optical flow estimations by generating promising results. One such CNN based method for generating optical flow, called RAFT [1], is introduced in Section 2.5.

While CNN based optical flow estimations provide high-quality results, they have their drawbacks as well. For a supervised neural network to recognize patterns to an extent that will deliver good results, the network needs a large amount of training data with respective ground truth optical flows. These ground truth optical flows however are hard to generate for most real image sequences. Hence why most supervised CNN based methods train on synthetic datasets, that include these sought-after ground truth flows. However, these synthetic datasets often struggle to recreate the complexity of real-world imagery with its brightness variations, shadows, blur, or noise. Thus, training on synthetic data can cause difficulties for creating an optical flow for real-world sequences. To avoid this problem, an unsupervised approach for optical flow estimation with neural networks was introduced in recent years [40, 41]. While this approach evades the need for ground truth flows, many of these approaches don't achieve the same accuracy as supervised approaches.

So far the methods for generating optical flow were explained in terms of two input frames. Sometimes it can be useful to use multiple input frames to overcome challenges such as occlusion and disocclusion. ProFlow [2] is one such method that uses three input images instead of just two. Its baseline version is discussed in Section 2.4.

2.3 Challenges for Optical Flow Estimation

Since optical flow aims to capture 3D motion in a 2D setting there are many challenges in the generation of optical flow. Firstly, optical flow estimates a linear motion. While this may be sufficient for small motions between frames, it can cause problems for larger nonlinear motion. This problem and its effects on the resulting interpolation are further explained in Section 3.6. Additionally, optical flow methods typically estimate the motion between two consecutive frames without consideration of the motion after or before that. However, this additional information might be useful in many cases. Occlusions and disocclusions play a further role in the quality of the optical flow. In the case of occlusions in the second frame, the question arises of how to determine where certain areas moved from the first frame to the second, if this area is not visible anymore in the second frame. Analogously, in case an area gets disoccluded in the second frame, meaning it becomes visible again, the question is how it can be determined to which area this corresponds to in the first frame. Obtaining

the information of occluded and disoccluded areas can thus be highly beneficial. For this, a method for generating occlusion maps is given in Section 3.7.

2.4 The ProFlow Baseline

To analyze how the quality of an optical flow influences the quality of the interpolation, we need different optical flow estimation methods. The first method for generating optical flow which is used in our conducted experiments is the baseline version of ProFlow [2]. Unlike most methods which use two input frames, it uses three input frames $t-1$, t, and $t+1$. For this, ProFlow generates the forward flow from frame t to frame $t+1$ and the backward flow from frame t to frame $t-1$. Normally, ProFlow uses an unsupervised CNN embedded in a pipeline approach to estimate optical flow. However, for the later shown experiments, only the baseline version of ProFlow was used which omits a CNN based step and uses only the traditional pipeline steps to generate an optical flow. Such a pipeline approach, as proposed by J. Revaud *et al*. [42], consists of four main steps, (1) *matching*, (2) *outlier filtering*, (3) *inpainting*, and (4) *variational refinement*. The results of each step can be seen in Figure 2.4. While step (2) is performed analogously to the method used in [42], for steps (1), (3), and (4) ProFlow uses more recent methods. In the following, each step is examined closer.

(1) **Matching:**
Starting with the first step, matching, ProFlow uses a coarse-to-fine PatchMatch approach of Y. Hu *et al.* [43] to generate matches, which form an initial sparse flow. PatchMatch algorithms find correspondences, so-called *matches*, between regions, or *patches*, of images. The idea of a coarse-to-fine approach is to use resolution pyramids, starting at the top level, which corresponds to the lowest resolution, to obtain satisfying results for large motion. For this, the input images are downsampled with an appropriate factor. On each level, a PatchMatch is performed iteratively based on an initial flow. The resulting matches of each level then serve as the initial flow for the lower level. The resulting initial flow field however consists of outliers that need to be removed.

(2) **Outlier Filtering:**
After generating these sparse flows in the first step, the second step performs outlier filtering with a bi-directional consistency check. Such consistency checks check whether following the forward flow from a starting position and then going back with the backward flow results in the same starting position. This step is done analogously to the outlier filtering performed in the original pipeline approach by J. Revaud *et al*. [42].

(3) **Inpainting:**
As this initial flow is only sparse, the third step consists of inpainting the matches in order to obtain a dense flow field. This is performed with a robust interpolation technique [44], introduced by Y. Hu *et al.* Here, the input image is over-segmented into superpixels and

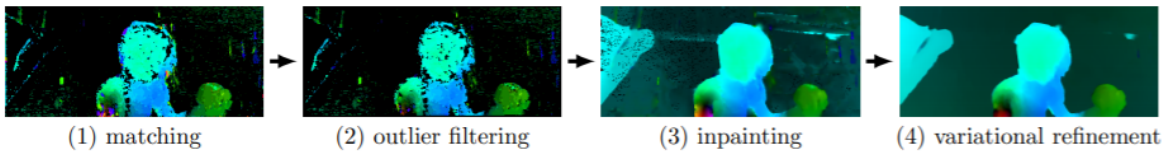


Figure 2.4: Four steps of a pipeline. Image source: [42].

a flow model for each superpixel is estimated. The estimation of such models relies on the matches of their neighbor superpixels.

(4) **Variational Refinement:**
Finally, in the last step, an order-adaptive illumination-aware refinement of D. Maurer *et al*. [45] is performed to further refine the flow. For this, a variational energy functional is minimized.

While this method for estimating optical flow delivers good results, it is not optimal for larger motion and outlines of moving objects. New methods emerged in recent years which provide better results for these problems. One of these new approaches is presented in the following.

2.5 Recurrent All-Pairs Field Transforms (RAFT)

The second method for optical flow estimation that is used in the experiments is the recently introduced recurrent all-pairs field transforms (RAFT) [1], by Z. Teed and J. Deng. Unlike the ProFlow baseline approach, RAFT uses a neural network to generate optical flow. Similar to all supervised CNN based methods, RAFT needs datasets with ground truth flows to train the network. The architecture of RAFT consists of three main parts, (1) a *feature* and *context encoder*, (2) a *correlation volume*, and (3) an *update operator*. This architecture can be seen in Figure 2.5. Each component is briefly discussed in the following:

(1) **Feature and Context Encoder:**
The feature encoder extracts 256 features from each pixel of both input frames which are first downsampled to an $\frac{1}{8}th$ resolution. The context encoder does the same, but with only the first input frame.

(2) **Correlation Volume**
In the next step, 4D correlation volumes are calculated. A correlation volume consists of the scalar product of all pairs of the feature vectors that were extracted in step (1). The dimension of such a correlation volume is thus H×W×H×W, with H being the height of

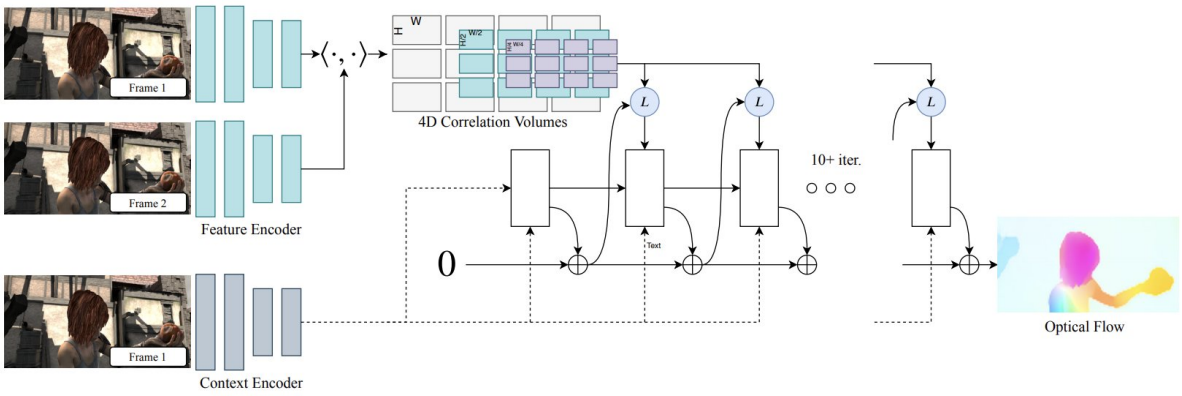


Figure 2.5: Architecture of RAFT, consisting of (1) a feature and context encoder, (2) 4D correlation volumes, and (3) an update operator. Image source: [1].

the downsampled frames and W the width. Additionally, to a H×W×H×W correlation volume, three more correlation volumes are calculated where the last two dimensions are pooled with kernel sizes 2,4, and 8. This results in four correlation volumes with the first volume having dimensions H×W×H×W, the second with H×W×H/2×W/2, the third with H×W×H/4xW/4, and the fourth with H×W×H/8×W/8. These four correlation volumes then form a 4D correlation volume. The different resolutions provide the benefit of detecting the motion of small objects as well as coping with large displacement.

(3) **Update Operator**
The update operator uses a modified gated recurrent unit (GRU) [46] to iteratively generate the flow. For this, the flow gets initialized at zero and for each iteration, the update operator looks up values from the correlation volumes. This imitates the steps of an iterative optimization-based approach.

Since the input images are downsampled to an $\frac{1}{8}th$ resolution, the network generates an optical flow with an $\frac{1}{8}th$ resolution as well. For this, the optical flow has to be upsampled again in the final step.

Generally, optical flows estimated with RAFT are more accurate than the ones generated with the baseline version of ProFlow. A closer examination of this is given in the experiment chapter. In order to evaluate which flow is more accurate, a metric is needed for the evaluation. One such metric is introduced in the following section.

2.6 Error Metric for Optical Flow

Lastly, to establish whether an optical flow is accurate based on a given ground truth flow, a suitable metric is needed. For this, there exist different approaches, such as the angular error, bad pixel error, or endpoint error. In this thesis, only the endpoint error (EPE) is used to determine the quality of an optical flow as it is the most intuitive. The endpoint error is determined by the euclidean distance of each vector of the generated optical flow and the ground truth flow. With $(u, v)^T$ being a displacement vector of the generated optical flow, and $(u_{GT}, v_{GT})^T$ a displacement vector of the ground truth flow, the EPE can be calculated as follows:

$$\text{EPE} = \left\| \begin{pmatrix} u \\ v \end{pmatrix} - \begin{pmatrix} u_{GT} \\ v_{GT} \end{pmatrix} \right\| = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2}. \tag{2.2}$$

In order to obtain a singular value, we then average the EPE values of all vectors.

To give an initial idea of the results that RAFT and the ProFlow baseline produce, Figure 2.6 shows the optical flow for scene *Alley 1* (a) of the Sintel dataset [5], generated with RAFT (d) and the ProFlow baseline (c). In (b), the ground truth flow is also visualized for a better comparison in terms of accuracy. The EPE for the RAFT optical flow is 0.118, and for the ProFlow baseline the EPE is 0.127, thus RAFT has a better result. While the ProFlow baseline optical flow seems to capture small details a little better, as can be seen in the hair area, the outlines also tend to be less sharp as with RAFT. More details about these differences are provided in Section 4.1 of the experiment chapter.

Figure 2.6: Visualization of optical flows for scene *Alley 1* (a) of the Sintel benchmark. With the ground truth flow (b), optical flow generated with the ProFlow baseline with EPE=0.127 (c), and RAFT optical flow with EPE=0.118 (d). Image source for (a): [5].

3 Interframe Interpolation

In this chapter, we go into detail about every step needed in order to perform an interframe interpolation, as well as how to determine its resulting quality. Additionally, we also look at a few challenges that one might need to consider when performing an interpolation.

The idea of motion-compensated interframe interpolation is to take the optical flow and interpolate along the path of the flow field. A central concept of these interpolation methods is the so-called image warping, which is introduced in Section 3.1. Image warping describes the mapping of a position (x, y) in an image to a new position (x', y'), via an arbitrary mapping function. The color of each position (x, y) then gets put into the mapped position (x', y') which results in a warped image. To make use of such image warping, the optical flow is then used as a mapping function to warp either one or both input frames to a resulting interpolated frame. In the latter case, the warped frames are then combined to form one resulting frame. How this can be done, is explained in detail in Section 3.2, where the two interpolation methods, called forward warping and forward forward warping are introduced. These methods serve as a basis for the conducted experiments in Chapter 4. Besides these two methods, a brief summary of additional methods is also given in the same section. These additional methods are shortly examined in Chapter 4 as well. A problem that can occur during a mapping is when pixels get mapped to interpixel positions. To deal with this, the notion of splatting is introduced in Section 3.3. Here, different methods are introduced that describe which color gets splatted on neighboring pixels of an interpixel position. However, this can lead to cases where pixels get no color splatted to them. Such pixels are called holes. To deal with such pixels, a method of inpainting is introduced in Section 3.4, which deals with filling in such holes to obtain a complete image. In Section 3.5, we reiterate all needed steps of an interpolation to summarize the discussed concepts. Following this, a brief introduction to possible challenges that can occur in an interframe interpolation is given in Section 3.6. One big challenge is dealing with occluded and disoccluded areas. To determine such areas, a method called forward-backward consistency check can be used. How this method works is explained in Section 3.7. Wrapping up this chapter, three different error metrics for determining the quality of a distorted image are explained in Section 3.8. These metrics include the frequently used root mean square error (RMSE) and the structural similarity index measure (SSIM), as well as a relatively new metric, called video multimethod assessment fusion (VMAF).

3.1 Image Warping

The interframe interpolation methods that are used in the later shown experiments are all based on a central concept called image warping. What image warping is, how it works and how it can be combined with the optical flow to be used for an interpolation is introduced in this section.

3.1.1 Introduction to Image Warping

Image warping describes the transformation of a source image to a reshaped, *warped* image by mapping all positions (x, y) from the source image to new positions (x', y') in the destination image. This mapping is performed via an arbitrary mapping function *M*:

$$M : \mathbb{R}^2 \to \mathbb{R}^2, \quad M(x, y) \to (x', y'). \tag{3.1}$$

There are two different approaches to image warping. One entails using the source image I_{source} to map positions from this image to new positions. The information of the initial position is then transferred, or *forwarded*, into the mapped position. This forwarding of information creates a new, *warped* image I_{warped}. Such an approach is called *forward warping* and can be described as follows:

$$I_{warped}(M(x, y)) = I_{source}(x, y). \tag{3.2}$$

The $I_{source}(x, y)$ refers to the information contained in position (x, y) of the source image I_{source}. Analogously, $I_{warped}(M(x, y))$ describes the color at the mapped position $M(x, y)$ in the warped image I_{warped}. This concept is used in the later described interpolation methods, Section 3.2.

Besides warping the image in a forward way, one can realize image warping in an inverse way as well. For this, each position (x', y') in a warped image I_{warped} would be mapped to its corresponding position (x, y) in the original source image I_{source} via an inverse mapping function $M^{-1}(x, y)$. In each position of the warped image, the information is sampled from the corresponding position in the source image. Thus, the equation for such an inverse warping, also called *backward warping*, reads as follows:

$$I_{warped}(x, y) = I_{source}(M^{-1}(x, y)). \tag{3.3}$$

A forward warping can be seen in Figure 3.1 (a), where a position (x, y) gets mapped to a position (x', y') via a mapping function M, to form a warped image. A backward warping is shown in Figure 3.1 (b). Here, a position (x', y') in the warped image gets mapped to its corresponding position (x, y) in the source image via the inverse mapping function M^{-1}.

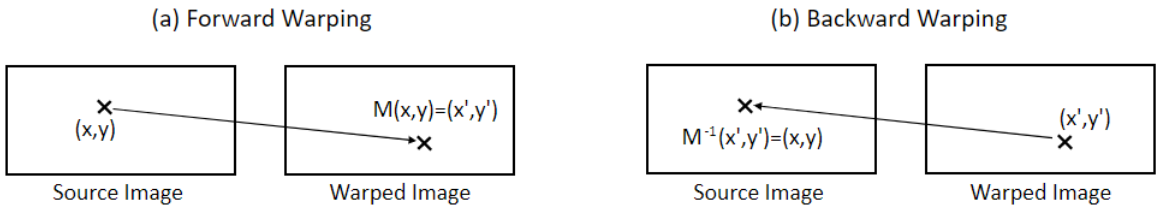


Figure 3.1: (a) Forward warping via mapping function M. (b) Backward warping via inverse mapping function M^{-1}.

While the backward warping variant is not the basis for the two interpolation methods shown in Section 3.2, it is worth mentioning since it is used for other possible interpolation methods that are briefly discussed in Subsection 3.2.3. Additionally, a backward warping is also used in the forward-backward consistency check, shown in Section 3.7.

Besides using image warping for interpolation methods, there have been many different applications of image warping throughout the years. The core idea of image warping was used as early as the late 1870s by F. Galton in his work "Composite Portraits, Made by Combining Those of Many Different Persons into a Single Resultant Figure" [47]. Galton used image warping to average images of different faces in order to form a 'facial prototype'. Deriving from this, other works emerged that used image warping to transform faces in an image to different genders or ages [48]. Furthermore, D. W. Thompson used the idea of image warping in his work [49], to visualize change in biological organisms. Among other things, his work consisted of altering skulls of primates into human skulls and showing how bones are related among different species. Another application of image warping can be for removing distortions in images caused by a camera and its positioning [50, 51], or constructing high-resolution images from low-resolution images [52]. Additionally, image warping can also be used to create visually efficacious 3D environments as described in this work [53].

Depending on the choice of the mapping function M, one can achieve different transformations. The most common being $scaling$, $rotation$, and $translation$, or a combination of either one of them. A rotation of an image turns the image around an axis. In scaling, the image gets scaled to a different size by a scaling factor. Here, the mapping function M is defined as a multiplication of x and y with a scaling factor. Either the same scaling factor is chosen for x and y, describing a uniform scaling, or different factors are chosen, for a non-uniform scaling. While these two are interesting concepts, the transformation that is important for

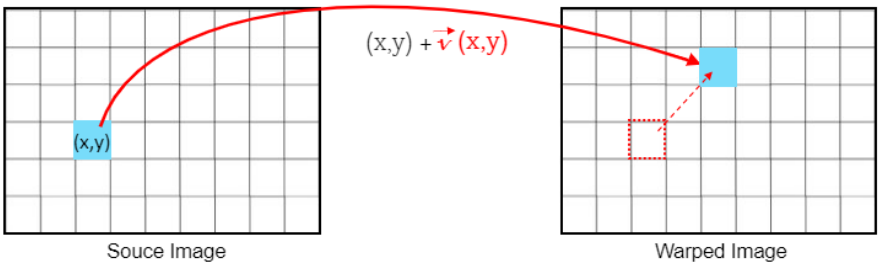


Figure 3.2: Forward warping with optical flow as the mapping function.

the interpolation methods is a translation. For a translation, the mapping function can be described as adding a position-dependent offset Δx and Δy to the x and y coordinate:

$$M(x, y) \rightarrow (x + \Delta x, y + \Delta y). \tag{3.4}$$

3.1.2 Combining Optical Flow with Image Warping

To make use of this image warping concept, one can use the previously defined optical flow as a mapping function M. For the interpolation we use a dense flow, which is defined for each pixel. This means that the domain of the mapping function now changes from continuous to discrete. Thus, Equation (3.1) gets adjusted with a discrete domain:

$$M : \mathbb{N}^2 \rightarrow \mathbb{R}^2. \tag{3.5}$$

M can now be defined as taking the displacement of each pixel from an image to its consecutive image and adding it to that pixel. The displacement for each pixel (x, y) is described as the displacement vector $\overrightarrow{v}(x, y)$, as described in Equation (2.1). The mapping function for a translation, shown in Equation (3.4), can now be adapted with the horizontal displacement u being the offset Δx, and the vertical displacement v replacing the offset Δy:

$$M(x, y) = (x + u, y + v). \tag{3.6}$$

Figure 3.2 shows such a forward warping with the optical flow. The color in the source image at pixel (x, y) gets forwarded to the position at $(x, y) + \overrightarrow{v}(x, y)$ in the warped image. Thus, Equation (3.2) gets updated with this new mapping function:

$$I_{warped}((x, y) + \overrightarrow{v}(x, y)) = I_{source}(x, y). \tag{3.7}$$

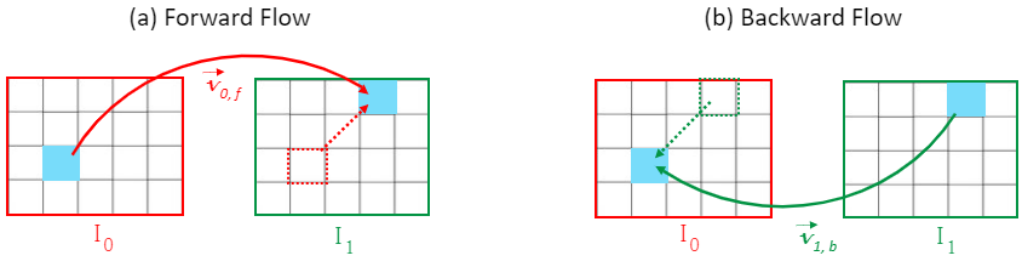


Figure 3.3: (a) Forward flow $\overrightarrow{v_{0,f}}$ describing the displacement from image I_0 to image I_1 and (b) backward flow $\overrightarrow{v_{1,b}}$ describing the displacement from image I_1 to image I_0.

How this concept can be used to interpolate a frame between two consecutive frames, is explained in the following section.

3.2 Interpolation Methods

In the previous section, the notion of image warping, and how it can be combined with the optical flow was already introduced. In this section, two specific interpolation methods that apply image warping based on optical flow are presented. These methods were introduced in the bachelor thesis of N. B. Senn [12], where they were compared to each other based on the resulting images. Additionally, other possible methods, also introduced in [12] are briefly discussed as well.

As illustrated in Figure 3.3, we define a given pair of images, image I_0 and its consecutive image I_1. For these images, a forward flow is defined as the displacement vector $\overrightarrow{v_{0,f}}$ for each pixel from image I_0 to image I_1, shown in (a). Additionally, a backward flow describes the displacement from image I_1 to image I_0 via the displacement vector $\overrightarrow{v_{1,b}}$ in each pixel, shown in (b). Our desired interpolated image at the time t is denoted as image I_t.

3.2.1 Forward Warping

The first interpolation method that is utilized later on is called *forward warping*, analogous to the forward warping concept described in Section 3.1. So far, this forward warping strategy warped the first image along the trajectory of the optical flow which went from image I_0 to image I_1, shown in Figure 3.4 (a). However, the goal of interpolation is to create an image I_t inbetween image I_0 and image I_1 at the temporal distance t. For this, the flow has to be adjusted, to go from image I_0 to the desired interpolated image I_t. Thus, the displacement

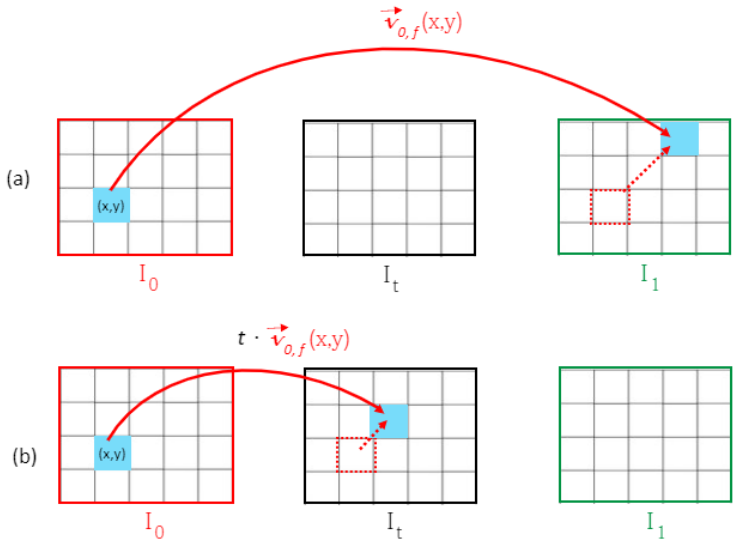


Figure 3.4: (a) Original forward flow $\overrightarrow{v_{0,f}}$ describing the displacement from image I_0 to image I_1. (b) Shortened forward flow $t \cdot \overrightarrow{v_{0,f}}$ describing the displacement from image I_0 to image I_t, showcasing a forward warping interpolation.

of the forward flow has to be shortened. This is achieved by multiplying the forward flow $\overrightarrow{v_{0,f}}$ with the distance t. Figure 3.4 (b) shows this adjustment of the original forward flow to describe a displacement from image I_0 to image I_t. The Equation (3.7), applied to a forward warping interpolation, now reads as follows:

$$I_t((x,y) + t \cdot \overrightarrow{v_{0,f}}(x,y)) = I_0(x,y). \tag{3.8}$$

This describes that the color at pixel (x,y) in image I_0 gets forwarded to the interpolated image I_t at the position $(x,y) + t \cdot \overrightarrow{v_{0,f}}(x,y)$.

Due to its simple nature, such an approach of generating interpolated images has been applied in other works as well, such as in the work of S. Niklaus and F. Liu [54]. In their work, an approach for dealing with the mapping of multiple pixels from the source image to the same pixel in the interpolated image is introduced. This described challenge and the different approaches to deal with it are also introduced in this thesis, in Section 3.3. Additionally, forward warping an image can also be used to obtain information about occlusions. Y. Wang *et al.* use forward warping in their work [41] to generate occlusion maps.

So far only image I_0 was used to perform a warping. To make use of additional information from image I_1, such as brightness changes, one can also use the backward flow $\overrightarrow{v_{1,b}}$ to

give forth this information to the interpolated image. The second interpolation method utilizes this bidirectional idea.

3.2.2 Forward Forward Warping

Analogous to its name, the method *forward forward warping*, uses two separate forward warpings, as described in Subsection 3.2.1, to generate two temporary interpolated images. These images then get combined to a resulting interpolation image at the end. This approach was introduced by N. B. Senn in his thesis [12] as well.

The first forward warping is done analogously to the one shown in Equation (3.8), in which image I_0 gets warped with the shortened forward flow $t \cdot \overrightarrow{v_{0,f}}$ to generate an interpolated image at temporal distance t. For this, we denote the resulting interpolated image as $I_{t,0}$. Applying Equation (3.8) to this forward warping case yields the following:

$$I_{t,0}((x_0, y_0) + t \cdot \overrightarrow{v_{0,f}}(x_0, y_0)) = I_0(x_0, y_0). \tag{3.9}$$

The second forward warping is performed by taking the second image, I_1, as a source and forward warping it with the backward flow $\overrightarrow{v_{1,b}}$. Since the temporal distance between image I_1 and the desired interpolated image is $1 - t$, the backward flow $\overrightarrow{v_{1,b}}$ has to be shortened to reach that distance. Here, we denote the interpolated image as image $I_{t,1}$. Applying Equation (3.8) to this second forward warping yields:

$$I_{t,1}((x_1, y_1) + (1 - t) \cdot \overrightarrow{v_{1,b}}(x_1, y_1)) = I_1(x_1, y_1). \tag{3.10}$$

Note that (x_0, y_0) stands for each pixel in the forward warping of image I_0 and (x_1, y_1) stands for each pixel in the forward warping of image I_1. These two temporary interpolated images $I_{t,0}$ and $I_{t,1}$ now have to be combined into one resulting interpolated image I_t. For this, the color at each pixel in I_t becomes a weighted sum of the color at that pixel in $I_{t,0}$ and $I_{t,1}$. The weight for the color in each pixel in $I_{t,0}$ is the distance $1 - t$, and the weight for the color in $I_{t,1}$ is t. This results in the following equation:

$$I_t(x, y) = (1 - t) \cdot I_{t,0}(x, y) + t \cdot I_{t,1}(x, y). \tag{3.11}$$

In Figure 3.5 such forward forward warping is illustrated. At the top, the two original flows, the (a) forward flow $\overrightarrow{v_{0,f}}$ and (b) backward flow $\overrightarrow{v_{1,b}}$ are displayed. The flows which are shortened with the respective distance in order to reach the interpolated image I_t, are shown at the bottom (c).

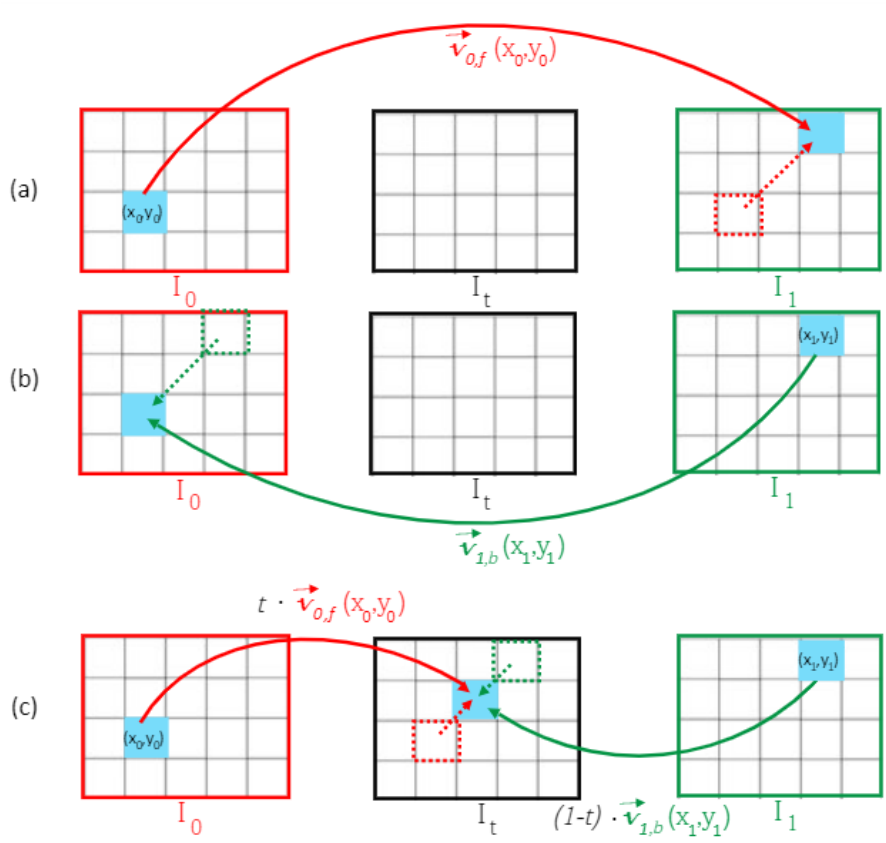


Figure 3.5: (a) Original forward flow $\overrightarrow{v_{0,f}}$ describing the displacement from image I_0 to image I_1. (b) Original backward flow $\overrightarrow{v_{1,b}}$ describing the displacement from image I_1 to image I_0 (c) Shortened forward and backward flow describing the displacement from I_0 to I_t and from I_1 to I_t, showcasing a forward forward warping.

3.2.3 Other Methods

In his thesis, N. B. Senn [12] introduces several other methods for generating interpolated images with an optical flow. Since most of our experiments were conducted with either a forward warping or forward forward warping method, these are the ones that we focus on. The other methods are used to give an overview of how each optical flow method performed with each method. For this, the other methods are introduced shortly in the following. However, we are not going into detail about them as we analyze them only briefly. Therefore, please refer to [12] for further details and illustration of these other methods.

Forward Warping with a backward flow: The already introduced forward warping method from Subsection 3.2.1 can be used analogously to the second forward warping performed in

the forward forward warping method, Equation (3.10). Meaning, a forward warping with source image I_1 and backward flow $\overrightarrow{v_{1,b}}$.

Backward Warping: Analogous to a forward warping method, the backward warping concept can be used as an interpolation method as well. This approach is called *backward warping*. First, a forward warping is applied on the forward flow $\overrightarrow{v_{0,f}}$ to create a new flow, where the starting position is the interpolated image I_t instead of image I_0. A flow warping is performed the same way as a normal image warping. Instead of forwarding a color to each pixel, the u and v coordinates of the displacement vector are forwarded. Since the warped flow has the same length as the original forward flow $\overrightarrow{v_{0,f}}$, it is reaching too far. Consequently, it has to be shortened to reach image I_1. With this warped and shortened flow, a backward warping is performed from image I_1.

Backward Warping with a Backward Flow: Optionally one could also perform a backward warping interpolation, with a warped backward flow $\overrightarrow{v_{0,f}}$ on image I_0.

Backward Backward Warping: Analogous to the forward forward warping method, [12] introduced a *backward backward warping* method as well. Here, the idea is to perform two separate backward warpings that get combined to a resulting interpolated image. For the two backward warpings, first, two flow forward warpings have to be performed on the forward flow $\overrightarrow{v_{0,f}}$ and backward flow $\overrightarrow{v_{1,b}}$. This is done analogously to the flow warping in the backward warping method. Then the warped flows get multiplied with the needed distance to reach image I_1, or I_0. In the end, a backward warping of image I_1 with the appropriately shortened and warped backward flow is performed, as well as a backward warping of image I_1 with its appropriately shortened and warped forward flow. The resulting interpolated image I_t is then described as a weighted sum of these two backward warpings, analogously to the forward forward warping.

Symmetric Forward Warping: A *symmetric forward warping* is similar to a forward forward warping, in that it performs two separate forward warpings that result in two temporary interpolated images that are combined to one resulting image at the end. Unlike a forward forward warping, however, a symmetric forward warping does not use the backward flow $\overrightarrow{v_{1,b}}$ in the second forward warping in Equation (3.10). Here, the forward flow $\overrightarrow{v_{0,f}}$ gets forward warped to have its start in image I_1, to imitate a forward flow $\overrightarrow{v_{1,f}}$ starting in I_1 and reaching a consecutive image I_2. For this warped flow to then reach the interpolated image I_t, it has to be shortened and mirrored. The rest of the method is performed the same as in a forward forward warping.

Symmetric Forward Warping with a Backward Flow: Optionally one could also use a warped backward flow $\overrightarrow{v_{1,b}}$ for a symmetric forward warping instead. Analogously, here, the warped backward flow $\overrightarrow{v_{1,b}}$ would be shortened and mirrored for the usage in the first forward warping, Equation (3.9). The rest would work the same as in a forward forward warping.

Symmetric Backward Warping: Similarly, a *symmetric backward warping* also performs two separate backward warpings, as is done in a backward backward warping. However, instead of two flow forward warpings, here, only the forward flow $\overrightarrow{v_{0,f}}$ gets forward warped. The backward warping with image I_0 thus has to now use a mirrored warped forward flow in order for the direction of the flow to reach the image I_0. The rest of the method also stays the same as in the backward backward warping approach.

Symmetric Backward Warping with a Backward Flow: Optionally, one could also use a warped backward flow $\overrightarrow{v_{1,b}}$ for a symmetric backward warping instead. In this case, the backward warping with image I_1 has to use a mirrored version of that warped backward flow. The two backward warpings work the same, except here, the warped flow has to be shortened differently for each backward warping.

Bidirectional Symmetric Backward Warping: A *bidirectional symmetric backward warping* performs 4 different backward warpings, two as in a symmetric backward warping method with warped forward flow $\overrightarrow{v_{0,f}}$ and the other two as in a symmetric backward warping method with warped backward flow $\overrightarrow{v_{1,b}}$. For this, forward warpings on both flows $\overrightarrow{v_{0,f}}$ and $\overrightarrow{v_{1,b}}$ are performed. The backward warpings are then averaged. The idea of such a bidirectional approach is to have a more consistent method by eliminating an unintentional greater usage of one flow. In case the forward flow or the backward flow is worse than the other, the results should not suffer.

Bidirectional Symmetric Forward Warping Lastly, a *bidirectional symmetric forward flow* also combines the two symmetric forward warpings approaches, one with a warped forward flow $\overrightarrow{v_{0,f}}$ and one with a warped backward flow $\overrightarrow{v_{1,b}}$. For this, the two flows are warped, and four separate temporary interpolated images are created in four forward warpings. In the end, these are also averaged to obtain a resulting image.

3.3 Splatting

As already briefly mentioned in Subsection 3.2.1, a problem that can occur in a forward warping is when multiple pixels in the source image I_{source} get mapped to one single pixel. How this can happen and what approaches can be used to deal with such a problem are introduced in this section. The approaches that are explained in the following were also discussed in [12].

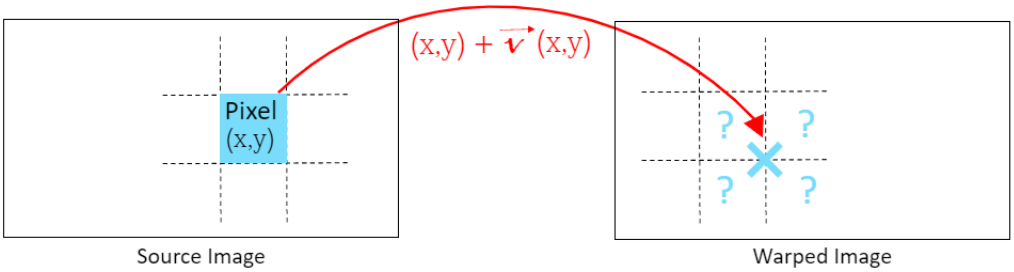


Figure 3.6: Splatting problem: pixel (x, y) gets mapped to an interpixel position $(x, y) + \overrightarrow{v}(x, y)$. On which neigboring pixel should the color be *splatted*?

3.3.1 Splatting Strategies

While using optical flow as a mapping function for a forward warping results in a discrete domain, the range still remains continuous, as seen in Equation (3.5). This can cause a pixel (x, y) in source image I_{source} to get mapped to an interpixel position. Meaning, position $(x, y) + \overrightarrow{v}(x, y)$ in the warped image I_{warped} can lie inbetween pixels. In this case, the question arises which neighboring pixel of $(x, y) + \overrightarrow{v}(x, y)$ gets the color of (x, y) *splatted* on them. This splatting problem is illustrated in Figure 3.6.

Each interpixel position can have a different amount of neighbors depending on its position in the image, with four being the maximum and most common amount of neighbors. For a position $(x', y') = (x, y) + \overrightarrow{v}(x, y)$ in a warped image I_{warped}, we define $N(x', y')$ as the set of all neighboring pixels of (x', y'). Further, we define $D(x', y') \subseteq N(x', y')$ as the set of neighboring pixels on which the color of (x, y) from I_{source} should be splatted on. For a mapped position $(x', y') = (x, y) + \overrightarrow{v}(x, y)$ in image I_{warped}, we define its four possible neighbors (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) as follows:

$$(x_1, y_1) = (\lfloor x' \rfloor, \lfloor y' \rfloor), \qquad (x_2, y_2) = (\lfloor x' \rfloor + 1, \lfloor y' \rfloor), \tag{3.12}$$

$$(x_3, y_3) = (\lfloor x' \rfloor, \lfloor y' \rfloor + 1), \qquad (x_4, y_4) = (\lfloor x' \rfloor + 1, \lfloor y' \rfloor + 1). \tag{3.13}$$

Figure 3.7 (a) shows where each neighboring pixel lies in relation to position (x', y'). Since an interpixel position is rarely right in the middle of each neighboring pixel, it has a different distance to its neighboring pixels. This is illustrated in Figure 3.7 (b), which also shows additional weights w_i for each neighboring pixel. This is relevant later on.

In the following, two strategies are presented that determine on which neighboring pixel

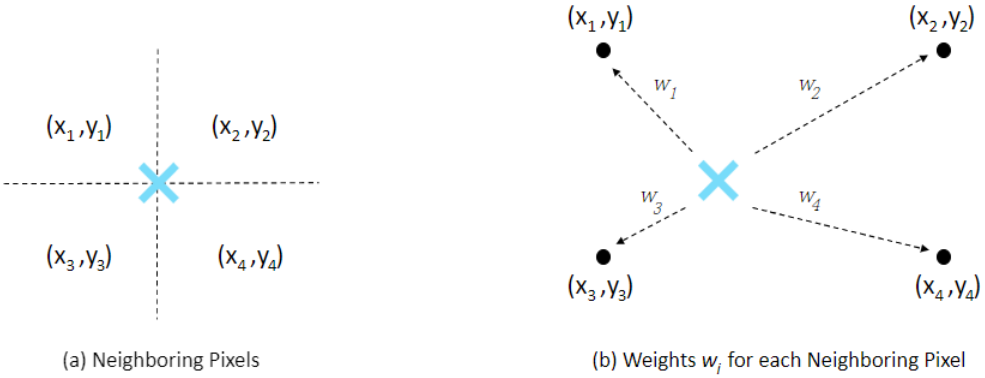


Figure 3.7: (a) Neighboring pixels (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) for an interpixel position, marked in blue. (b) Different distances with individual weight w_i for neighboring pixels.

should be splatted on: 1N and 4N.

1N / Nearest Neighbor
The first splatting strategy would be to splat only to one neighboring pixel (x_i, y_i). More specifically, the one with the shortest euclidean distance to the position (x', y'). In [12], this strategy is often referred to as *1N*, or *nearest neighbor*. Such a 1N strategy defines the set $D(x', y')$ as follows:

$$D(x', y') = \{(x_i, y_i) | \quad (x_i, y_i) \in N(x', y'), \sqrt{(x_i - x')^2 + (y_i - y')^2} \quad \text{is minimal}\}. \quad (3.14)$$

To give an example of how this strategy works in a concrete case, we consider Figure 3.7 (b). In this case pixel (x_3, y_3) would be colored according to the 1N strategy since this pixel has the shortest distance to the mapped position. However, this strategy can lead to many holes in the warped image. The concept of holes and how to avoid them is introduced in the next section. To reduce the number of holes to be filled in, a second strategy, which splats on multiple neighbors is better suited for this.

4N
The simplest strategy for reducing holes would be to simply splat on all neighboring pixels. This is referred to as a *4N*, for four neighbors. Although not all mapped interpixel positions have four neighbors, this is the easiest way to name the strategy. A 4N strategy would thus define the set $D(x', y')$ as follows:

$$D(x', y') = \{(x_i, y_i) | \quad (x_i, y_i) \in N(x', y')\}. \quad (3.15)$$

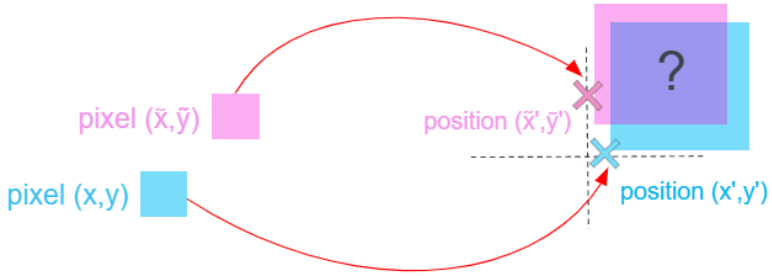


Figure 3.8: Two pixels (x, y) and $(\tilde{x}, \tilde{y})$ get mapped to interpixel positions (x', y') and $(\tilde{x}', \tilde{y}')$, which share a neighbor. This can lead to a collision if both pixels splat on their shared neighbor.

3.3.2 Collision Strategies for Splatting

As mentioned at the beginning of this section, a problem that can occur in forward warping is when multiple pixels get mapped to one single pixel. This can happen in case multiple pixels get mapped to an interpixel position which share a neighboring pixel. Depending on the splatting strategy both positions can now splat on this shared neighbor. This results in the neighbor pixel getting multiple colors splatted on them. Thus, a collision ensues that has to be dealt with.

To illustrate this with an example, we consider the case of two different pixels, (x, y) and $(\tilde{x}, \tilde{y})$. These pixels both get mapped to an interpixel position, as illustrated in Figure 3.8. Let position $(x', y') = (x, y) + \overrightarrow{v}(x, y)$ be the mapped position of pixel (x, y), and let position $(\tilde{x}', \tilde{y}') = (\tilde{x}, \tilde{y}) + \overrightarrow{v}(\tilde{x}, \tilde{y})$ be the mapped position of pixel $(\tilde{x}, \tilde{y})$. We define pixel (x_2, y_2) as the shared neighbor. A collision in pixel (x_2, y_2) can now occur in case both pixels (x, y) and $(\tilde{x}, \tilde{y})$ splat their color onto this shared neighbor. This can happen either when a 4N splatting strategy or 1N splatting strategy is used. For the 1N strategy the euclidean distance of position (x', y') to pixel (x_2, y_2) has to be the smallest out of $N(x', y')$, and the euclidean distance of $(\tilde{x}', \tilde{y}')$ to pixel (x_2, y_2) has to be the smallest out of $N(\tilde{x}', \tilde{y}')$ as well. To determine which color the pixel (x_2, y_2) should adopt, three different strategies are presented. These are called *uniform*, *weight*, and *weight winner*.

Each strategy defines individual weights w_i for each neighbor (x_i, y_i) of an interpixel position. The first two strategies, *uniform* and *weight*, then assign a color to a pixel with a collision according to Algorithm 3.1. Strategy *weight winner* assigns a color according to Algorithm 3.2. However, before introducing these algorithms, we first examine how each strategy defines the weights w_i.

Uniform
The first strategy, called *uniform*, aims to adopt the sum of all colors that get splatted on a pixel. Thus, the weights are uniformly distributed on all neighbors. For this, all weights w_i of neighboring pixels (x_i, y_i) of an interpixel position (x', y') equal 1:

$$w_i = 1, \quad \forall (x_i, y_i) \in D(x', y'). \tag{3.16}$$

Weight
The second strategy, called *weight*, uses a bilinear interpolation for defining the weights. The larger this distance the smaller the weight. It defines the weights w_i of neighboring pixels (x_i, y_i) based on their distance to the interpixel position (x', y'). Let (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) be the neighboring pixels of an interpixel position (x', y'). For $\epsilon_x = x' - \lfloor x' \rfloor$ and $\epsilon_y = y' - \lfloor y' \rfloor$, we define the weights w_i as follows:

$$w_1 = (1 - \epsilon_x) \cdot (1 - \epsilon_y), \qquad w_2 = \epsilon_x \cdot (1 - \epsilon_y), \tag{3.17}$$

$$w_3 = (1 - \epsilon_x) \cdot \epsilon_y, \qquad w_4 = \epsilon_x \cdot \epsilon_y. \tag{3.18}$$

Algorithm 3.1, describes how a warped image I_{warped} is generated by using either a uniform or weight strategy. As an input the algorithm gets the source image I_{source} and the optical flow $\overrightarrow{v}$, which describes the displacement of I_{source} to the destination image I_{warped}. Note that in this pseudo code algorithm the images are represented as only two-dimensional arrays. In practice, the arrays representing the images would have to be three-dimensional in order to store the RGB value. However, for simplicity reasons we define a color as just a one-dimensional value in this algorithm. Meaning for example the color in pixel (x, y) in image I_{source} would correspond to $I_{source}[x][y]$. The algorithm then generates a warped image I_{warped} by iterating through all pixels in the source image I_{source} and mapping them to a new position with the optical flow as a mapping function. For each of these positions, the set D is then determined according to either a 1N or 4N strategy. Further, each position defines their individual weights w_i for each of the neighboring pixels contained in D. The color of each pixel (x_i, y_i) in I_{warped} is then calculated as the sum of all colors that get splatted on this pixel and multiplied with the respective weight w_i specified for (x_i, y_i). These weights are all stored in the array W. This sum then gets divided by the sum of all weights specified for pixel (x_i, y_i). In case the sum of weights specified for a pixel (x_i, y_i) equals zero, then $I_{warped}(x_i, y_i)$ gets no color assigned and is thus a hole that has to be inpainted.

Weight Winner
The third strategy is called *weight winner* and it defines the weights w_i the same as the strategy *weight* does, shown in Equations (3.17), (3.18). The difference here is that the color in a pixel (x_i, y_i) in image I_{warped} is not described as a weighted and normalized sum of all colors that get splatted on it. In a weight winner strategy, the pixel (x_i, y_i) gets the color of only one

Algorithm 3.1 Algorithm for Generating a Warped Image with *Uniform* or *Weight* Splatting

procedure GENERATEWARPEDIMAGE($I_{source}, \overrightarrow{v}$)
  initialize empty array I_{warped}
  initialize empty array W // An array to save the sum of weights w_i for pixel (x_i, y_i)
  for all (x, y) in I_{source} **do**
    $(x', y') \leftarrow (x, y) + \overrightarrow{v}(x, y)$
    determine $D(x', y')$ // either 1N or 4N
    for all $(x_i, y_i) \in D(x', y')$ **do**
      calculate($w_i, (x', y')$) // calculate weight w_i of pixel (x_i, y_i) for position (x', y') with either uniform or weight strategy
      $I_{warped}[x_i][y_i] \leftarrow$I$_{warped}[x_i][y_i] + w_i \cdot I_{source}[x][y]$
      $W[x_i][y_i] \leftarrow W[x_i][y_i] + w_i$
    end for
  end for
  for all (x_i, y_i) in I_{warped} **do**
    if $W[x_i][y_i] > 0$ **then**
      $I_{warped}[x_i][y_i] \leftarrow I_{warped}[x_i][y_i]/W[x_i][y_i]$
    else
      $I_{warped}[x_i][y_i]$ is a hole
    end if
  end for
  return I_{warped}
end procedure

pixel, namely the pixel that defines the largest weight w_i. Thus, the winner of all weights gets chosen. Algorithm 3.2 describes how an image I_{warped} gets created with the usage of a weight winner strategy. The red part showcases what changed to the previous algorithm, Algorithm 3.1. Analogous to the previous algorithm, every pixel in the source image I_{source} is mapped to a new position and the set D is defined for each position corresponding to a 1N or 4N strategy. Each pixel in I_{warped} then gets the color of the pixel that is a *weight winner* assigned to it. Analogous to the uniform and weight strategy, if the sum of all weights for a pixel equals zero then I_{warped} has a hole in this pixel.

To illustrate the introduced strategies, let us apply each collision strategy to the example in Figure 3.8. For this, let us assume that both pixels splat on their shared neighboring pixel, (x_2, y_2). Further let w_2 be the weight that position (x', y') defines for (x_2, y_2), and let $\tilde{w}_2$ be the weight that position $(\tilde{x}', \tilde{y}')$ defines for (x_2, y_2).

Algorithm 3.2 Algorithm for Generating a Warped Image with *Weight Winner* Splatting

procedure GENERATEWARPEDIMAGEWEIGHTWINNER($I_{source}, \overrightarrow{v}$)
  initialize empty array I_{warped}
  initialize empty array W // An array to save the sum of weights w_i for pixel (x_i, y_i)
  for all (x, y) in I_{source} **do**
    $(x', y') \leftarrow (x, y) + \overrightarrow{v}(x, y)$
    determine($D(x', y')$) // either 1N or 4N
    for all $(x_i, y_i) \in D(x', y')$ **do**
      calculate($w_i, (x', y')$) // calculate weight w_i of pixel (x_i, y_i) for position (x', y') with weight winner strategy

      if $w_i > W[x_i][y_i]$ **then**
        $I_{warped}[x_i][y_i] \leftarrow I_{source}[x][y]$
        $W[x_i][y_i] \leftarrow w_i$
      end if
    end for
  end for
  for all (x_i, y_i) in I_{warped} **do**
    if $W[x_i][y_i] == 0$ **then**
      $I_{warped}[x_i][y_i]$ is a hole
    end if
  end for
  return I_{warped}
end procedure

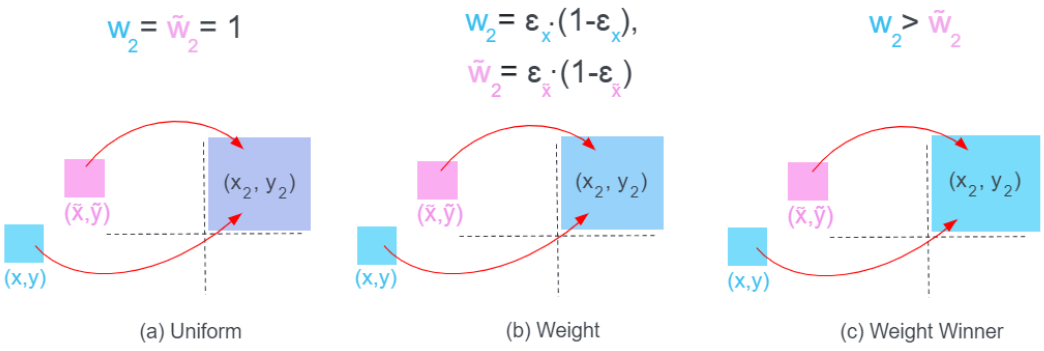


Figure 3.9: Strategies (a) *uniform*, (b) *weight* and (c) *weight winner* applied to the example collision of Figure 3.8.

In case of a *uniform* collision strategy, all weights would equal one, $w_2 = \tilde{w}_2 = 1$ and thus the color in pixel (x_2, y_2) in the warped image would be calculated as follows:

$$I_{warped}(x_2, y_2) = \frac{I_{source}(x, y) + I_{source}(\tilde{x}, \tilde{y})}{2}$$

Figure 3.9 (a) shows what color the pixel (x_2, y_2) would adopt according to a uniform strategy. Since both weights equal 1, the RGB values of Pixel (x, y) and $(\tilde{x}, \tilde{y})$ would be added and divided by 2.

For a *weight* strategy, w_2 and $\tilde{w}_2$ would be calculated according to Equation (3.17). Pixel (x_2, y_2) in I_{warped} would then be calculated as follows:

$$I_{warped}(x_2, y_2) = \frac{w_2 \cdot I_{source}(x, y) + \tilde{w}_2 \cdot I_{source}(\tilde{x}, \tilde{y})}{w_2 + \tilde{w}_2}$$

Figure 3.9 (b) shows how an example of a weight strategy applied to the example of Figure 3.8 could look like. In this example, the w_2 is bigger than $\tilde{w}_2$, hence why the blue color of pixel (x, y) is more prominent in the resulting color.

For the last strategy, *weight winner*, the color at I_{warped} equals:

$$I_{warped}(x_2, y_2) = \begin{cases} I_{source}(x, y), & if \quad w_2 > \tilde{w}_2 \\ I_{source}(\tilde{x}, \tilde{y}), & if \quad w_2 < \tilde{w}_2 \end{cases}$$

In case both weights are the same, then the color of the pixel which is the first in the iteration is chosen. Figure 3.9 (c) shows how a weight winner strategy applied to the example of Figure 3.8 would look like in case $w_2 > \tilde{w}_2$. Since w_2 is larger, the color in pixel (x, y) gets splatted on pixel (x_2, y_2).

From these two different splatting strategies and three different collision strategies, one can combine each splatting and collision strategy, which results in six different cases. Each of these combinations was used for the conducted experiments in Chapter 4.

Analogous to the splatting problem, backward warping can have a similar problem. Here, a problem can occur when a pixel in the warped image gets mapped to an interpixel position in the source image. In this case, the question is, from which neighboring pixels the color should be sampled. For this, there are strategies that work similarly to the ones introduced for splatting. Here, individual weights w_i for each neighboring pixel are defined as well. One can sample from either one neighbor (1N), namely the nearest, or from all neighbors (4N). For the latter, one can either take the average of all neighboring colors or use the weights w_i for a normalized and weighted sum of the neighboring colors. In this case, the weights w_i are defined analogously to Equations (3.17), (3.18). For the later conducted experiments, only

the sampling strategy called *4N weight* was used. However, since this thesis focuses on the interpolation methods forward warping and forward forward warping, the concept of sampling is not explained in full detail. For a more elaborate explanation, please refer to [12].

3.4 Inpainting

In the last section it was already mentioned that in case no color is splatted on a pixel in a warped image, this pixel then remains a hole that has to be filled with a color. Filling such holes is referred to as *inpainting*. There are many approaches to inpaint empty regions in an image, ranging from a synthesis based approach [55, 56] of copying existing pixels into holes, or approaches where the color from neighboring pixels is propagated, or *diffused* into the holes [57, 58]. One example of the latter diffusion based approach is explained in this section. Diffusion based inpainting methods are based on partial differential equations (PDEs) to compute an inpainted image u. Typically, the Laplace equation is chosen for this [59]. Additionally, different boundary conditions are defined, usually in the form of Dirichlet and Neumann boundary conditions. Since images are of a discrete form, the underlying PDE then has to be discretized. For this, an approximation is defined for the PDE, which is then solved by an iterative method such as the Gauß-Seidel method. In the following, an isotropic diffusion based inpainting approach is explained, which works analogously to the reduced formulation introduced by M. Mainberger *et al.* in their work [60].

For a continuous image f with domain Ω, we define Ω_K as the pixels of the image f that contain a color, and $\Omega \setminus \Omega_K = \Omega_E$ as empty pixels of the image. The goal is to now fill each pixel in domain Ω_E with a color to obtain a resulting inpainted image u. Such an inpainted image u is computed by solving the Laplace equation:

$$\Delta u(x, y) = 0, \quad \forall (x, y) \in \Omega_E. \tag{3.19}$$

Here, Δ denotes the Laplace operator. To ensure that a diffusion is only applied to regions that do not contain any color, the following Dirichlet boundary condition is defined:

$$u(x, y) = f(x, y), \quad \forall (x, y) \in \Omega_K. \tag{3.20}$$

Lastly, for $\partial\Omega$ being the image boundary, a Neumann boundary condition on the derivative in outer normal direction $\partial_n u$ is defined:

$$\partial_n u(x, y) = \overrightarrow{n} \cdot \nabla u(x, y) = 0, \quad \forall (x, y) \in \partial\Omega. \tag{3.21}$$

Here, $\nabla u(x, y)$ represents the gradient vector of $u(x, y)$.

As already mentioned, the PDE has to be discretized . For each pixel $(i,j) \in \Omega_E$ the following approximation for Equation (3.19) is defined as follows:

$$0 = \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} \frac{u(\hat{\imath},\hat{\jmath}) - u(i,j)}{h^2} \quad + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} \frac{f(\hat{\imath},\hat{\jmath}) - u(i,j)}{h^2}. \tag{3.22}$$

Here, $h = (h_x, h_y)^T$ is the grid spacing in x and y direction. Furthermore, N^{Ω_E} stands for all neighbors that are holes, and N^{Ω_K} describes all neighbors that already have a color. Depending on the position, a pixel can have either one neighbor in case of a corner, three neighbors in case of an edge, or four in the rest of the cases. Since this equation holds for every pixel $(i,j) \in \Omega_E$, it is apparent that there are $|\Omega_E|$ many of these equation. By assuming $h = 1$, Equation (3.22) now yields:

$$\begin{aligned} 0 &= \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) - u(i,j) + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}) - u(i,j) \\ &= \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) - \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(i,j) + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}) - \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} u(i,j). \end{aligned} \tag{3.23}$$

This can now be simplified as follows:

$$\sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(i,j) + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} u(i,j) = \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) \quad + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}) \tag{3.24}$$

$$\sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega}(i,j)} u(i,j) = \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) \quad + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}) \tag{3.25}$$

$$|N^{\Omega}(i,j)| \cdot u(i,j) = \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) \quad + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}) \tag{3.26}$$

$$u(i,j) = \frac{1}{|N^{\Omega}(i,j)|} \cdot \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath}) \quad + \sum_{(\hat{\imath},\hat{\jmath}) \in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}). \tag{3.27}$$

First we move the sums over $u(i,j)$ to the left hand side of the Equation (3.23). This results in Equation (3.24). Adding all neighbors with a color and all neighbors that are holes of a

pixel corresponds to the sum of all neighbors of this pixel. Meaning the union of $N^{\Omega_K}(i,j)$ and $N^{\Omega_E}(i,j)$ corresponds to the set of all neighbors, $N^{\Omega}(i,j)$. This is done in Equation (3.25). Furthermore, we shorten the sum over all elements in $N^{\Omega}(i,j)$ to multiplying $u(i,j)$ with $|N^{\Omega}(i,j)|$ in Equation (3.26). Lastly, we divide the right hand side with $|N^{\Omega}(i,j)|$ in Equation (3.27) and we obtain an approximation for pixel (i,j).

Applying a Gauß-Seidel method to Equation (3.27) results in the following equation that can be iteratively solved:

$$u(i,j)_{GS}{}^{k+1} = \frac{1}{|N^{\Omega}(i,j)|} \cdot \sum_{(\hat{\imath},\hat{\jmath})\in N_{-}{}^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath})^{k+1} + \sum_{(\hat{\imath},\hat{\jmath})\in N_{+}{}^{\Omega_E}(i,j)} u(\hat{\imath},\hat{\jmath})^{k} + \sum_{(\hat{\imath},\hat{\jmath})\in N^{\Omega_K}(i,j)} f(\hat{\imath},\hat{\jmath}). \tag{3.28}$$

Here, $N_{-}{}^{\Omega_E}$ denotes the neighbors whose values have already been calculated in the current iteration, and $N_{+}{}^{\Omega_E}$ denotes the neighbors whose values still have to be computed in the current iteration.

To accelerate the computation, a successive over-relaxation (SOR) is applied on top of the Gauß-Seidel solver. With a predefined relaxation parameter $\omega = 1.95$ the following equation holds for applying a SOR:

$$u(i,j)^{(k+1)} = u(i,j)^{k} \cdot 1.95 \cdot (u(i,j)_{GS}{}^{(k+1)} - u(i,j)^{k}). \tag{3.29}$$

As previously investigated in [12], a valid result is already achieved with 250 iterations.

3.5 All Steps of a Complete Interpolation Procedure

To summarize what was introduced in terms of creating an interpolated image, we first apply one of the interpolation methods introduced in Section 3.2 which is based on image warping, shown in Section 3.1. To deal with a conflicting mapping in a pixel we apply a splatting strategy as described in Section 3.3. For this, one of the six possible strategies (1N uniform, 1N weight, 1N weight winner, 4N uniform, 4N weight, 4N weight winner) is chosen. Since such splatting can leave holes in the image, a final inpainting is applied, as described in Section 3.4. Algorithm 3.3 showcases the steps of a complete forward warping interpolation method. First, a normal forward warping is performed on image I_0 with forward flow $t \cdot \overrightarrow{v_{0,f}}$ with a chosen splatting strategy. This results in an interpolated image containing holes, which are then inpainted in the last step, yielding a resulting image I_t.

With image I_0, image I_1, forward flow $\overrightarrow{v_{0,f}}$ and backward flow $\overrightarrow{v_{1,b}}$ as an input, the steps of a complete forward forward warping method are shown in Algorithm 3.4. First, a forward warping is performed on image I_0 with forward flow $t \cdot \overrightarrow{v_{0,f}}$ and a chosen splatting strategy.

Algorithm 3.3 Complete Forward Warping Interpolation Method

Input: image I_0, forward flow $\overrightarrow{v_{0,f}}$

forward warping of image I_0 with forward flow $t \cdot \overrightarrow{v_{0,f}}$ and a chosen splatting strategy

inpainting of holes

return interpolated image I_t

This results in a temporary interpolated image $I_{0,t}$. Then a forward warping on image I_1 with backward flow $(1-t) \cdot \overrightarrow{v_{1,b}}$ and a chosen splatting strategy is performed, resulting in image $I_{1,t}$. Afterwards the two images $I_{0,t}$ and $I_{1,t}$ are combined as a weighted sum into the resulting image I_t. This image still contains holes, which get inpainted in a last step, yielding an inpainted image I_t.

Algorithm 3.4 Complete Forward Forward Warping Interpolation Method

Input: image I_0, image I_1, forward flow $\overrightarrow{v_{0,f}}$, backward flow $\overrightarrow{v_{1,b}}$

forward warping of image I_0 with forward flow $t \cdot \overrightarrow{v_{0,f}}$ and a chosen splatting strategy, resulting in temporary image $I_{0,t}$

forward warping of image I_1 with backward flow $(1-t) \cdot \overrightarrow{v_{1,b}}$ and a chosen splatting strategy, resulting in temporary image $I_{1,t}$

combining $I_{0,t}$ and $I_{1,t}$

inpainting of holes

return interpolated image I_t

3.6 Challenges to Interframe Interpolation

While interframe interpolation provides promising results for small motions, many factors can decrease the quality of the interpolated image. Most interpolation methods assume a linear motion between frames. However, objects in the real world typically don't move in a linear way. In case of a big temporal difference or large motion between two consecutive frames, this motion cannot be estimated linearly. Figure 3.10 showcases this problem, where a ball is

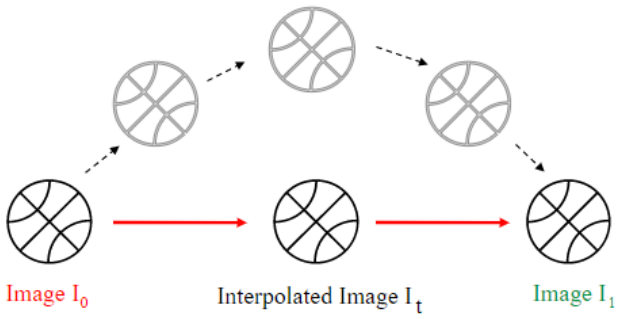


Figure 3.10: Non-linear motion depicted in interpolation results.

being thrown in a non-linear motion. Image I_0 captures the ball at the beginning of the throw and image I_1 captures the end of the throw. In between these images, the ball moves up in a parabola motion. However, since most interframe interpolations assume a linear motion, the interpolated image I_t does not capture the correct position of the throw in-between the images. To correctly depict the motion of the ball in the interpolated image, the images would need to be captured closer together in time, or a non-linear interpolation method [61] is needed. While such non-linear interpolation methods are an interesting topic by themselves, they are outside the scope of this thesis. Additionally, occluded and disoccluded areas can be a challenge as well. This problem was already discussed in Section 2.3. In the following section, an approach for detecting occluded and disoccluded areas called forward-backward consistency check is given.

3.7 Forward-Backward Consistency Check

Determining which objects or parts of objects disappear in the consecutive frame can be beneficial in the process of interpolation. This information can be represented in terms of occlusion maps. Occlusion maps give information about each pixel whether it is occluded in the next frame or not. To obtain such occlusion maps an algorithm called forward-backward consistency check can be used. The forward-backward consistency check is based on a forward-backward consistency assumption [62]. The idea is that if we follow the path of a forward flow and then follow back the path of the backward flow we should arrive at the same position. Figure 3.11 showcases this idea. Here, we first follow the trajectory of the forward flow, denoted as $\overrightarrow{v_0}$, from a starting position in image I_0. This leads to a new position in image I_1. From this new position, we then go back with the backward flow, here denoted as $\overrightarrow{v_1}$, to see if the starting position in I_0 is reached again. In case the starting position is not reached again, the

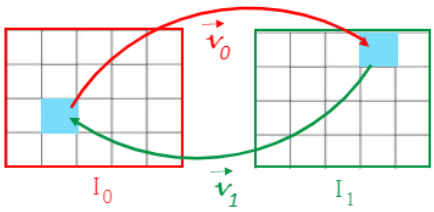


Figure 3.11: Idea of a forward-backward consistency check. First, we follow the path of the forward flow $\overrightarrow{v_0}$ and then go back with backward flow $\overrightarrow{v_1}$ to see whether we reach the same starting position in I_0.

flows are not consistent and thus an occlusion can be present. Meaning that the object at the aforementioned starting position might be occluded in image I_1. For a disoccluded area, the idea is the same, except here, we use starting points in image I_1. Hence, we first follow the path of the backward flow $\overrightarrow{v_1}$ and then use the forward flow $\overrightarrow{v_0}$ to reach image I_1 again. Here, an inconsistency means that objects in the starting position become visible again, or disoccluded in I_1.

Algorithm 3.5 describes the implementation of such a forward-backward consistency check. For this, we define the forward flow $\overrightarrow{v_0}$ for a pixel (i, j), ranging from image I_0 to image I_1 as the following:

$$\overrightarrow{v_0}(i,j) = \begin{pmatrix} u_{0_{i,j}} \\ v_{0_{i,j}} \end{pmatrix}$$

Analogously, the backward flow $\overrightarrow{v_1}$ for pixel (i, j), ranging from image I_1 to image I_0 is denoted as follows:

$$\overrightarrow{v_1}(i,j) = \begin{pmatrix} u_{1_{i,j}} \\ v_{1_{i,j}} \end{pmatrix}$$

In the first step, a backward warping is performed on the backward flow $\overrightarrow{v_1}$ with the forward flow $\overrightarrow{v_0}$. This results in a new warped flow $\overrightarrow{v_w}$, which is defined for a pixel (i, j) as follows:

$$\overrightarrow{v_w}(i,j) = \begin{pmatrix} u_{w_{i,j}} \\ v_{w_{i,j}} \end{pmatrix}$$

For each pixel in image I_0 we then add the squared sum of the horizontal components to the squared sum of the vertical components of the flow $\overrightarrow{v_0}$ and $\overrightarrow{v_w}$. In the end, we square root this sum. The result for each pixel (i, j) is then stored in the array $Result$. Since an optical flow estimation is rarely accurate, we allow a small deviation. For this, we define a

threshold T. An inconsistency is then seen only if this threshold is exceeded. In a resulting array $OcclusionMap$ we then store for each pixel whether the forward-backward check result exceeded this threshold or not. If the threshold is exceeded we define it as an inconsistency. Typically, to visualize such an occlusion map, the inconsistent pixels are assigned a white color and the consistent pixels are colored black. An occlusion map for the scene *Temple 2* of the Sintel benchmark [5] can be seen in Figure 3.12 (d). In the scene, the wing of the dragon moves up in the second frame, Figure 3.12 (c), thus this area is occluded in the second frame. This corresponding area is thus marked white in the occlusion map.

Algorithm 3.5 Forward-Backward Consistency Check Generating Occlusion Maps

Input: forward flow $\overrightarrow{v_0}$, backward flow $\overrightarrow{v_1}$, threshold T, image I_0

initialize empty array $Results$ // stores for each pixel the result value of the forward-backward check

initialize array $OcclusionMap$ // stores for each pixel if we have a consistency

$\overrightarrow{v_w}(x,y) = \overrightarrow{v_1}((x,y) + \overrightarrow{v_0}(x,y))$ // backward warping of backward flow $\overrightarrow{v_1}$ with the forward flow $\overrightarrow{v_0}$, resulting in a new warped flow $\overrightarrow{v_w}$

for all pixel (i,y) in image I_0 **do**
$Results[i][j] = \sqrt{(u_{0_{i,j}} + u_{w_{i,j}})^2 + (v_{0_{i,j}} + v_{w_{i,j}})^2}$

if $Results[i][j] > T$ **then**
$OcclusionMap[i][j] =$ inconsistent

else
$OcclusionMap[i][j] =$ consistent

end if

end for

return $OcclusionMap$

Analogously, we can also generate a disocclusion map with Algorithm 3.5. For this, we would use image I_1 as a starting point. The calculation would be the same, except here, we would swap the forward and backward flow. Meaning the forward flow $\overrightarrow{v_0}$ would be backward warped with the backward flow $\overrightarrow{v_1}$ instead. Algorithm 3.6 shows this modified version of Algorithm 3.5.

The resulting disocclusion map can be seen in Figure 3.12 (b). Here, part of the wing in the first frame, shown in Figure 3.12 (a), is occluding an area that becomes visible again in the second frame, shown in 3.12 (c). Thus, this area is marked white.

Algorithm 3.6 Modified Forward-Backward Consistency Check Generating Disocclusion Maps

Input: forward flow $\overrightarrow{v_0}$, backward flow $\overrightarrow{v_1}$, threshold T, image I_1

initialize empty array $Results$ // stores for each pixel the result value of the forward-backward check

initialize array $DisocclusionMap$ // stores for each pixel if we have a consistency

$\overrightarrow{v_w}(x,y) = \overrightarrow{v_0}((x,y) + \overrightarrow{v_1}(x,y))$ // backward warping of forward flow $\overrightarrow{v_0}$ with the backward flow $\overrightarrow{v_1}$, resulting in a new warped flow $\overrightarrow{v_w}$

for all pixel (i,y) in image I_1 **do**
$Results[i][j] = \sqrt{(u_{1_{i,j}} + u_{w_{i,j}})^2 + (v_{1_{i,j}} + v_{w_{i,j}})^2}$

if $Results[i][j] > T$ **then**
$DisocclusionMap[i][j] =$ inconsistent

else
$DisocclusionMap\ [i][j] =$ consistent

end if

end for

return $DisocclusionMap$

Such forward-backward consistency check was used to generate occlusion and disocclusion maps which are relevant for the experiments in Chapter 4.

3.8 Error Metrics

In order to be able to compare the results of interpolation methods, one has to be able to determine the quality of an interpolated image with the help of an appropriate metric. For this, three different metrics are introduced in this section.

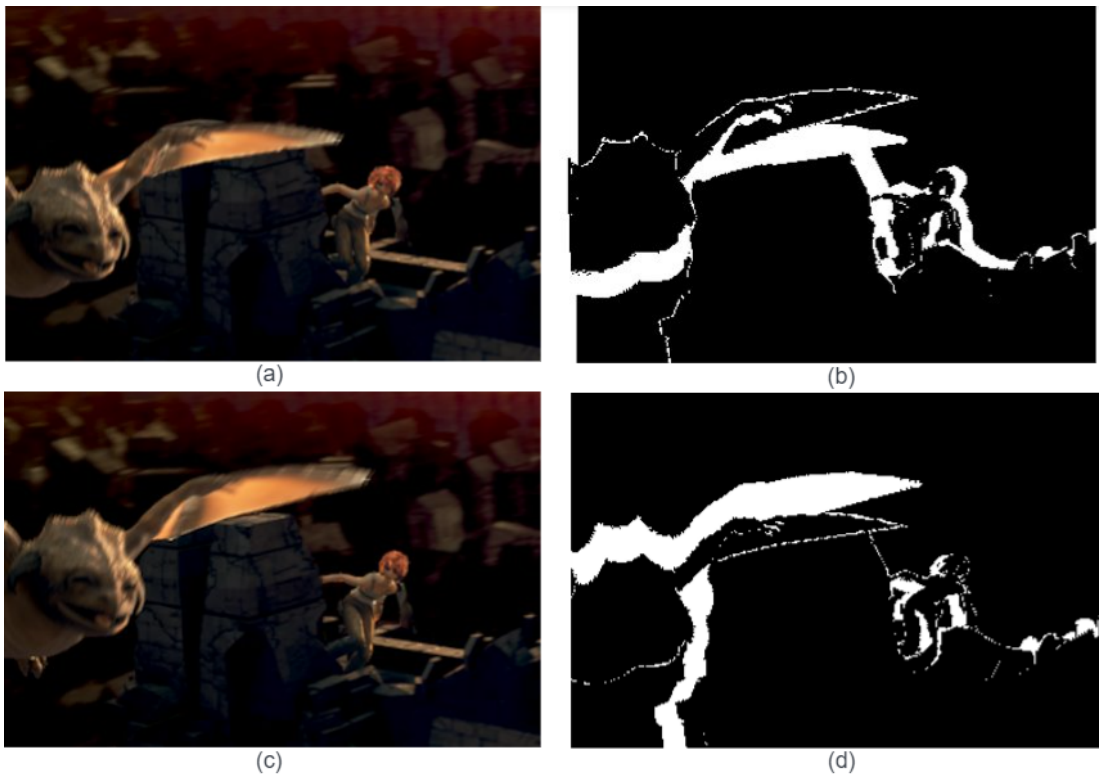


Figure 3.12: First frame I_0 (a) and second frame I_1 (c) of the *Temple 2* scene of the Sintel benchmark [5], and the occlusion map (d) and disocclusion map (b) for this scene.

3.8.1 RMSE

The first metric is called the root mean square error (RMSE). It is the squared version of the frequently used metric mean square error (MSE) and belongs to the category of full-reference image quality metrics (IQM). Full-reference metrics measure a distorted image based on an original undistorted image [63]. As the name suggests, the RMSE measures the square root of the mean of a squared per-pixel color difference between the distorted and reference image. For a distorted image I with height H and width W, and its ground truth reference image I_{GT}, the RMSE can be calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{WH} \cdot \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} (I(x,y) - I_{GT}(x,y))^2}. \tag{3.30}$$

While this metric is simple and straightforward, it is not always a preferred metric. In many cases, the RMSE fails to fully reflect the perceived quality of an image, as it only measures the

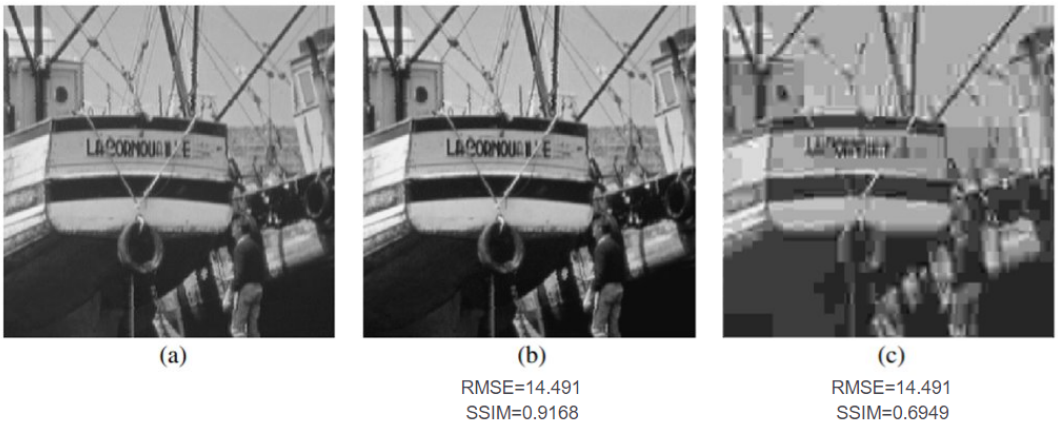


Figure 3.13: Two differently distorted images with the same RMSE and different SSIM. Original undistorted image (a), distorted image with $RMSE = 14.491$, $SSIM = 0.9168$ (b), distorted image with $RMSE = 14.491$, $SSIM = 0.6949$ (c). Image source: [64].

absolute error. Figure 3.13 shows an example of two differently distorted images that have the same RMSE. Even though both distorted images have the same quality according to the RMSE, comparing both distorted images to the original image (a), it is apparent that image (b) is visually much better than the image (c). For cases like these, a different metric is needed to better capture human perception in image quality assessment. The second metric aims to achieve this.

3.8.2 SSIM

The second metric, structural similarity index measure (SSIM), was introduced by Z. Wang *et al.* [64] As already mentioned, it aims to solve the problem of not capturing perceived quality, as is the case with metrics such as the RMSE.

The human visual system is very complex and errors in images are not always perceived the same way. For example artifacts in high contrast regions or areas with more texture are much less visible to the human eye than artifacts in a low contrast environment. This phenomenon is called contrast masking. Another phenomenon is luminance masking, where artifacts are less visible in brighter areas. Thus, the quality of an image depends on different characteristics of the underlying errors. To reflect these phenomena, a metric has to consider these different aspects. The SSIM aims to do just that by viewing the distortion of an image as a perceived change in structural information with the consideration of luminance and contrast masking. The structure of objects in a scene is independent of luminance and contrast,

hence why the SSIM measures the structure, luminance, and contrast separately. Similar to the RMSE, SSIM is also a full-reference metric. In contrast to the RMSE however, here, a higher SSIM value indicates better quality. SSIM values range from -1 to 1. Additionally, the SSIM does not consider a pixel-wise difference between a distorted and reference image, but rather a difference of multiple patches. The underlying idea is that spatially close pixels have interdependencies. For a patch x in the distorted image I and the corresponding patch y in the ground truth image I_{GT}, the luminance, structure and contrast are defined as follows:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{{\mu_x}^2{\mu_y}^2 + C_1} \tag{3.31}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{{\sigma_x}^2{\sigma_y}^2 + C_2} \tag{3.32}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \tag{3.33}$$

Equation (3.31) describes the luminance comparison $l(x,y)$, Equation (3.32) the contrast comparison $c(x,y)$, and Equation (3.33) the structure comparison $s(x,y)$ of a distorted image I and its ground truth image I_{GT}. Furthermore, μ_x describes the average of x and μ_y the average of y, σ_x is the standard deviation of x, σ_y the standard deviation of y, σ_x^2 is the variance of x, σ_y^2 is the variance of y, and σ_{xy} the covariance. Constants C_1, C_2 and C_3 are added for stability and can be calculated according to Equation (3.32). Here, L stands for the dynamic range of pixel values, and $K_1 \ll 1$ and $K_2 \ll 1$ are small constants.

$$C_1 = (K_1L)^2, \quad C_2 = (K_2L)^2, \quad C_3 = \frac{C_2}{2}. \tag{3.34}$$

From the luminance, contrast and structure difference, the SSIM of (x,y) is then determined as follows:

$$SSIM(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma. \tag{3.35}$$

Usually, $\alpha = \beta = \gamma = 1$ is set to simplify the expression. Applying this to Equation (3.35) would yield the following:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_2)}. \tag{3.36}$$

Finally, to get one SSIM value for the entire image, the average of all SSIM values of all (N) patches is taken:

$$SSIM(I, I_{GT}) = \sum_{x \in I, y \in I_{GT}} \frac{SSIM(x, y)}{N}. \tag{3.37}$$

Going back to the example in Figure 3.13, the SSIM for image (b) is 0.9168 and the SSIM for image (c) equals 0.6949. Meaning according to the SSIM, image (b) is much better. Thus, for this example, the SSIM metric captures the perceived quality more correctly than the RMSE.

3.8.3 VMAF: Video Multimethod Assessment Fusion

A fairly new metric to assess the perceptual visual quality in videos has been introduced by Netflix in 2016 [65]. This metric, called video multimethod assessment fusion, short VMAF, promises better results for estimating video quality than traditional metrics, such as the peak signal-to-noise ratio (PSNR) or the structural similarity index measure (SSIM). The idea behind VMAF is to fuse together several image quality metrics with a machine learning approach. Similarly, to the other metrics, VMAF compares a distorted video to an undistorted reference video.

Figure 3.14 shows the basic composition of a VMAF framework. The first part of the framework consists of human visual system modeling. For this, different spatial and temporal feature metrics are extracted from each frame. The spatial metrics include the *visual information fidelity* (VIF), which describes information fidelity loss, and a *detail loss metric* (dLM), which measures the loss of details. A temporal feature metric describes the motion taking place in the video. The second part of the framework implements a machine learning approach. Here, lab-based subjective scores on a variety of representative video distortion examples are collected. For this, test subjects rate different videos on an absolute category rating (ACR) scale, consisting of *bad, poor, fair, good* and *excellent*. These scores defined by test subjects are then linearly mapped to VMAF scores, in which *bad* corresponds to 20, *poor* to 40, *fair* to 60, *good* to 80 and *excellent* to 100. Thus, the higher the VMAF score, the better the quality of the underlying video. Then VMAF trains models to predict the scores a human would give to a distorted video. Support vector machines (SVM) regression is then used to fuse together the different metrics and align them with the collected scores. Since VMAF defines a score for each video frame, these per-frame scores are then summarized in a temporal pooling, usually with an arithmetic mean, to assign one score for an entire video.

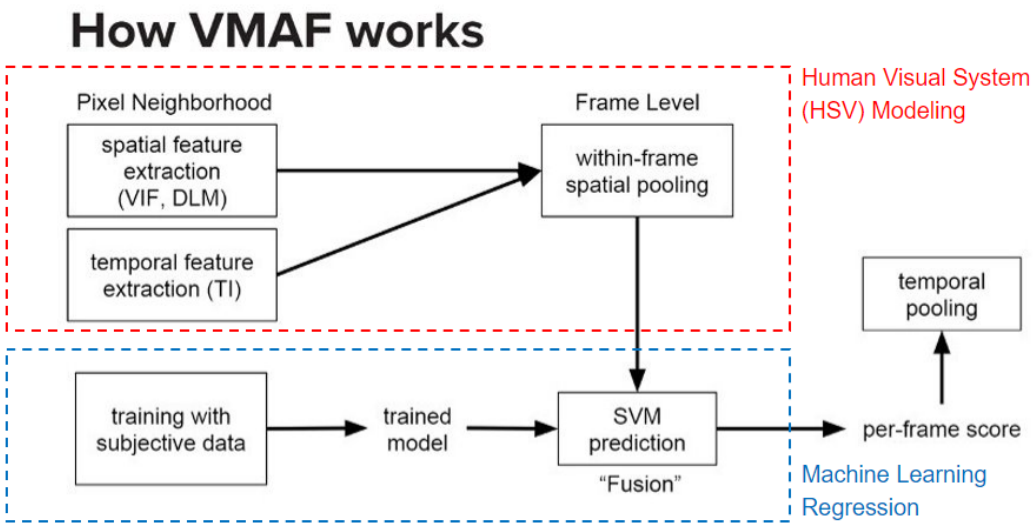


Figure 3.14: VMAF framework. Image source: [66].

4 Experiments

The main question of this thesis is how the quality of an optical flow influences the interpolation. Intuitively, one would assume that the better a flow estimation is, the better the resulting interpolated images are. To test this, we perform several interpolation experiments with a RAFT and ProFlow baseline flow and compare what influence each flow has on the interpolated images. Additionally, we make use of the ground truth flows to obtain an oracle that tells us how 'good' an interpolation can get when the most accurate flow is used. For these experiments, we compare how the different interpolation methods, discussed in Section 3.2, perform based on which optical flow is used. Here, the focus lies on the forward warping and forward forward warping method, since a forward warping method enables a quick computation and a forward forward warping yields better results.

Starting off the experiments, we compare which areas are captured better with a RAFT flow than a ProFlow baseline flow in Section 4.1. Following this, we then examine the interpolation results. A general overview on which optical flow yields better interpolation results is given in Section 4.2. Here, we consider all interpolation methods mentioned in Subsection 3.2.3. For evaluating the results, the metrics SSIM, shown in Subsection 3.8.2 and RMSE, shown in Subsection 3.8.1, are used, with the focus being on SSIM. Afterwards, more detailed results of a forward warping and forward forward warping method are provided to give more context. In Section 4.3, we go further and investigate which areas of the resulting images are erroneous and whether they correspond to the shown optical flow results. Here, we calculate the SSIM for each pixel separately and compare the results. Furthermore, an investigation on the used error metrics is performed as well. This leads to the evaluation of the results using a third metric, the VMAF, introduced in Section 3.8.3. Concluding the examination of the resulting images, we compare the main differences between the interpolated images generated with a RAFT and ProFlow baseline flow. To gain more insight as to how good the results of an interpolation can get, we use the ground truth flows provided for the Sintel dataset for the interpolation. This is done in the form of an oracle experiment in Section 4.4. Here, the results of a forward warping and forward forward warping are presented. Concluding the experiments, in Section 4.5 we examine the main problems that occur in an interpolation using a ground truth flow. To investigate what the cause of these problems can be, we test various approaches, such as using a Gaussian filter on the flow, leaving out the inpainting step, or making use of disocclusion maps. In a final summary in Section 4.6, we recapitulate the results and information gathered throughout this chapter and highlight the main points. To save space, the flows estimated with the baseline of ProFlow are referred to as only ProFlow in

the tables throughout this chapter.

For the implementation of the interpolation methods, the code provided by N.B.Senn was used. Additionally, for calculating the RMSE, the code of N. B. Senn was used as well. To calculate the SSIM, the scikit-image library [67] in Python was utilized. The RAFT optical flows were generated with the code available on Github and provided by Z. Teed *et al.* The ProFlow baseline flows were provided for this thesis and generated as described in [2].

In the following experiments, three different benchmarks are used. From these benchmarks, five or seven different scenes were chosen, each containing three consecutive frames. In order to evaluate an interpolated image with one of the metrics from Section 3.8, we need a ground truth image. For the used interpolation methods, discussed in Section 3.2, the first frame of each scene is chosen as image I_0 and the third frame is chosen as image I_1. Meaning the forward flow $\overrightarrow{v_{0,f}}$ goes from the first to the third frame, and the backward flow $\overrightarrow{v_{1,b}}$ ranges from the third to the first frame. For the distance t we chose 0.5, in order to interpolate in the middle of I_0 and I_1. Meaning the interpolation is performed between the first frame and the third frame. This way we can use the second frame as the ground truth image. Figure 4.1 showcases this for the scene *Basketball* of the Middlebury benchmark [3].

To build on the work of N. B. Senn [12], we use the same dataset. This is the Middlebury benchmark [3] consisting of seven, mostly real-life sequences. These scenes include *Army, Backyard, Basketball, Dumptruck, Evergreen, Grove* and *Mequon*. For each scene in the Middlebury benchmark, frames 10, 11, and 12 were chosen. Frame 10 represents image I_0, frame 12 stands for image I_1 and frame 11 is chosen as the ground truth image. Most of these sequences contain a movement that is solely due to the objects moving in the scene, rather than a camera movement. To obtain more data results, additional two benchmarks are used in the experiments. These are the KITTI 2015 version [4] and Sintel [5] benchmark.

The KITTI benchmark consists of real-life sequences capturing various traffic scenes. Most of these scenes contain moving objects, mainly cars driving, as well as camera movement. The used scenes were taken from the folder "image_2" inside of the folder "testing". These scenes are “00000”, “000104”, “000123”, “000167”, and “000199”. To simplify the usage of these, we refer to “00000” as *Scene 1*, “000104” as *Scene 2*, “000123” as *Scene 3*, “000167” as *Scene 4*, and “000199” as *Scene 5*. For each scene, the frames 10, 11, and 12 were taken and used analogously to the frames from the Middlebury benchmark. Meaning frame 10 represents I_0, frame 12 stands for I_1, and frame 11 is used as the ground truth image.

For the Sintel benchmark, the scenes *Alley 1, Ambush 4, Cave 2, Sleeping 2* and *Temple 2* are used. From the scene *Alley 1* and *Sleeping 2*, frames 1, 2, and 3 were chosen. For *Ambush 4* frames 7, 8, and 9 were chosen. From *Cave 2*, frames 17, 18, and 19 are used and frames 9, 10, and 11 were taken for scene *Temple 2*. Unlike the other two benchmarks, Sintel consists solely of synthetic sequences. These scenes range from small and easy motion to more complex

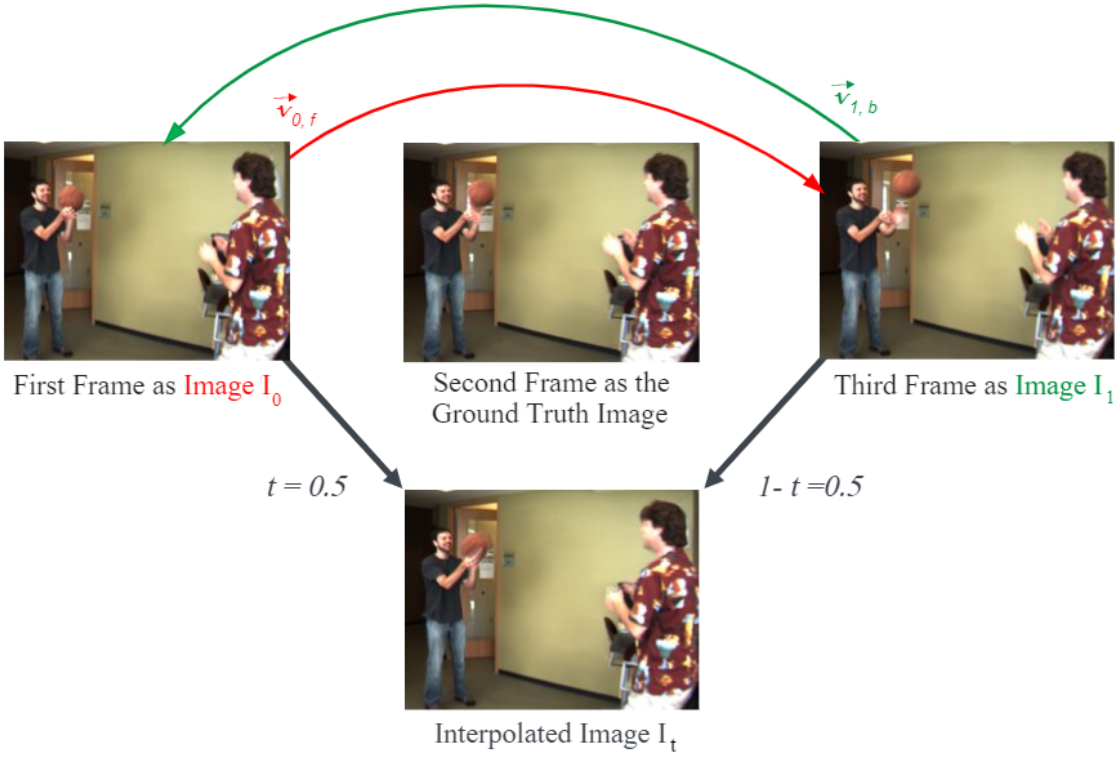


Figure 4.1: Demonstration of how the frames of each dataset are used for interpolation. Image source for the used frames: [3].

motions. In this benchmark, most scenes depict a camera motion as well as moving objects in the scene, similarly to the KITTI benchmark. Since the Sintel benchmark consists of synthetic scenes, a ground truth flow is provided for each scene, which can be used to calculate the EPE for an optical flow. This is done in the following section.

4.1 Optical Flow with RAFT vs the ProFlow Baseline

The central question of this thesis is whether a better optical flow corresponds to better interpolation results. For this, we looked at two separate optical flow estimation methods, RAFT and the ProFlow baseline in Section 2.5 and 2.4. Generally speaking, RAFT provides better optical flow estimations than the ProFlow baseline. To showcase this on the used datasets, this section provides details on the optical flows generated with RAFT and the ProFlow baseline, as well as an EPE investigation for the Sintel dataset.

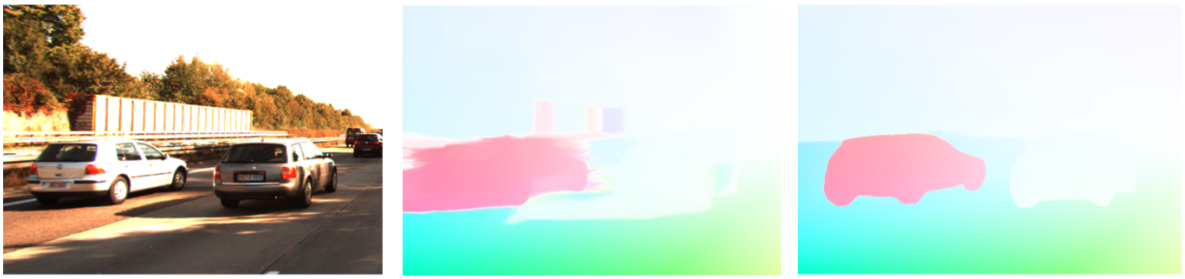
Figure 4.2: Optical flow visualization of scene *Backyard*(a) with the ProFlow baseline (b) and RAFT (c) flow, showing blurred outlines of the children in (b) and sharp outlines in (a). Image source of (c): [3].

4.1.1 Comparison of Visualized Optical Flows

First, to get a general idea about the differences between the RAFT and the ProFlow baseline flows we compare their visualization. For this, we use the visualization with the color map that was shown in Section 2.1 in Figure 2.3. In Section 2.6, we have already seen the optical flow for *Alley 1* in Figure 2.6. The main visible differences were that the RAFT flow has more clean-cut outlines than the ProFlow baseline one. On the other hand, the ProFlow baseline flow appears to have more details in textured areas such as the hair. To investigate this further, more examples are given in the following. For these following examples, the forward flows ranging from image I_0 to I_1 are shown. To provide more reference, the respective image I_0 is shown as well.

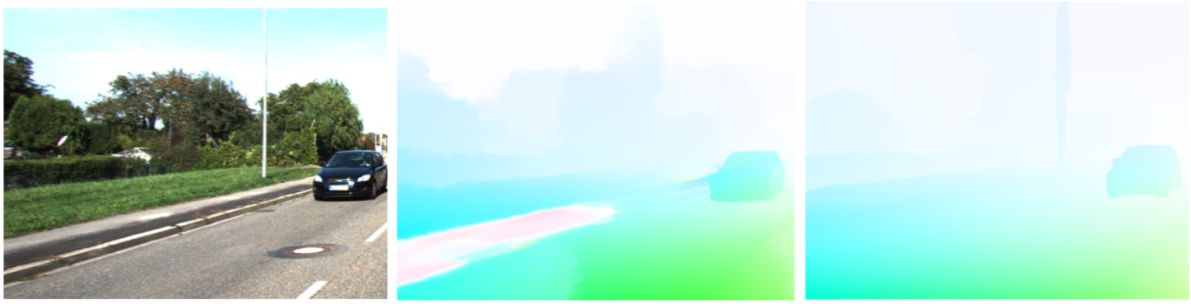
Figure 4.2 shows the optical flows, generated with the ProFlow baseline (b) and RAFT (c) for the scene *Backyard* (a) of the Middlebury dataset. Here, it is clear that the RAFT flow can detect outlines very well for objects that are easily distinguishable from the background. This is visible in the outlines of the children in the scene. In the flow generated with the ProFlow baseline, we can see that the outlines are much more blurred and some parts of the moving objects are not captured at all, such as the legs of the child in the center and in the back.

In Figure 4.3 we can also see a similar behavior. This figure shows a cropped *Scene 5* (a) of the KITTI dataset, with the ProFlow baseline flow (b) and the RAFT flow (c). Here, we can see that the outlines of the moving cars are better captured with the RAFT flow. On the other hand, the ProFlow baseline flow is able to capture details in textured areas better. This can be seen in the columns of the fence over the cars. RAFT does not capture these at all, while they are visible in the ProFlow baseline flow.



(a) I_0 (b) ProFlow Baseline (c) RAFT

Figure 4.3: Optical flow visualization of *Scene 5* (a) with the ProFlow baseline (b) and RAFT (c) flow, showing blurred outlines of the car in (b) and sharp outlines in (c). On the other hand, (b) provides more details in the fence above the cars which are not visible in (c) at all. Image source of (a): [4].



(a) I_0 (b) ProFlow Baseline (c) RAFT

Figure 4.4: Optical flow visualization of *Scene 2* (a) with the ProFlow baseline (b) and RAFT (c) flow, with (b) showing more details in the trees in the background than (c). Image source of (a): [4].

Another example of such a textured area that is better captured with the ProFlow baseline is depicted in Figure 4.4. In this figure, *Scene 2* (a) of the KITTI dataset is shown with the optical flow estimated with the ProFlow baseline (b) and RAFT (c). Here, we can see that the texture of the trees is better captured in the ProFlow baseline flow while RAFT appears to smooth over such textured areas. Additionally, the sidewalk on the left is also not depicted in the RAFT flow at all.

Lastly, the aforementioned characteristics of the flows can also be seen in Figure 4.5. Here, we see the ProFlow baseline flow (a) and the RAFT flow (b) for the scene *Temple 2* (c) of the Sintel benchmark. As we can see, the outlines of the temple, the dragon, and the person in the scene are more clean-cut in the RAFT flow. In the ProFlow baseline flow, these objects appear more blurred.

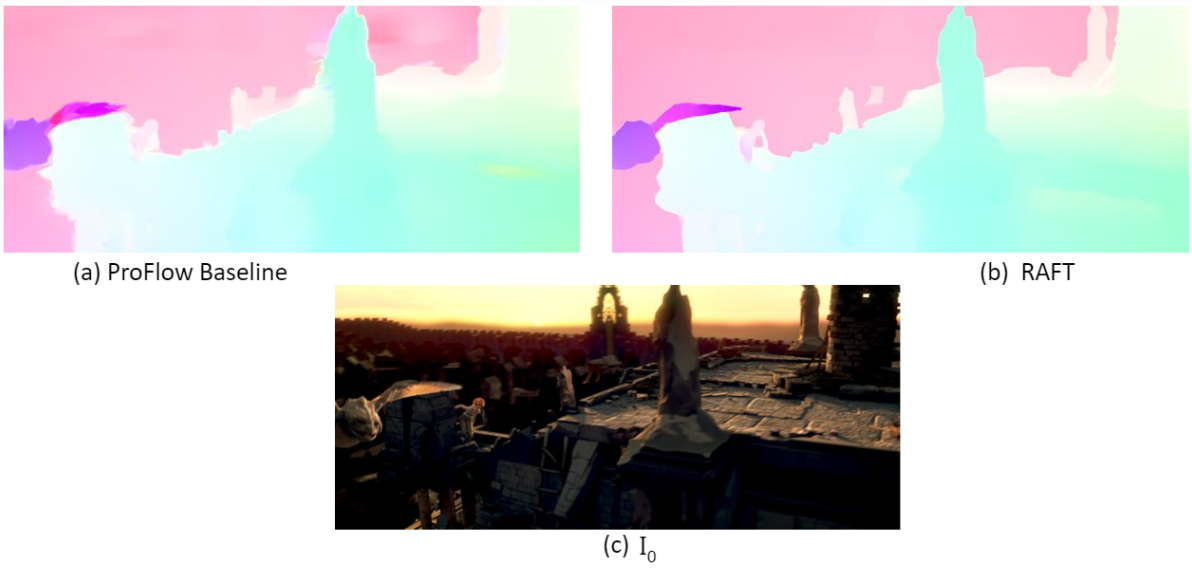


Figure 4.5: Optical flow visualization of scene *Temple 2* (c) with the ProFlow baseline (a) and RAFT (b) flow, showing blurred outlines in (a) and sharper outlines in (b). Image source of (c): [5].

Overall, we can conclude that RAFT generally provides better and sharper outlines of moving objects. Textured areas such as leaves of trees or hair are better depicted with the ProFlow baseline flows. In these areas, RAFT seems to smooth over the texture and only focus on the outlines of the objects. To investigate further how accurate a RAFT or ProFlow baseline flow is, we use the metric for determining the quality of an optical flow that was discussed in Section 2.6.

4.1.2 Comparison of EPE

Determining the quality of an optical flow by examining its visualization is not always easy. Hence why we investigate the quality based on a metric. For this, we calculate the EPE of the generated optical flows with RAFT and the ProFlow baseline. Since the Sintel benchmark provides ground truth flows, we can compute the EPE of the flows generated for the used Sintel sequences. Table 4.1 shows the EPE of the RAFT optical flow and the ProFlow baseline optical flow. In the interpolation experiments we use flows ranging from the first frame I_0 to the third frame I_1. However, the ground truth flows only range from a frame to its immediate consecutive frame. Hence why we compare either the flows ranging from the first frame to the second frame, so from image I_0 to the ground truth image, or from the second frame to the third frame, meaning from the ground truth image to I_1. Additionally, to investigate the used

	EPE - Sintel									
	Alley 1		Ambush 4		Cave 2		Sleeping 2		Temple 2	
	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
Forward Flow of First Frame to Second Frame	0.127	**0.118**	61.711	**22.594**	3.298	**1.338**	**0.049**	0.075	1.000	**0.510**
Forward Flow of Second Frame to Third Frame	0.145	**0.123**	57.922	**13.903**	4.465	**1.749**	**0.050**	0.070	0.917	**0.552**
Halved Forward Flow of First Frame to Third Frame	0.224	**0.203**	57.527	**42.946**	7.062	**4.708**	**0.041**	0.061	0.946	**0.583**

Table 4.1: EPE for optical flows estimated with the ProFlow baseline and RAFT for the Sintel benchmark. Bold marks the better EPE for the scene.

flows ranging from the first frame I_0 to the third frame I_1, we halve them. For these halved flows we then calculate the EPE with the ground truth flow ranging from the first I_0 frame to the second frame, the ground truth image. The first row shows the EPE of the optical flow ranging from the first frame to the second frame. The second row presents the EPE of the flow for the second frame to the third frame. In the third row, the EPE for the halved flows ranging from the first frame to the third frame is shown. The boldly printed EPE signalizes the better EPE for the scene. Since the main focus lies on the interpolation results, we are not going into EPE comparisons of other benchmarks. For EPE results of additional datasets, please refer to [1] and [2].

From Table 4.1, we see that for four out of five scenes, namely *Alley 1, Ambush 4, Cave 2*, and *Temple 2*, RAFT provides a better optical flow estimation than the ProFlow baseline. For scenes *Alley 1* and *Sleeping 2* the difference between the EPE of RAFT and the ProFlow baseline is the smallest. These scenes capture the smallest motion. *Sleeping 2* also contains more camera movement than *Alley 1*. Additionally, in this scene, there is no moving object. The only motion taking place is from the camera movement. This is something the ProFlow baseline appears to capture better. The flows for *Ambush 4* are of the worst quality out of all scenes. This is due to the fact that this scene has the largest, most complex motion. The RAFT flow for this scene is much better than the one generated with the ProFlow baseline. The scenes *Cave 2* and *Temple 2* contain a rather large motion as well. Here, RAFT also provides a much better flow. We can see that the larger the motion is, the bigger the discrepancy between the EPE of RAFT and the ProFlow baseline, with RAFT having a better EPE. Although the halved flows are generally worse than the other flows, overall they still provide good results, except for scene *Ambush 4*. Overall, for larger motion RAFT provides much better results than the ProFlow baseline. For smaller motion RAFT seems to overall do better as well. Most of these scenes contained a mix of well-visible outlines as well as a little texture. Still, the RAFT flow is better for most scenes.

To get an even better understanding of which areas are better according to the EPE, we consider

(a) ProFlow Baseline (b) RAFT

Figure 4.6: Visualization of bad EPE areas in the optical flows of *Alley 1* generated with the ProFlow baseline (a) and RAFT (b). The red areas mark the worse EPE. Thus the outlines in (a) are less accurate than in (b).

the EPE of each pixel individually. This way we can see which areas are better detected in the optical flow. To visualize this, an RGB image is chosen to represent areas of worse optical flow. Here, the G and B values have 0 assigned to them in each pixel, while the R value corresponds to the calculated EPE for this pixel. The higher the EPE, the larger the R value. Thus, the larger the EPE of a pixel, the redder this pixel is. And the smaller the EPE of a pixel, the more black this pixel gets. This way, the redder an area is, the worse the optical flow estimates this area. In the following, such visualizations are presented for the scenes *Alley 1* and *Temple 2*.

Figure 4.6 showcases such EPE visualization for a cropped *Alley 1* scene with the ProFlow baseline flow (a) and the RAFT flow (b). Since this scene has a rather small motion, mostly taking place in the center, we only examine the cropped version containing the biggest differences. As we have seen previously in Figure 2.6, the ProFlow baseline optical flow for this scene captures the outlines of the moving person not as well as RAFT. This is also reflected in Figure 4.6. Here, the outlines of the moving person are much brighter and redder in the ProFlow baseline visualization. Meaning, according to the EPE, RAFT provides much better outlines of the moving person in the scene.

Another example of this is shown in Figure 4.7. Here, the ProFlow baseline flow (a) and the RAFT flow (b) are shown for the scene *Temple 2*. This scene is cropped as well to showcase the biggest differences between RAFT and the ProFlow baseline. In Figure 4.5, we have seen that the outlines of the moving objects are more blurred in the ProFlow baseline flow. Looking at these EPE visualizations, we can see that these outlines are much brighter for the ProFlow baseline flow. This is especially visible in the outlines of the tower in the center of the scene or the outlines of the dragon. Meaning here, the outlines are better estimated with the RAFT flow as well.

(a) ProFlow Baseline (b) RAFT

Figure 4.7: Visualization of bad EPE areas in the optical flows of *Temple 2* generated with the ProFlow baseline (a) and RAFT (b). The red areas mark the worse EPE. Thus the outlines in (a) are less accurate than in (b).

To summarize the findings of this section, we have seen that RAFT generally provides better optical flow estimations, particularly for larger motions. This is especially the case for the outlines of moving objects. These appear much more clean-cut and well defined, whereas in a ProFlow baseline flow they appear more blurred and not as clear. On the other hand, the ProFlow baseline captures textured areas a little better, as seen in Subsection 4.1.1. Here, RAFT smoothes over such areas. Additionally, in case of camera movement, the ProFlow baseline tends to work here better as well, such as for the scene *Sleeping 2*.

Now that we have a basic idea of which optical flow estimation is better and what areas seem to be captured better with a RAFT or ProFlow baseline optical flow, we can examine whether the quality of the interpolation results corresponds to these findings.

4.2 General Comparison of Interpolation Results

In this section, we test how an interpolation performs based on which flow was used. First, we perform each interpolation method, introduced in Section 3.2, with a RAFT and ProFlow baseline flow. For this, we showcase the winner of each interpolation method for each benchmark in a separate table. To calculate the winner, first, each interpolation method was used with a 1N uniform, 1N weight, 1N weight winner, 4N uniform, 4N weight, and 4N weight winner strategy on each scene of the used dataset. Then the result of each used strategy was evaluated with the SSIM and RMSE metric. The optical flow that had a higher number of better results out of those six strategies was chosen as the winner. In case both optical flow methods had the same amount of better results, the winner remains a blank. Additionally, the average of each

scene was taken and compared as well. The winner of these averages is given in the table as well. To showcase which optical flow had more wins, the cells are colored accordingly. Light blue symbolizes a better result with the ProFlow baseline, and purple signalizes that RAFT had better results. Although both RMSE and SSIM are provided, the focus is predominantly on the SSIM results. Table 4.2 showcases the winners for the Middlebury dataset, Table 4.5 for the KITTI dataset and Table 4.8 for the Sintel dataset. The results of these tables are discussed in the following subsections.

To get more details on the actual SSIM and RMSE results of each scene with each splatting strategy, the results of a forward warping interpolation and a forward forward warping interpolation are shown as well. Tables 4.3, 4.6 and 4.9 show the results of a forward warping interpolation for the Middlebury, KITTI and Sintel dataset, and Tables 4.4, 4.7 and 4.10 show the results of a forward forward warping interpolation. For each scene, the ProFlow baseline result is given on the left side and the RAFT result on the right. Both SSIM and RMSE are used, with the SSIM result as the first result and RMSE below. A result was calculated for each of the six splatting strategies, mentioned in Section 3.3. For each scene, the better result of either RAFT or the ProFlow baseline is marked in bold. Lastly, the average of all scenes is pictured on the right. Here, the winner is also showcased in bold. Additionally, to save space, the splatting strategies shown in the following Tables are shortened from uniform to 'U', weight to 'W', and weight winner to 'WW'. The results of these tables are also explained in detail in the following subsections.

4.2.1 Middlebury

Starting with the Middlebury dataset, a first glance at Table 4.2 shows that here an interpolation using a RAFT flow generally performs better. The average of each scene also has more RAFT winners, with both metrics. Overall, using the SSIM metric yields more RAFT 'winners', than using the RMSE metric. Looking at only the SSIM results, RAFT performs better in almost every scene, except for *Mequon*. Using the RMSE metric, however, the results for scenes *Army* and *Evergreen* are better with the ProFlow baseline. For *Dumptruck* both SSIM and RMSE yield RAFT as the winner in all cases. Overall, *Backyard, Basketball, Dumptruck* and *Grove* have the most RAFT winners. The scenes *Backyard, Basketball* and *Dumptruck* have predominantly moving objects in a mostly static background. Additionally, these moving objects have mostly clean-cut and well-visible outlines that separate them well from the background. For example, the thrown ball in *Backyard* or *Basketball*, or the moving car in *Dumptruck* are easily distinguishable from the non-moving background. This is in contrast to other scenes where the moving object is not as easily differentiated from the background, such as in *Evergreen*. This seems to correspond with how the RAFT optical flow captures clean-cut outlines of moving objects better if they are in high contrast from the background, as we have seen in Section 4.1. For scenes *Army* and *Evergreen*, the ProFlow baseline performed better in some cases, although the overall SSIM winner remains RAFT. As mentioned above,

		Middlebury							
		Army	Backyard	Basketball	Dumptruck	Evergreen	Grove	Mequon	∅
Forward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	RAFT	RAFT	RAFT
Forward Warping with Backward Flow	SSIM	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	-
Backward Warping	SSIM	ProFlow	RAFT	-	RAFT	ProFlow	RAFT	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	RAFT
Backward Warping with Backward Flow	SSIM	ProFlow	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
Symmetric Forward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	RAFT	ProFlow	ProFlow	RAFT
Symmetric Forward Warping with Backward Flow	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	RAFT	RAFT	RAFT
Symmetric Backward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	ProFlow	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT
Symmetric Backward Warping with Backward Flow	SSIM	RAFT	RAFT	RAFT	RAFT	-	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	ProFlow	ProFlow	RAFT
Forward Forward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	RAFT	RAFT	RAFT
Backward Backward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	-	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	RAFT
Bidirectional Symmetric Forward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	RAFT	RAFT	RAFT
Bidirectional Symmetric Backward Warping	SSIM	RAFT	RAFT	RAFT	RAFT	RAFT	RAFT	ProFlow	RAFT
	RMSE	ProFlow	RAFT	RAFT	RAFT	ProFlow	RAFT	RAFT	RAFT

Table 4.2: 'Winners' of each interpolation method for the Middlebury dataset. For each scene, the SSIM and RMSE results are calculated for each of the six splatting strategies. The optical flow whose interpolated image has the higher amount of 'wins' out of these six splatting strategies is marked as the winner for the scene. In the case of a tie, the winner remains blank. On the right, the average results of all scenes are calculated and the winner of these averages is given.

Evergreen contains more textured areas and less visible outlines, such as the leaves of the tree in the scene. For these characteristics, RAFT seems to not perform as well. This is especially the case for the RMSE results, where, as mentioned, the ProFlow baseline has more winners. Lastly, although *Mequon* does have moving objects with clear outlines, it also contains a moving background. This is something that seems to work better with a ProFlow baseline flow.

From this general comparison, we now go further and examine the forward warping interpolation in Table 4.3 more closely. As seen in the overall comparison from Table 4.2, the forward warping method has overall better results with a RAFT flow. The average is also better with RAFT, although the SSIM values are quite close for RAFT and the ProFlow baseline. As seen before, *Backyard, Basketball, Dumptruck* and *Grove* have RAFT as a clear winner. The scene *Evergreen* has much better SSIM and RMSE results with a ProFlow baseline flow. As mentioned before, this seems to be the case due to the fact that the moving object in the scene has outlines that are hard to differentiate from the background. Comparing the differences in the resulting SSIM values between RAFT and the ProFlow baseline, we can see that *Dumptruck* and *Mequon* have the biggest differences. For *Dumptruck*, RAFT performs much better, whereas,

		Forward Warping - Middlebury															
		Army		Backyard		Basketball		Dumptruck		Evergreen		Grove		Mequon		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	0.879	**0.882**	0.727	**0.742**	0.480	**0.487**	0.848	**0.868**	**0.757**	0.748	0.652	**0.668**	**0.752**	0.718	0.728	**0.730**
	RMSE	**8.220**	8.641	22.201	**20.692**	22.863	**20.576**	30.497	**20.971**	**23.697**	25.366	38.779	**37.417**	17.382	**16.341**	23.377	**21.429**
1N W	SSIM	0.879	**0.882**	0.727	**0.741**	0.480	**0.487**	0.848	**0.868**	**0.757**	0.748	0.652	**0.668**	**0.752**	0.719	0.728	**0.730**
	RMSE	**8.224**	8.619	22.201	**21.021**	22.860	**20.737**	30.493	**21.125**	**23.698**	25.359	38.777	**37.475**	17.402	**16.274**	23.379	**21.516**
1N WW	SSIM	0.878	**0.881**	0.726	**0.737**	0.480	**0.484**	0.848	**0.868**	**0.756**	0.747	0.652	**0.662**	**0.750**	0.715	0.727	**0.728**
	RMSE	**8.284**	8.790	**22.260**	22.850	22.903	**21.684**	30.509	**22.222**	**23.798**	25.400	38.853	**38.615**	17.622	**16.950**	23.461	**22.359**
4N U	SSIM	**0.824**	0.819	0.636	**0.645**	0.471	**0.473**	0.717	**0.736**	**0.728**	0.720	0.650	**0.665**	**0.765**	0.745	0.684	**0.686**
	RMSE	**9.929**	10.725	24.713	**24.041**	22.399	**20.172**	35.861	**29.307**	**24.027**	25.685	36.150	**35.214**	16.652	**15.923**	24.247	**23.010**
4N W	SSIM	**0.906**	0.905	0.734	**0.748**	0.495	**0.500**	0.852	**0.874**	**0.779**	0.772	0.694	**0.708**	**0.802**	0.773	0.752	**0.754**
	RMSE	**6.479**	7.406	21.454	**20.185**	22.065	**19.760**	29.508	**19.910**	**22.571**	24.359	35.143	**34.463**	16.194	**15.277**	21.916	**20.194**
4N WW	SSIM	0.878	**0.880**	0.726	**0.736**	0.479	**0.483**	0.848	**0.867**	**0.755**	0.747	0.651	**0.658**	**0.749**	0.714	**0.727**	0.726
	RMSE	**8.311**	8.942	**22.313**	23.114	22.954	**21.924**	30.524	**22.556**	**23.886**	25.505	**39.032**	39.398	17.680	**17.090**	23.529	**22.647**

Table 4.3: SSIM and RMSE results of a forward warping interpolation of the Middlebury dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right, the average of all scenes is given for each splatting strategy.

for *Mequon*, the ProFlow baseline has much better results. The biggest difference between these scenes is that, as mentioned above, the background in *Mequon* moves, as opposed to *Dumptruck*, where it is static. As already discussed, this is something that tends to work better with the ProFlow baseline. Whereas a static background with well-visible outlines of moving objects works better with RAFT.

In Table 4.3 we see the forward forward warping interpolation results. Generally, the forward forward warping results are better than the forward warping ones for each scene. Similar to the forward warping results, here, RAFT has overall better results than the ProFlow baseline. The average results are also better with RAFT in a forward forward warping interpolation. However, the difference in the average results between RAFT and the ProFlow baseline is larger than in a forward warping. Another difference to the forward warping results is that the scene *Evergreen* is no longer better with the ProFlow baseline according to the SSIM. Nonetheless, *Mequon* remains better with the ProFlow baseline, if evaluated with the SSIM. Scenes *Backyard, Basketball, Dumptruck* and *Grove* also have not changed, and remain better with RAFT. Although for a forward warping of scene *Army*, the ProFlow baseline had better SSIM results for a few splatting strategies, RAFT was overall better. In forward forward warping, this changes and *Army* has better results with the ProFlow baseline. This might be due to the fact that the motion in this scene is very small and a clear 'winner' is not as easily determined since both flows provide good results.

To sum up the presented results for the Middlebury dataset, a RAFT flow seems to work better for scenes where we have less camera movement and more moving objects in the scene. Additionally, scenes where the objects have simple and well-visible outlines and are thus easily distinguished from the background, are also better with RAFT. For scenes such as *Backyard, Basketball* and *Dumptruck* this was predominantly the case. On the other hand, *Evergreen* contained more outlines that were harder to distinguish from the background. Here, the

		Forward Forward Warping - Middlebury															
		Army		Backyard		Basketball		Dumptruck		Evergreen		Grove		Mequon		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	0.919	**0.922**	0.768	**0.781**	0.524	**0.532**	0.869	**0.886**	0.800	**0.801**	0.702	**0.721**	**0.817**	0.792	0.771	**0.776**
	RMSE	**5.882**	6.139	18.683	**16.410**	17.169	**16.077**	24.006	**17.849**	**18.540**	19.267	31.975	**30.774**	14.268	**13.633**	18.646	**17.164**
1N W	SSIM	0.919	**0.922**	0.768	**0.781**	0.524	**0.532**	0.869	**0.886**	0.800	**0.801**	0.702	**0.721**	**0.817**	0.792	0.771	**0.776**
	RMSE	**5.882**	6.109	18.683	**16.799**	17.177	**16.174**	24.007	**17.870**	**18.542**	19.283	31.975	**30.790**	14.271	**13.649**	18.648	**17.239**
1N WW	SSIM	0.918	**0.922**	0.768	**0.778**	0.524	**0.529**	0.869	**0.886**	0.800	**0.801**	0.702	**0.716**	**0.816**	0.790	0.771	**0.775**
	RMSE	**5.901**	6.151	18.707	**18.119**	17.232	**16.837**	24.023	**18.394**	**18.582**	19.397	32.044	**31.455**	14.344	**14.286**	18.690	**17.806**
4N U	SSIM	**0.865**	0.863	0.692	**0.699**	0.505	**0.507**	0.764	**0.770**	**0.779**	0.772	0.673	**0.693**	**0.789**	0.774	0.724	**0.725**
	RMSE	**7.387**	7.814	20.617	**18.883**	16.668	**15.620**	28.641	**24.728**	**18.901**	20.050	31.646	**30.343**	14.142	**13.648**	19.715	**18.727**
4N W	SSIM	0.917	**0.921**	0.771	**0.786**	0.534	**0.541**	0.871	**0.892**	0.815	**0.816**	0.708	**0.738**	**0.820**	0.805	0.777	**0.786**
	RMSE	**5.771**	5.928	18.367	**15.941**	16.541	**15.419**	23.504	**16.564**	**17.834**	18.648	30.892	**29.151**	13.974	**13.053**	18.126	**16.386**
4N WW	SSIM	0.919	**0.922**	0.768	**0.779**	0.523	**0.529**	0.869	**0.887**	0.800	**0.801**	0.704	**0.722**	**0.817**	0.791	0.771	**0.776**
	RMSE	**5.835**	6.077	18.773	**18.264**	17.264	**16.913**	24.000	**18.400**	**18.622**	19.462	32.013	**31.387**	14.329	**14.200**	18.691	**17.815**

Table 4.4: SSIM and RMSE results of a forward forward warping interpolation of the Middlebury dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right, the average of all scenes is given for each splatting strategy.

ProFlow baseline had more 'wins' than in the previously mentioned scenes. This corresponds to the findings of Section 4.1, where we have seen that a RAFT flow captures clean-cut outlines well for objects that are easily distinguishable from the background. For scenes with a more moving background, the ProFlow baseline tends to work better, which was seen in the results of the scene *Mequon*.

4.2.2 KITTI

Unlike the previous dataset, KITTI consists mostly of scenes with a moving camera. Since all used KITTI scenes capture traffic sequences, all scenes contain backgrounds consisting of large textured areas with less clear outlines, such as trees. As seen in Section 4.1, these areas are better depicted with the ProFlow baseline flow.

In the general comparison in Table 4.5, we can see that interpolation with the ProFlow baseline flow yields better results for the majority of the scenes. According to the SSIM, only *Scene 1* is better with RAFT and the rest is better with the ProFlow baseline. With the RMSE, *Scene 2* is also better with RAFT. The rest is better with the ProFlow baseline according to both SSIM and RMSE. Looking at the average, both RMSE and SSIM results indicate that the ProFlow baseline was better in all cases. These findings correspond to what was seen so far, that the ProFlow baseline tends to work better for camera movement.

In Table 4.6 the results of a forward warping interpolation are shown. Similar to the general comparison, here, the ProFlow baseline has overall better results. The average results are also better with the ProFlow baseline. We can see that for *Scene 2, Scene 3* and *Scene 5*, the ProFlow baseline has the better SSIM results. *Scene 1* is the only scene that is better with RAFT, according to both the SSIM and RMSE. Interestingly, here, we have large camera movement

		KITTI					
		Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	∅
Forward Warping	SSIM	RAFT	ProFlow	ProFlow	RAFT	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Forward Warping with Backward Flow	SSIM	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Backward Warping	SSIM	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
Backward Warping with Backward Flow	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Symmetric Forward Warping	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Symmetric Forward Warping with Backward Flow	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Symmetric Backward Warping	SSIM	-	ProFlow	ProFlow	RAFT	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Symmetric Backward Warping with Backward Flow	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Forward Forward Warping	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Backward Backward Warping	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Bidirectional Symmetric Forward Warping	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow
Bidirectional Symmetric Backward Warping	SSIM	RAFT	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	RAFT	ProFlow	RAFT	ProFlow	ProFlow	ProFlow

Table 4.5: 'Winners' of each interpolation method for the KITTI dataset. For each scene, the SSIM and RMSE results are calculated for each of the six splatting strategies. The optical flow whose interpolated image has the higher amount of 'wins' out of these six splatting strategies is marked as the winner for the scene. In case of a tie, the winner remains blank. On the right, the average results of all scenes are calculated and the winner of these averages is given.

as well as a textured background and no moving objects. However, the overall SSIM and RMSE results indicate that the interpolated image of this scene is one of the worst out of all scenes. As mentioned in Section 4.1.2, RAFT generally provides much better flows for scenes with larger and more complex motion. This could explain why the interpolation results with RAFT are better despite of the mentioned camera movement. For *Scene 4*, although the RMSE indicates that the ProFlow baseline is better, the SSIM is much better for RAFT. *Scene 4* is the only scene where the camera does not move and only has a car movement with clear outlines in the scene. This would again correspond to the findings of Section 4.1, where RAFT depicts moving objects with clear outlines in a static background very well. This scene is quite similar to the scene *Dumptruck* of Middlebury, where moving cars in a still background are depicted. As seen prior, *Dumptruck* also had mostly RAFT winners.

		Forward Warping - KITTI											
		Scene 1		Scene 2		Scene 3		Scene 4		Scene 5		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	0.286	**0.289**	**0.298**	0.292	**0.489**	0.481	0.788	**0.791**	**0.581**	0.570	**0.488**	0.485
	RMSE	41.308	**40.687**	**31.958**	33.053	46.765	**45.808**	**31.245**	32.977	**26.476**	30.581	**35.550**	36.621
1N W	SSIM	0.286	**0.289**	**0.298**	0.292	**0.489**	0.481	0.788	**0.791**	**0.581**	0.570	**0.488**	0.485
	RMSE	41.319	**40.679**	**31.979**	33.089	46.770	**45.867**	**31.341**	32.981	**26.478**	30.617	**35.577**	36.647
1N WW	SSIM	0.284	**0.288**	**0.297**	0.291	**0.487**	0.480	0.787	**0.791**	**0.580**	0.570	**0.487**	0.484
	RMSE	41.619	**40.858**	**32.554**	33.740	47.927	**47.086**	33.166	**33.082**	**26.815**	31.264	**36.416**	37.206
4N U	SSIM	**0.318**	0.315	**0.306**	0.302	**0.515**	0.507	**0.653**	0.652	**0.597**	0.588	**0.478**	0.473
	RMSE	38.456	**38.341**	**30.918**	31.965	43.957	**43.086**	**38.480**	40.898	**25.700**	29.682	**35.502**	36.794
4N W	SSIM	0.311	**0.312**	**0.309**	0.303	**0.520**	0.511	0.790	**0.792**	**0.608**	0.599	**0.508**	0.503
	RMSE	39.152	**38.796**	**30.966**	32.113	44.142	**43.281**	**29.985**	32.912	**24.942**	29.022	**33.837**	35.225
4N WW	SSIM	0.281	**0.285**	**0.293**	0.286	**0.481**	0.475	0.786	**0.790**	**0.576**	0.566	**0.483**	0.480
	RMSE	42.185	**41.439**	**33.430**	34.558	48.465	**47.778**	33.477	**33.375**	**27.206**	31.554	**36.953**	37.741

Table 4.6: SSIM and RMSE results of a forward warping interpolation of the KITTI dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right, the average of all scenes is given for each splatting strategy.

		Forward Forward Warping - KITTI											
		Scene 1		Scene 2		Scene 3		Scene 4		Scene 5		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	**0.298**	**0.298**	**0.304**	0.280	**0.512**	0.501	**0.807**	0.803	**0.608**	0.587	**0.506**	0.494
	RMSE	39.062	**38.650**	**30.722**	33.053	42.379	**41.239**	**33.221**	35.619	**23.465**	26.000	**33.770**	34.912
1N W	SSIM	**0.298**	**0.298**	**0.304**	0.280	**0.512**	0.501	**0.807**	0.803	**0.608**	0.587	**0.506**	0.494
	RMSE	39.061	**38.653**	**30.722**	33.065	42.400	**41.331**	**33.360**	35.695	**23.470**	26.025	**33.803**	34.954
1N WW	SSIM	0.296	**0.297**	**0.303**	0.279	**0.510**	0.498	**0.805**	0.802	**0.607**	0.585	**0.504**	0.492
	RMSE	39.185	**38.814**	**31.000**	33.382	42.939	**42.191**	**34.219**	36.242	**23.714**	26.446	**34.211**	35.415
4N U	SSIM	**0.322**	0.320	**0.311**	0.287	**0.526**	0.516	**0.704**	0.664	**0.614**	0.593	**0.495**	0.476
	RMSE	37.138	**36.918**	**29.962**	32.467	40.800	**39.628**	**36.976**	42.560	**23.108**	25.447	**33.597**	35.404
4N W	SSIM	0.313	**0.314**	**0.313**	0.289	**0.530**	0.520	**0.809**	0.805	**0.623**	0.604	**0.518**	0.506
	RMSE	37.995	**37.575**	**30.057**	32.566	41.059	**39.853**	**32.378**	35.350	**22.727**	24.998	**32.843**	34.068
4N WW	SSIM	0.293	**0.295**	**0.303**	0.280	**0.511**	0.500	**0.806**	0.803	**0.609**	0.587	**0.504**	0.493
	RMSE	39.478	**39.145**	**31.182**	33.775	42.973	**42.315**	**33.360**	36.111	**23.683**	26.267	**34.135**	35.523

Table 4.7: SSIM and RMSE results of a forward forward warping interpolation of the KITTI dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right, the average of all scenes is given for each splatting strategy.

With a forward forward warping interpolation, shown in Table 4.7, we have similar occurrences for *Scene 2, Scene 3* and *Scene 5* as in a forward warping. Interestingly, *Scene 4* yields better results with the ProFlow baseline this time, even though this scene contains no camera movement and consists of clear outlines. An investigation of this is given in Subsection 4.3.2. Lastly, the average of all scenes is better with the ProFlow baseline as well. Overall, the results of a forward forward warping of the KITTI scenes are better than of a forward warping. Additionally, the differences between the RAFT SSIM results and the ProFlow baseline SSIM results differ more than in a forward warping interpolation. This was also the case for the Middlebury dataset.

Summing up the collected results of the KITTI interpolation, here, an interpolation with the ProFlow baseline gave predominantly better results. The KITTI dataset consists of mostly camera movement and a lot of textured background. This ties into what was already seen in the results of the Middlebury dataset, in Subsection 4.2.1, where a ProFlow baseline flow works better for these aspects in an interpolation. An exception is *Scene 4*, where no camera movement is present, yet a forward forward warping interpolation yields better SSIM results for the ProFlow baseline. However, this is further investigated in Subsection 4.3.2.

4.2.3 Sintel

The last used benchmark differs from the previous in terms of the depicted scenes. Here, all scenes are synthetically generated. In Subsection 4.1.2, the optical flows generated with RAFT and the ProFlow baseline for this dataset were already compared. With this knowledge, we can now investigate how the interpolation performed and whether a better optical flow resulted in better interpolation results. The results of each interpolation are seen in Table 4.8. Similar to the KITTI dataset, here, the results with the ProFlow baseline are overall better, even though as we have seen, the RAFT optical flow is better for most of the scenes. Scenes *Ambush 4* and *Cave 2* have better SSIM results with a RAFT flow. As seen in Table 4.1, for these scenes RAFT provided a better optical flow. Meaning here, the quality of the optical flow correlates with the interpolation results. These scenes also contain larger motion, which tends to have better interpolation results when a RAFT flow is used. For scenes *Alley 1, Sleeping 2*, and *Temple 2*, the ProFlow baseline gave better results. For *Sleeping 2*, the ProFlow baseline also provided a better optical flow. This again supports the correlation of the quality of the optical flow and the interpolation results. However, for scenes *Alley 1* and *Temple 2*, RAFT provided a better optical flow. This contradicts the theory that an optical flow results in better interpolation results.

To gain more insight into the resulting SSIM and RMSE values, we consider Table 4.9. In this table, the results of a forward warping interpolation are presented. As we have seen in the EPE comparison from Table 4.1, RAFT had a much better EPE for scene *Ambush 4*. One would expect the interpolation results to reflect this, however, although the results are better with RAFT, the actual SSIM and RMSE values are not that much different from the ProFlow baseline results. Even scene *Temple 2* has bigger differences between the RAFT SSIM results and the ProFlow baseline SSIM results. However, the ProFlow baseline has better interpolation results here. This scene contains some simulated camera movement. We have seen that for this, a ProFlow baseline flow seems to work better. However, RAFT provided a better optical flow for this scene. Yet the interpolation results are not better with RAFT, meaning that the quality of the optical flow does not guarantee a good interpolation result in this case. *Alley 1* also has a relatively significant discrepancy between the RAFT SSIM and the ProFlow baseline SSIM results, despite the EPE differences not being as large. As already mentioned, the optical flow for this scene was better estimated with RAFT, yet the interpolation results with a ProFlow baseline flow are much better. This is again contradicting the assumption

		Sintel					
		Alley 1	Ambush 4	Cave 2	Sleeping 2	Temple 2	∅
Forward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	-	ProFlow	ProFlow	ProFlow
Forward Warping with Backward Flow	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
Backward Warping	SSIM	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	-	ProFlow	ProFlow	RAFT
Backward Warping with Backward Flow	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Symmetric Forward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	-	ProFlow	ProFlow	RAFT
Symmetric Forward Warping with Backward Flow	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Symmetric Backward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	RAFT
Symmetric Backward Warping with Backward Flow	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Forward Forward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
Backward Backward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow	ProFlow
Bidirectional Symmetric Forward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow
Bidirectional Symmetric Backward Warping	SSIM	ProFlow	RAFT	RAFT	ProFlow	ProFlow	ProFlow
	RMSE	ProFlow	RAFT	ProFlow	ProFlow	ProFlow	ProFlow

Table 4.8: 'Winners' of each interpolation method for the Sintel dataset. For each scene, the SSIM and RMSE results are calculated for each of the six splatting strategies. The optical flow whose interpolated image has the higher amount of 'wins' out of these six splatting strategies is marked as the winner for the scene. In case of a tie, the winner remains blank. On the right, the average results of all scenes are calculated and the winner of these averages is given.

that a better optical flow results in better interpolation results. Looking at the average, even though the results are quite close, the ProFlow baseline still provides better results. This is interesting considering the fact that the RAFT optical flows for this dataset are generally better.

To examine whether these findings are the same in a generally better interpolation method, we consider Table 4.10, containing the results of a forward forward warping interpolation. Here, we can see that the average is more mixed. *Ambush 4* also has a much bigger discrepancy in terms of SSIM results than it did in a forward warping interpolation. However, the ProFlow baseline remains the 'winner' for scenes *Alley 1* and *Temple 2*, even though the RAFT optical flow estimation is better for these scenes. Merely the results of scene *Cave 2* changed. Here, we have more RMSE results that are better with the ProFlow baseline, contradicting the SSIM results. An investigation of this is done in Subsection 4.3.2. However, generally, the better

		Forward Warping - Sintel											
		Alley 1		Ambush 4		Cave 2		Sleeping 2		Temple 2		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	**0.893**	0.885	0.083	**0.088**	0.330	**0.348**	**0.906**	0.902	**0.833**	0.815	**0.609**	0.608
	RMSE	**9.843**	10.437	69.711	**66.278**	20.642	**20.474**	**6.559**	7.392	**24.198**	28.063	**26.191**	26.529
1N W	SSIM	**0.893**	0.885	0.083	**0.088**	0.329	**0.348**	**0.906**	0.902	**0.833**	0.815	**0.609**	0.608
	RMSE	**9.708**	10.579	69.722	**66.235**	20.658	**20.509**	**6.555**	7.383	**23.915**	28.090	**26.112**	26.559
1N WW	SSIM	**0.892**	0.884	0.082	**0.088**	0.325	**0.342**	**0.906**	0.902	**0.832**	0.813	**0.607**	0.606
	RMSE	**10.411**	12.102	70.149	**67.232**	**21.516**	21.957	**6.569**	7.463	**24.333**	30.346	**26.596**	27.820
4N U	SSIM	**0.906**	0.899	0.085	**0.093**	0.356	**0.374**	**0.887**	0.883	**0.830**	0.814	**0.613**	**0.613**
	RMSE	**8.656**	9.326	69.689	**66.065**	**20.158**	20.180	**6.791**	7.227	**23.692**	27.849	**25.797**	26.129
4N W	SSIM	**0.926**	0.918	0.084	**0.092**	0.349	**0.367**	**0.943**	0.939	**0.859**	0.841	**0.632**	0.631
	RMSE	**7.992**	8.834	69.813	**66.131**	20.325	**20.276**	**3.913**	5.120	**23.378**	27.578	**25.084**	25.588
4N WW	SSIM	**0.891**	0.883	0.080	**0.087**	0.324	**0.341**	**0.906**	0.901	**0.831**	0.813	**0.606**	0.605
	RMSE	**10.427**	12.160	70.655	**67.373**	**21.723**	22.150	**6.589**	7.530	**24.464**	30.569	**26.772**	27.956

Table 4.9: SSIM and RMSE results of a forward warping interpolation of the Sintel dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right the average of all scenes is given for each splatting strategy.

		Forward Forward Warping - Sintel											
		Alley 1		Ambush 4		Cave 2		Sleeping 2		Temple 2		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
1N U	SSIM	**0.933**	0.928	0.103	**0.125**	0.334	**0.336**	**0.956**	0.952	**0.858**	0.837	**0.637**	0.636
	RMSE	**6.207**	6.697	56.215	**53.282**	**18.707**	19.085	**3.330**	4.036	**19.007**	20.994	**20.693**	20.819
1N W	SSIM	**0.933**	0.929	0.103	**0.125**	0.334	**0.336**	**0.956**	0.952	**0.858**	0.838	**0.637**	0.636
	RMSE	**6.162**	6.702	56.209	**53.373**	**18.706**	19.099	**3.329**	4.030	**18.912**	21.122	**20.664**	20.865
1N WW	SSIM	**0.932**	0.928	0.100	**0.121**	**0.331**	**0.331**	**0.956**	0.952	**0.857**	0.836	**0.635**	0.634
	RMSE	**6.499**	7.194	56.500	**54.999**	**19.022**	19.678	**3.343**	4.088	**19.231**	23.055	**20.919**	21.803
4N U	SSIM	**0.925**	0.921	0.103	**0.126**	0.348	**0.352**	**0.914**	0.911	**0.847**	0.828	0.627	**0.628**
	RMSE	**6.162**	6.557	56.335	**52.916**	**18.579**	18.969	**5.113**	5.400	**18.852**	20.796	21.008	**20.928**
4N W	SSIM	**0.935**	0.931	0.101	**0.125**	0.342	**0.347**	**0.951**	0.949	**0.864**	0.844	**0.639**	**0.639**
	RMSE	**5.997**	6.432	56.431	**52.996**	**18.694**	19.036	**3.661**	4.041	**18.707**	20.658	20.698	**20.633**
4N WW	SSIM	**0.932**	0.928	0.094	**0.117**	0.329	**0.332**	**0.957**	0.952	**0.860**	0.839	**0.634**	**0.634**
	RMSE	**6.526**	7.195	57.191	**54.859**	**19.298**	19.828	**3.287**	4.058	**19.441**	23.062	**21.149**	21.800

Table 4.10: SSIM and RMSE results of a forward forward warping interpolation of the Sintel dataset, using a ProFlow baseline and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, on the right, the average of all scenes is given for each splatting strategy.

RAFT flows did not results in better forward forward warping results in all cases.

Overall, we can see some correlation between the quality of the optical flow and the interpolation results. For scenes *Ambush 4* and *Sleeping 2* the better optical flow corresponded with a better interpolation result. However, even though RAFT provided better optical flow estimations for the Sintel dataset, most of the interpolation results are better with the ProFlow baseline. This can be seen in scenes *Alley 1* and *Temple 2*. Meaning the theory that a better flow estimation provides a better interpolation result is not always the case.

4.2.4 Short Summary

To summarize the results of all datasets, we find that generally, RAFT works better for scenes with moving objects in a static background. Additionally, RAFT also works better when the moving objects have clear outlines and are in high contrast to their background. This is predominantly the case for most Middlebury scenes, where RAFT provided overall more 'winners' than the ProFlow baseline. For more textured areas and a moving background, the ProFlow baseline yields better results. Such characteristics are present in the KITTI dataset, which explains why the images generated with a ProFlow baseline flow yielded overall better results than with RAFT. However, as seen in the Sintel results, a better optical flow does not always correspond to better interpolation results. This is especially the case for the scenes *Alley 1* and *Temple 2*, where a better RAFT flow is given, yet the interpolation results remain better with a ProFlow baseline flow. To investigate how the interpolated images look when a RAFT and a ProFlow baseline flow is used, the next section provides more details on these resulting images.

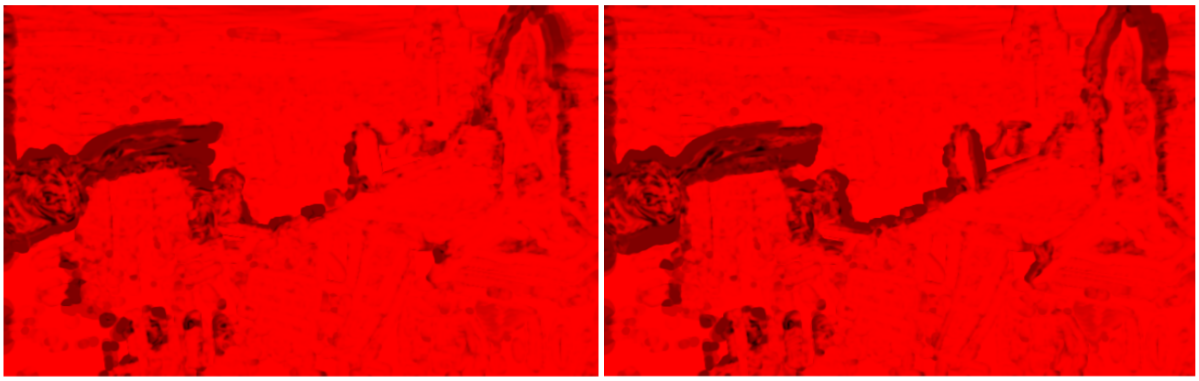
4.3 A Closer Look at the Resulting Interpolated Images

In this Section, we go further into detail on the resulting interpolated images and consider individual examples. As we have seen in the previous section, in cases such as *Alley 1* or *Temple 2* of the Sintel dataset, a better optical flow does not correspond to better interpolation results. To examine this further, we investigate the SSIM for each pixel in the interpolated images in order to understand which areas contain more errors. Furthermore, we compare the differences between the SSIM and RMSE results, to see where the SSIM might have failed as an accurate metric. From this, we explore another metric, the VMAF, to investigate whether this offers an alternative to the SSIM. Lastly, we examine how the interpolated images look like when a RAFT or ProFlow baseline flow is used and what errors appear commonly for each flow.

4.3.1 Erroneous Areas in Interpolated Images

First, we investigate why the scenes *Alley 1* and *Temple 2* had better interpolation results with a ProFlow baseline flow, despite RAFT providing a better optical flow estimation for these scenes. While a general SSIM result of an image is helpful to evaluate it against others, it does not give information about which areas are better or worse. For this, we take the SSIM result of each individual pixel and mark the areas with a worse SSIM black.

Before we can examine the results, we first need to implement a method that calculates the SSIM for each pixel individually. Since the SSIM method provided in the used scikit-image



(a) ProFlow Baseline (b) RAFT

Figure 4.8: Erroneous areas of the interpolated images of *Temple 2*, generated with a ProFlow baseline (a) and RAFT (b) flow. The darker areas showcase the worse SSIM. Thus (b) contains more errors around the outlines than (a). A forward warping method and 4N weight strategy were used for the interpolated images.

library [67] generates an average SSIM for all pixels, this method has to be modified. Normally, the method generates an average SSIM for first, the R value, then the G value, and lastly, the B value of a distorted RBG image. Afterwards, the average of these three values is taken as the resulting SSIM value. To change this, the SSIM for the R value, G value, and B value was generated for each pixel separately. Afterwards, for each pixel, the average SSIM value of its R value, G value, and B value is taken. This results in one SSIM value for each pixel of the image. To visualize the worse SSIM areas, we proceed analogously to Section 4.1.2. We visualize the resulting SSIM values in an RGB image, where we set the G and B value of each pixel to 0. The R value corresponds to the SSIM for this pixel. Since a bigger SSIM corresponds to better quality, this time the red areas showcase a better quality, and the darker the area the worse.

In Figure 4.8 such SSIM visualizations are given for the scene *Temple 2*. Here, we crop the scene to showcase the most important part. For the interpolation in this example, a forward warping method was used as well as a 4N weight winner strategy. The visualization of the interpolated image using a ProFlow baseline flow is shown in (a) and the interpolated image generated with a RAFT flow is seen in (b). Although both images appear quite similar, if we look closer at the outlines of the lower part of the dragon or the top of the tower on the right, we can see that in the RAFT image, they appear darker than in the ProFlow baseline one. Meaning according to the SSIM, the outlines are worse in the image using a RAFT flow. However, if we look back at the EPE visualization in Figure 4.7, these same areas are better detected with the RAFT flow. Thus, a better flow estimation actually generates a worse interpolation in this case.

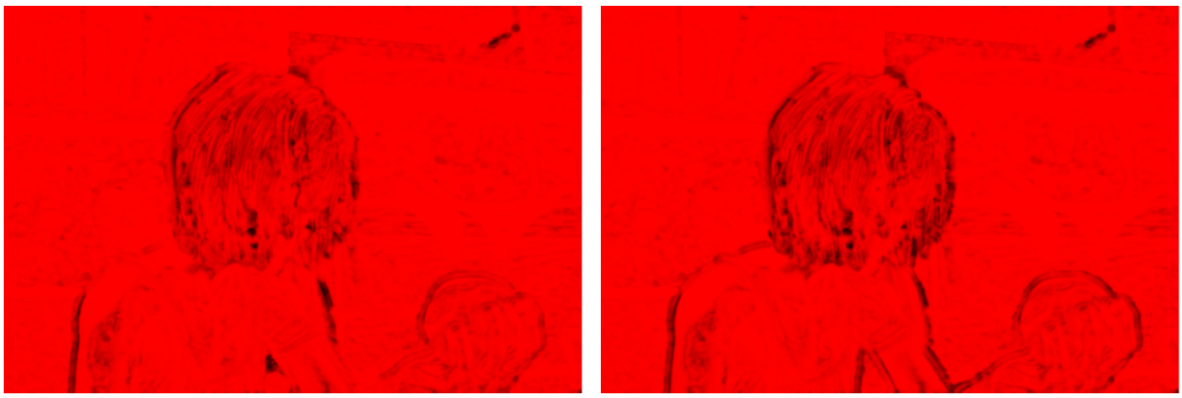


Figure 4.9: Erroneous areas of the interpolated images of *Alley 1*, generated with a ProFlow baseline (a) and RAFT (b) flow. The darker areas showcase the worse SSIM. Thus (b) contains more errors around the outlines than (a). A forward warping method and 4N weight strategy were used for the interpolated images.

The other scene we inspect is *Alley 1*. As mentioned, here, we also have better interpolation results with a ProFlow baseline flow, despite RAFT providing a better flow estimation for this scene. Figure 4.9 shows the SSIM visualization of the interpolated image for this scene, performed with a forward warping method and a 4N weight strategy. In (a) we have the visualization of the image generated with a ProFlow baseline flow and in (b) the RAFT version. The scene is also cropped in order to showcase only the relevant aspects. Similar to the previous example, here, we also see that the outlines of the moving person are darker in the RAFT version than in the ProFlow baseline one. Again, if we look at the EPE results for this scene from Figure 4.6, the outlines of these objects are better depicted with the RAFT flow, contradicting what we see in the SSIM visualization. In this case, we have the same phenomenon of a better flow leading to poorer quality of the interpolation.

This raises the question, why the SSIM results are better for an interpolated image generated with a ProFlow baseline flow when RAFT provides a better flow estimation. As we have seen in the general comparisons in Section 4.2, in many cases the SSIM and RMSE differ in results, in regards to which flow worked better for the scene. To investigate this, we compare these differences, and see whether the SSIM metric holds as the more accurate one, or whether there are some shortcomings as well.

4.3.2 Investigation of Used Error Metrics

After investigating the SSIM and RMSE values in the previous section, we concluded what scenes seem to work better with a RAFT flow or a ProFlow baseline flow. While the values of error metrics enable an evaluation of the interpolation as a whole, it can be beneficial to examine the individual interpolated images. This also allows to investigate whether the SSIM or RMSE results correspond to the perceived quality of the images. As we have discussed in Section 3.8, the SSIM is a generally accurate metric for determining perceived quality. However, we want to investigate whether this holds in all cases, in order to get answers as to why so many scenes have better results with a ProFlow baseline flow.

SSIM vs RMSE

The first interpolated images we examine are shown in Figure 4.10. Here, we see the cropped version of the interpolation results of *Scene 4* of the KITTI dataset. The interpolation was performed with a forward forward warping method and a 4N weight strategy. In (a) we have the interpolated image generated with the ProFlow baseline flow and in (b) the RAFT counterpart. According to both the SSIM and RMSE, the interpolated image that was generated with a ProFlow baseline flow, is better than the one generated with a RAFT flow. However, looking closely at both versions, we can see more artifacts in the ProFlow baseline image than in the RAFT image. For example, the image generated with the ProFlow baseline flow appears to have some ghosting artifacts at the handle of the car door, as well as some more artifacts at the back of the car. While the RAFT image has some artifacts as well, they are less visible in this case. Meaning here, the RAFT version appears to have much better quality, despite having a worse SSIM as well as RMSE result. Both SSIM and RMSE fail to capture the perceived quality well in this case.

An additional example is given in Figure 4.11. Here, we see two cropped areas of *Scene 3* of the KITTI dataset, using a forward forward warping method and a 4N weight strategy. The interpolated image generated with a ProFlow baseline flow is given in (a) and the one generated with a RAFT flow is given in (b). Here, according to the SSIM, the image generated with a ProFlow baseline flow has a better perceived quality. However, on the car in the top row, we can see some clear ghosting artifacts in this image. Whereas in the RAFT image, these artifacts are much less visible and overall the car appears less visibly distorted. In the bottom row, the pole of the green light also is much more distorted in the ProFlow baseline image than in the RAFT one. Interestingly, this time the RMSE yields a better result for the RAFT image. Meaning the RMSE corresponds better to the perceived quality in this instance, making the RMSE more suitable than the SSIM.

Another such instance where the RMSE generally corresponds better to the perceived quality is given in Figure 4.12. Here, a forward forward warping interpolation of scene *Cave 2* with

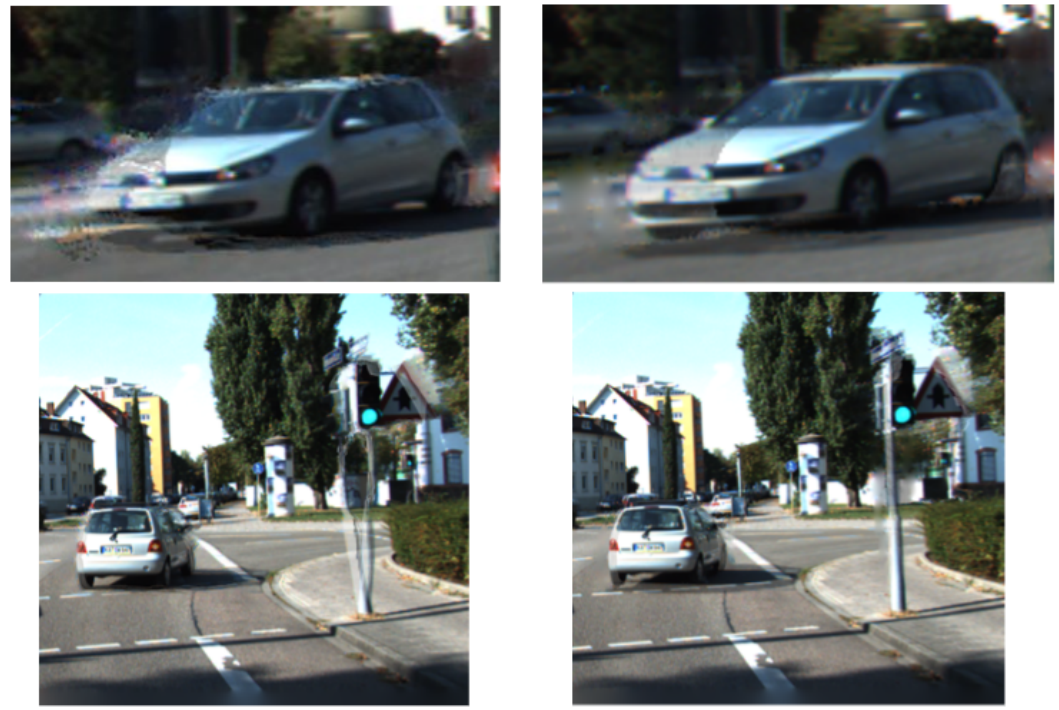
(a) ProFlow Baseline with **SSIM 0.809, RMSE 32.378**

(b) RAFT with SSIM 0.805, RMSE 35.350

Figure 4.10: Forward forward warping results showcasing the cropped interpolated image of *Scene 4*, generated with the ProFlow baseline flow (a) and RAFT flow (b). With (a) SSIM=0.809, RMSE=32.378 and showcasing more artifacts around the moving car, and (b) SSIM=0.805, RMSE=35.350 and less artifacts. The better SSIM and RMSE results are symbolized in bold. A 4N weight strategy was used for the interpolation.

splatting strategy 4N weight is pictured. In (a), the cropped interpolated image using a ProFlow baseline flow is shown, and in (b) the cropped image generated with a RAFT flow can be seen. In this example, the SSIM value of the RAFT image is better than of the image generated with the ProFlow baseline flow. However, the RAFT image generally has much more visible distortion than the ProFlow baseline one. In the RAFT image, we have much more doubling artifacts, as can be seen for example around the blade. In this scene, the RMSE result of the ProFlow baseline image is better and thus the RMSE captures the perceived quality better again.

This raises the question, whether the SSIM is well suited as the primary metric used for the evaluation. Additionally, as we have seen for the Sintel datasets, the ProFlow basline flows work better than RAFT flows, despite RAFT providing better optical flow estimations. Since the SSIM does not always evaluate the quality accurately, this may be the reason why so many RAFT results are worse, when we expected them to be better. Hence why an alternative metric might be sensible. Since the RMSE is not always optimal as well, as seen in Subsection 3.8.1 of Chapter 3, another metric is needed. In Chapter 3, a metric called VMAF was already shown in Subsection 3.8.3. In the following, we make use of it to see whether it is better suited than the metrics used so far.



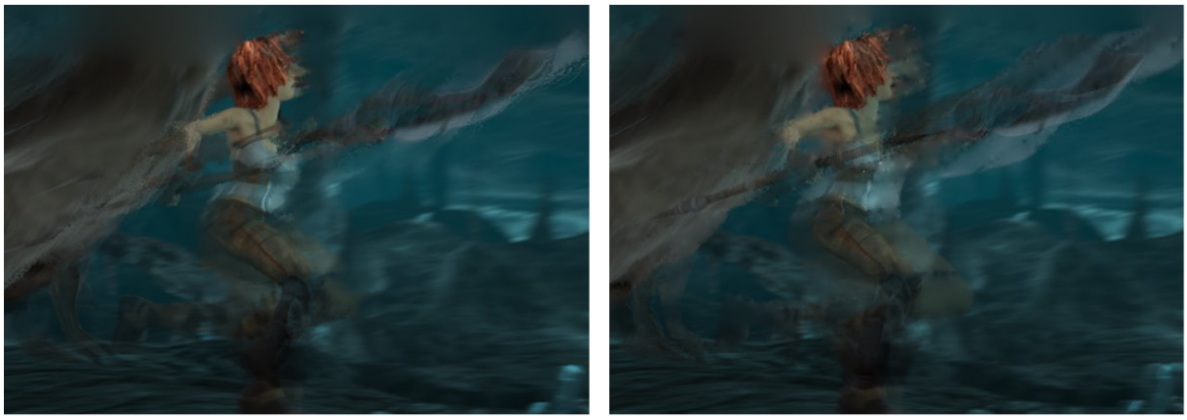
(a) ProFlow Baseline with **SSIM 0.530,** RMSE 42.973

(b) RAFT with SSIM 0.520, **RMSE 42.315**

Figure 4.11: Forward forward warping results showcasing the cropped interpolated image of *Scene 3* generated with the ProFlow baseline flow (a), and RAFT flow (b). With (a) SSIM=0.530, RMSE=42.973 and more artifacts around the car and pole, and (b) SSIM=0.520, RMSE=42.315 and less visible artifacts. The better SSIM and RMSE results are symbolized in bold. A 4N weight strategy was used for the interpolation.

VMAF: Implementation Details

Since VMAF is a video metric, we first need to create a video for the evaluation. For this, the first frame I_0, the interpolated image, and the third frame I_1 were added to form a video with a 20 fps rate. For the interpolated frame, the forward forward warping method was used with a 4N weight strategy, since these produce generally the best results. As a reference video, the first frame I_0, the ground truth image, and the third frame I_1 were put together to form a 20 fps video as well. Currently, there exist different models with which the VMAF framework gets trained. For the calculations shown in the following, the default model was used.



(a) ProFlow Baseline with SSIM 0.342, **RMSE 18.964**

(b) RAFT with **SSIM 0.347**, RMSE 19.036

Figure 4.12: Forward forward warping results showcasing the cropped interpolated image of *Cave 2* generated with the ProFlow baseline flow (a), and RAFT flow (b). With (a) SSIM=0.342, RMSE=18.964 and less doubling artifacts, and (b) SSIM=0.347, RMSE=19.036 and more doubling artifacts. The better SSIM and RMSE results are symbolized in bold. A 4N weight strategy was used for the interpolation.

VMAF: Comparing Results to Other Metrics

Table 4.11 shows the VMAF results for the Middlebury dataset. For reference, the SSIM and RMSE results are also provided. As we can see, for most scenes, the VMAF metric seems to correspond to the SSIM. This is primarily the case for the scenes where the RMSE and SSIM correspond as well. For scenes where the SSIM and RMSE results differ, such as scenes *Army, Evergreen* and *Mequon*, the VMAF result corresponds to the RMSE result. Except for scene *Army*, where the VMAF result is the same for RAFT and the ProFlow baseline. Interestingly, the difference between the RAFT and the ProFlow baseline SSIM and RMSE results in the scenes *Dumptruck* and *Mequon* is one of the largest. However, the VMAF result for these scenes is the same for both RAFT and the ProFlow baseline. On the other hand, for scene *Evergeen*, the VMAF results for RAFT and the ProFlow baseline have the largest difference between them, while the SSIM results are quite close together. For the average results, all three metrics seem to correspond. Overall, in the Middlebury dataset VMAF seems to correspond more to the RMSE results.

In Table 4.12, the same VMAF results are shown for the dataset KITTI. Here, we have the same occurrence as in the previous example. In case of a difference in the SSIM and RMSE result, the VMAF corresponds to the RMSE result, as seen in *Scene 3*. However, in *Scene 1*, VMAF does not correspond to either. As mentioned in the previous Section, *Scene 1* contains

		Forward Forward Warping - Middlebury															
		Army		Backyard		Basketball		Dumptruck		Evergreen		Grove		Mequon		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
4N W	VMAF	**97.430**	**97.430**	75.776	**78.532**	88.959	**91.195**	**97.430**	**97.430**	**82.406**	79.475	52.717	**53.270**	74.158	**74.338**	81.268	**81.667**
	SSIM	0.917	**0.921**	0.771	**0.786**	0.534	**0.541**	0.871	**0.892**	0.815	**0.816**	0.708	**0.738**	**0.820**	0.805	0.777	**0.786**
	RMSE	**5.771**	5.928	18.367	**15.941**	16.541	**15.419**	23.504	**16.564**	**17.834**	18.648	30.892	**29.151**	13.974	**13.053**	18.126	**16.386**

Table 4.11: VMAF results of a forward forward warping interpolation with 4N weight strategy on the Middlebury dataset, using a ProFlow baseline flow and RAFT flow. To calculate the VMAF, a video of the first frame I_0, the interpolated image, and the third frame I_1 was generated with a 20 fps rate. For the ground truth videos, the ground truth image was used as the second frame in the video. For reference, the SSIM and RMSE results are given as well. The winning result is showcased in bold. On the right, the average result of all scenes is given.

		Forward Forward Warping - KITTI											
		Scene 1		Scene 2		Scene 3		Scene 4		Scene 5		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
4N W	VMAF	**91.180**	90.860	**74.434**	71.469	68.611	**69.344**	**85.977**	85.027	**89.433**	82.805	**81.927**	79.901
	SSIM	0.313	**0.314**	**0.313**	0.289	**0.530**	0.520	**0.809**	0.805	**0.623**	0.604	**0.518**	0.506
	RMSE	37.995	**37.575**	**30.057**	32.566	41.059	**39.853**	**32.378**	35.350	**22.727**	24.998	**32.843**	34.068

Table 4.12: VMAF results of a forward forward warping interpolation with 4N weight strategy on the KITTI dataset, using a ProFlow baseline flow and RAFT flow. To calculate the VMAF, a video of the first frame I_0, the interpolated image, and the third frame I_1 was generated with a 20 fps rate. For the ground truth videos, the ground truth image was used as the second frame in the video. For reference, the SSIM and RMSE results are given as well. The winning result is showcased in bold. On the right, the average result of all scenes is given.

mainly camera movement with a lot of trees. This is something that tends to work better with a ProFlow baseline flow. Here, the VMAF metric may correspond more to this assumption. Overall, in the KITTI dataset, the VMAF corresponds more to the RMSE as well.

Lastly, in Table 4.13, the VMAF results from the Sintel dataset are given. Here, we see a similar occurrence as before. In case the RMSE and SSIM differ, the VMAF corresponds to the RMSE, as can be seen in *Cave 2*. For scenes *Alley 1, Sleeping 2* and *Temple 2*, the VMAF results for RAFT and the ProFlow baseline are the same. Scenes *Alley 1* and *Sleeping 2* have the smallest motion and their SSIM results are also quite similar as well. However, *Temple 2* contains rather large motion, and their SSIM and RMSE results are more differing. Here, the VMAF result does not work well. Additionally, scene *Ambush 4* also has a better ProFlow baseline VMAF result, despite RAFT having a better RMSE and SSIM result. This scene had the largest motion and the RAFT optical flow was much better than the ProFlow baseline one. For this scene, the VMAF result does not correspond to this.

Overall, the VMAF metric usually corresponds to the RMSE metric or has the same result for RAFT and the ProFlow baseline in many scenes. Since one of the reasons why we tested another metric was to see whether the RAFT results could get better, using the VMAF metric

		Forward Forward Warping - Sintel											
		Alley 1		Ambush 4		Cave 2		Sleeping 2		Temple 2		∅	
		ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT	ProFlow	RAFT
4N W	VMAF	**97.430**	**97.430**	**61.970**	59.134	**29.598**	25.917	**97.430**	**97.430**	**97.430**	**97.430**	**76.772**	75.468
	SSIM	**0.935**	0.931	0.101	**0.125**	0.342	**0.347**	**0.951**	0.949	**0.864**	0.844	**0.639**	**0.639**
	RMSE	**5.997**	6.432	56.431	**52.996**	**18.694**	19.036	**3.661**	4.041	**18.707**	20.658	20.698	**20.633**

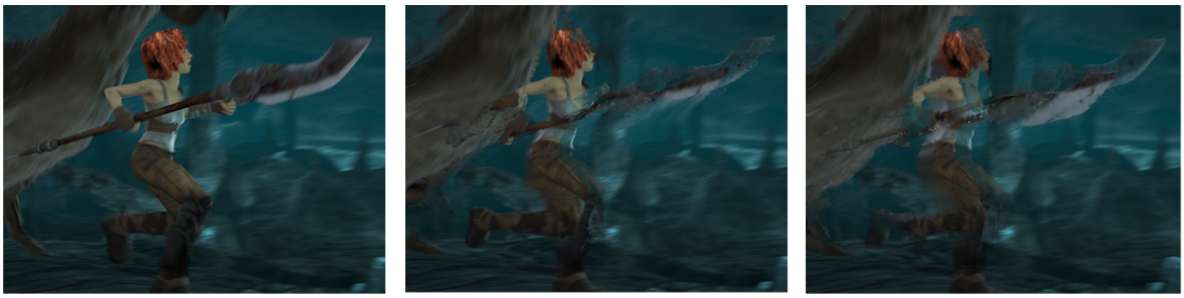
Table 4.13: VMAF results of a forward forward warping interpolation with 4N weight strategy on the Sintel dataset, using a ProFlow baseline flow and RAFT flow. To calculate the VMAF, a video of the first frame I_0, the interpolated image, and the third frame I_1 was generated with a 20 fps rate. For the ground truth videos, the ground truth image was used as the second frame in the video. For reference, the SSIM and RMSE results are given as well. The winning result is showcased in bold. On the right, the average result of all scenes is given.

does not help here. Generally, the RAFT results are even worse in some cases when using the VMAF metric. Additionally, generating the used videos and evaluating them takes additional time. Hence why we are not going to explore this metric any further for the RAFT and the ProFlow baseline experiments, but rather stick to the SSIM and RMSE. So far the focus was on SSIM, however, as we have seen, in some cases the RMSE does work better. For this reason, it can be beneficial to provide the results of both metrics.

4.3.3 Comparing Errors of Interpolated Images

In the last part of the investigation of interpolated images, we examine the most common differences in the images generated with a RAFT flow and a ProFlow baseline flow. However, for scenes with smaller motions, both flows tend to give relatively similar results. For these cases, it is hard to determine which optical flow worked better. Hence we only focus on the most evident and recurring differences.

The most common difference between the interpolated images with RAFT and the ProFlow baseline is that with RAFT the images tend to have more doubling artifacts for large motion, especially in the Sintel dataset. This is due to the fact that the moving objects depicted in image I_0 tend to not be removed in the interpolated image. What happens instead, is that the estimated position of the object in I_t is simply placed over the position of the object in I_0. Such an effect can be seen in Figure 4.13. This figure shows the cropped *Cave 2* scene of the Sintel dataset, with the ground truth image (a) and interpolated images with a ProFlow baseline flow (b) and a RAFT flow (c). For both interpolations, the forward warping method and a 4N strategy were used. Comparing the visual quality of both images, we can see a clear doubling artifact at the blade in the image generated with a RAFT flow. While the position of the blade at time t seems to be estimated quite well, the old position remains in the image as well. This is also visible in the doubling artifacts occurring at the area of the moving person. Such doubling effects are far less visible in the ProFlow baseline image. Here, the moving



(a) Ground Truth Image (b) ProFlow Baseline (c) RAFT

Figure 4.13: Cropped resulting images of a 4N weight forward warping interpolation for *Cave 2*, generated with a ProFlow baseline flow (b), and a RAFT flow (c). With (c) showcasing doubling artifacts round the moving person which are less visible in (b). For reference, the ground truth image (a) is given as well. Image source for (a): [5].

moving person in the scene appears more clear. The blade also does not showcase such clear doubling effect of two different positions.

Another example is seen in the scene Figure 4.14, showcasing two cropped areas of the *Temple 2* scene of the Sintel dataset. Here, we see the forward warping interpolation result with a ProFlow baseline flow (b) and a RAFT flow (c), and the ground truth image (a). For both interpolations, the 4N weight strategy was used. Looking at the RAFT image we can clearly see a doubling effect happening at the wing of the dragon in the bottom image. Although this is the case for the ProFlow baseline image as well, the 'old' position of the wing is much more visible in the RAFT image. Here, the new position of the wing is laid over the old position, creating a doubling effect of two different positions. While both interpolated images show doubling artifacts around the statue in the top image, the tower on the right has no doubling artifacts in the ProFlow baseline one. Here, the tower is depicted almost identically to the ground truth image. Whereas in the RAFT image, this tower has a well-visible doubling effect.

This raises the question, why the interpolated images using a RAFT flow are worse than the ones with a ProFlow baseline flow. As seen in Subsection 4.1.2, the RAFT optical flow for these scenes is better. One would expect the interpolated images to correspond to this, however, this is not the case here.

However, although these doubling effects appear visually bad, in some cases they are a sign that the interpolation worked better. While the images with the ProFlow baseline tend to have less of these effects, what happens is that they also tend to not showcase the motion of objects at the right distance. meaning the objects do not move to the right position in I_t but rather stay at the same position they were in I_0. To showcase this better, we compare the

Figure 4.14: Cropped resulting images of a 4N weight forward warping interpolation for *Temple 2*, generated with a ProFlow baseline flow (b), and a RAFT flow (c). With (c) showcasing more doubling artifacts around the wing and the tower than in (b). For reference, the ground truth image (a) is given as well. Image source for (a): [5].

interpolated images at distance $t = 0.8$ instead of 0.5. This is to see how the effects appear in interpolation that is a little farther away. While we do not have a ground truth image for this position, we can see how far the objects in motion are from the position in I_1. Since we used the distance 0.8, the objects should be closer to the position in I_1.

In Figure 4.15 we can see the cropped interpolated images for the scene *Basketball* at distance $t = 0.8$, generated with a ProFlow baseline flow (c) and a RAFT flow (d). The interpolation was performed with a forward warping and 4N weight strategy. For reference, we also depict the first frame I_0 in (a) and the third frame I_1 in (b). As we can see, the interpolated image with RAFT, again, contains doubling artifacts around the moving objects, such as the ball. However, in this image we can see that the new position was generated for the ball. Whereas in the interpolated image with a ProFlow baseline flow, the ball remains in the same position as in I_0. Meaning the ball has not moved at all, although we interpolated at a distance 0.8 and thus the ball should be closer to I_1. Despite looking visually more pleasing, the interpolated image with the ProFlow baseline is thus less correct in the sense that it does not depict the moving objects at the right position. While the interpolated image with a RAFT flow has more artifacts, here, a movement has at least occurred.

Figure 4.15: Cropped resulting images of a 4N weight forward warping interpolation for *Basketball* at distance $t = 0.8$, generated with a ProFlow baseline flow (c), and a RAFT flow (d). For reference, the first frame I_0 (a) and third frame I_1 (b) are given as well, showcasing how the ball in (c) has not moved at all. Despite the doubling artifacts in (d), the ball has moved to a new position. Image source for (a), (b): [3]

Figure 4.16: Cropped resulting images of a 4N weight forward warping interpolation for *Dumptruck* at distance $t = 0.8$, generated with a ProFlow baseline flow (c), and a RAFT flow (d). For reference, the first frame I_0 (a) and third frame I_1 (b) are given as well, showcasing how the car in (c) has not moved at all. Despite the doubling artifacts in (d), the car has moved to a new position. Image source for (a), (b): [3].

Another example that showcases this phenomenon is depicted in Figure 4.16. Here, a forward warping interpolation is performed on the scene *Dumptruck*, with a ProFlow baseline flow (c) and a RAFT flow (d). This interpolation was performed with a forward warping method and a 4N weight strategy as well. For reference, the first frame I_0 (a) and the third frame I_1 (b) are shown as well. As seen in the I_1 image, the car should be closer to the pole on the right. However, in the ProFlow baseline image the car remains at the same position as in I_0. Meaning here, a ProFlow baseline flow interpolation does not move the car at all. While the car is more visibly distorted in the RAFT image, it has at least moved closer to the pole.

As we can see, interpolated images with RAFT generally have more doubling artifacts due to not correctly removing old positions. However, while this is visually not pleasing, the new positions are still estimated quite well in many cases. Despite the fact that the ProFlow baseline interpolated images appear more visually pleasing, often the movement of objects is not performed at all. This effect may appear in scenes such as *Basketball* or *Dumptruck*. Nonetheless, this phenomenon is not always the case. In scenes such as *Cave 2* and *Temple 2*, the objects do get moved in the images generated with a ProFlow baseline flow. Moreover, they do not showcase as much doubling artifacts as with a RAFT flow. Thus, for these cases, we can say that the ProFlow baseline flow worked better in the interpolation. However, as we have seen in Section 4.1.2, the optical flow for these scenes is better estimated with RAFT. To

investigate why the interpolated images are not better with RAFT, we want to see how 'good' the results of an interpolation can get. For this, we utilize the ground truth flows provided for Sintel in the next section.

4.4 Oracle Experiments

As we have seen in the previous sections, RAFT yields overall a better optical flow estimation than the ProFlow baseline. Nonetheless, performing an interpolation with a RAFT optical flow does not always result in better interpolated images. While RAFT provides a better optical flow estimation, it is not perfect. However, if given the most accurate optical flow, one would expect the interpolation results to yield at least better results than so far. In this section, we test whether this is the case and see how 'good' the results of the interpolation can become if given such a flow. For this, we perform an interpolation with the ground truth flows provided with the Sintel dataset. This experiment provides us with an *oracle* that tells us how 'good' the result of an interpolation method can be.

For the following oracle experiment, we use the forward warping and forward forward warping method with each of the six discussed splatting strategies. So far, we have interpolated at time $t = 0.5$ between the first frame I_0 and the third frame I_1 in order to use the second frame as a ground truth image. However, as mentioned in Section 4.1, the provided ground truth flows only range from one frame to its immediate consecutive frame. Meaning we only have a flow from the first frame to the second frame, or from the second frame to the third frame. Additionally, we also have the backward flows going in the opposite directions. In order to still have a ground truth image to compare the results to, we have to modify the way we perform the interpolation. This is shown in Figure 4.17. For a forward warping method, we interpolate between the first frame and the second frame at distance $t = 1$. Here, we use the ground truth flow, represented as $\overrightarrow{v_{0,f}}$, ranging from the first frame to the second frame. This way, the second frame can remain as the ground truth image. For a forward forward warping method, we perform the first forward warping analogously. For the second forward warping, we warp the third frame I_1 with the backward flow $\overrightarrow{v_{1,b}}$. The backward flow $\overrightarrow{v_{1,b}}$ ranges from the third frame to the second frame. Here, we interpolate at distance $t = 1$ as well. The second frame is then used as the ground truth image. To save space, we refer to the ground truth flows as 'GT', the ProFlow baseline flows as 'PF', and RAFT flows as 'R' in the following tables.

Table 4.14 shows the results of a forward warping interpolation for the Sintel dataset. Analogous to the forward warping results shown in Section 4.2, the SSIM and RMSE results for each scene with each splatting strategy are given here as well. The SSIM and RMSE results using a ground truth flow are depicted on the left, the ProFlow baseline flow in the middle, and a RAFT flow on the right. Additionally, an average is given for each scene as well.

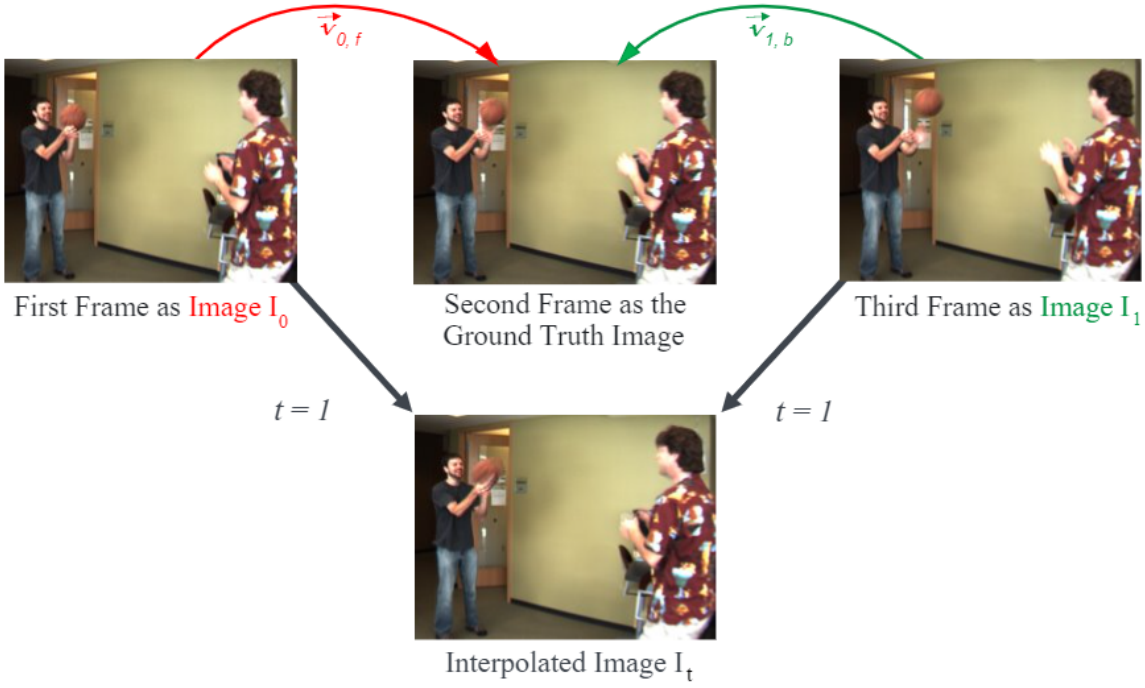


Figure 4.17: Demonstration of how the frames of each dataset are used for interpolation using a ground truth flow. Image source for the used frames: [3].

In Table 4.14, we can see in scenes *Ambush 4* and *Cave 2*, that although using the ground truth flows yields better results than with RAFT or the ProFlow baseline, the results are still not very good. Surprisingly, for scenes *Sleeping 2* and *Temple 2* the results with the ground truth flows are even worse than the ProFlow baseline results. Even in scene *Alley 1* we have worse RMSE results with the ground truth flows. Meaning, even when the 'most accurate' flow is used, we still do not always get a better quality in the interpolation.

In the forward forward warping results provided in Table 4.15, we can see the same phenomenon. *Ambush 4* and *Cave 2* still do not provide good results with the ground truth flow. Furthermore, scenes *Sleeping 2* and *Temple 2* still have better results when a ProFlow baseline flow is used. Additionally, scene *Alley 1* also yields worse RMSE results when the ground truth flow is used. This further illustrates that a better flow quality does not result in better interpolation quality for all cases.

As we can see from these two tables, in some cases the images generated with a ground truth flow yield even worse results than the ones generated with a ProFlow baseline flow. To rule out the possibility that the confusing results might be due to a bad metric, we additionally

		Forward Warping - Sintel																	
		Alley 1			Ambush 4			Cave 2			Sleeping 2			Temple 2			∅		
		GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R
1N U	SSIM	**0.904**	0.893	0.885	**0.223**	0.083	0.088	**0.607**	0.330	0.348	0.903	**0.906**	0.902	0.823	**0.833**	0.815	**0.692**	0.609	0.608
	RMSE	10.595	**9.843**	10.437	**63.643**	69.711	66.278	**17.324**	20.642	20.474	7.607	**6.559**	7.392	28.745	**24.198**	28.063	**25.583**	26.191	26.529
1N W	SSIM	**0.904**	0.893	0.885	**0.223**	0.083	0.088	**0.607**	0.329	0.348	0.903	**0.906**	0.902	0.823	**0.833**	0.815	**0.692**	0.609	0.608
	RMSE	11.010	**9.708**	10.579	**63.661**	69.722	66.235	**17.369**	20.658	20.509	7.604	**6.555**	7.383	29.009	**23.915**	28.090	**25.731**	26.112	26.559
1N WW	SSIM	**0.902**	0.892	0.884	**0.219**	0.082	0.088	**0.593**	0.325	0.342	0.902	**0.906**	0.902	0.822	**0.832**	0.813	**0.688**	0.607	0.606
	RMSE	13.691	**10.411**	12.102	**64.336**	70.149	67.232	**19.080**	21.516	21.957	7.642	**6.569**	7.463	31.452	**24.333**	30.346	27.240	**26.596**	27.820
4N U	SSIM	**0.920**	0.906	0.899	**0.236**	0.085	0.093	**0.621**	0.356	0.374	0.885	**0.887**	0.883	0.821	**0.830**	0.814	**0.697**	0.613	0.613
	RMSE	9.076	**8.656**	9.326	**63.237**	69.689	66.065	**16.980**	20.158	20.180	7.054	**6.791**	7.227	28.453	**23.692**	27.849	**24.960**	25.797	26.129
4N W	SSIM	**0.940**	0.926	0.918	**0.235**	0.084	0.092	**0.640**	0.349	0.367	0.940	**0.943**	0.939	0.850	**0.859**	0.841	**0.721**	0.632	0.631
	RMSE	8.589	**7.992**	8.834	**63.385**	69.813	66.131	**16.985**	20.325	20.276	4.999	**3.913**	5.120	28.184	**23.378**	27.578	**24.428**	25.084	25.588
4N WW	SSIM	**0.901**	0.891	0.883	**0.218**	0.080	0.087	**0.592**	0.324	0.341	0.902	**0.906**	0.901	0.822	**0.831**	0.813	**0.687**	0.606	0.605
	RMSE	13.689	**10.427**	12.160	**64.494**	70.655	67.373	**19.221**	21.723	22.150	7.618	**6.589**	7.530	31.632	**24.464**	30.569	27.331	**26.772**	27.956

Table 4.14: SSIM and RMSE results of a forward warping interpolation of the Sintel dataset, using a ground truth flow, the ProFlow baseline flow, and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, the ground truth flows are referred to as 'GT', the ProFlow baseline flows as 'PF', and RAFT flows as 'R'. On the right, the average of all scenes is given for each splatting strategy.

		Forward Forward Warping - Sintel																	
		Alley 1			Ambush 4			Cave 2			Sleeping 2			Temple 2			∅		
		GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R
1N U	SSIM	**0.942**	0.933	0.928	**0.246**	0.103	0.125	**0.556**	0.334	0.336	0.952	**0.956**	0.952	0.829	**0.858**	0.837	**0.705**	0.637	0.636
	RMSE	7.060	**6.207**	6.697	**50.460**	56.215	53.282	**16.852**	18.707	19.085	4.276	**3.330**	4.036	23.160	**19.007**	20.994	**20.362**	20.693	20.819
1N W	SSIM	**0.943**	0.933	0.929	**0.246**	0.103	0.125	**0.555**	0.334	0.336	0.952	**0.956**	0.952	0.830	**0.858**	0.838	**0.705**	0.637	0.636
	RMSE	7.164	**6.162**	6.702	**50.517**	56.209	53.373	**16.893**	18.706	19.099	4.274	**3.329**	4.030	23.353	**18.912**	21.122	**20.440**	20.664	20.865
1N WW	SSIM	**0.942**	0.932	0.928	**0.240**	0.100	0.121	**0.537**	0.331	0.331	0.952	**0.956**	0.952	0.830	**0.857**	0.836	**0.700**	0.635	0.634
	RMSE	8.246	**6.499**	7.194	**51.536**	56.500	54.999	**17.787**	19.022	19.678	4.364	**3.343**	4.088	25.515	**19.231**	23.055	21.490	**20.919**	21.803
4N U	SSIM	**0.936**	0.925	0.921	**0.257**	0.103	0.126	**0.567**	0.348	0.352	0.913	**0.914**	0.911	0.821	**0.847**	0.828	**0.699**	0.627	0.628
	RMSE	6.493	**6.162**	6.557	**49.742**	56.335	52.916	**16.525**	18.579	18.969	5.293	**5.113**	5.400	22.688	**18.852**	20.796	**20.149**	21.008	20.928
4N W	SSIM	**0.948**	0.935	0.931	**0.259**	0.101	0.125	**0.582**	0.342	0.347	0.949	**0.951**	0.949	0.840	**0.864**	0.844	**0.716**	0.639	0.639
	RMSE	6.224	**5.997**	6.432	**49.816**	56.431	52.996	**16.452**	18.694	19.036	4.084	**3.661**	4.041	22.413	**18.707**	20.658	**19.798**	20.698	20.633
4N WW	SSIM	**0.945**	0.932	0.928	**0.240**	0.094	0.117	**0.541**	0.329	0.332	0.953	**0.957**	0.952	0.836	**0.860**	0.839	**0.703**	0.634	0.634
	RMSE	7.655	**6.526**	7.195	**51.519**	57.191	54.859	**17.460**	19.298	19.828	4.162	**3.287**	4.058	24.790	**19.441**	23.062	**21.118**	21.149	21.800

Table 4.15: SSIM and RMSE results of a forward forward warping interpolation of the Sintel dataset, using a ground truth flow, ProFlow baseline flow, and RAFT flow. For each splatting strategy, the better result is marked bold. Here, the splatting strategy 'U' stands for uniform, 'W' for weight, and 'WW' for weight winner. Additionally, the ground truth flows are referred to as 'GT', the ProFlow baseline flows as 'PF', and RAFT flows as 'R'. On the right, the average of all scenes is given for each splatting strategy.

consider the VMAF results. Although the VMAF has not delivered any new insight in Subsection 4.3.2, we examine it nonetheless in order to see whether it brings more insight when a ground truth flow is used. For this, we take the forward warping results using a 4N weight strategy and create the same 20 fps videos as described in Subsection 4.3.2.

Such VMAF results can be seen in Table 4.16. For reference, we also add the corresponding SSIM and RMSE results in the table. In this table, we can see the VMAF results for the ground truth flow are still not better. On the contrary, here, even scene *Ambush 4* is better with RAFT according to the VMAF metric. For scenes *Alley 1* and *Sleeping 2* the VMAF results are the same for all three flows, providing no further insight. In scene *Temple 2*, the ground truth flow still

		Forward Warping - Sintel																	
		Alley 1			Ambush 4			Cave 2			Sleeping 2			Temple 2			∅		
		GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R	GT	PF	R
4N W	VMAF	**97.430**	**97.430**	**97.430**	56.855	58.732	**58.910**	**41.639**	30.772	25.568	**97.430**	**97.430**	**97.430**	94.966	**97.430**	**97.430**	**77.664**	76.359	75.354
	SSIM	**0.940**	0.926	0.918	**0.235**	0.084	0.092	**0.640**	0.349	0.367	0.940	**0.943**	0.939	0.850	**0.859**	0.841	**0.721**	0.632	0.631
	RMSE	8.589	**7.992**	8.834	**63.385**	69.813	66.131	**16.985**	20.325	20.276	4.999	**3.913**	5.120	28.184	**23.378**	27.578	**24.428**	25.084	25.588

Table 4.16: VMAF results of a forward warping interpolation with 4N weight strategy on the Sintel dataset, using a ground truth flow, a ProFlow baseline flow, and a RAFT flow. To calculate the VMAF, a video of the first frame I_0, the interpolated image, and the third frame I_1 was generated with a 20 fps rate. For the ground truth videos, the ground truth image was used as the second frame in the video. For reference, the SSIM and RMSE results are given as well. The winning result is showcased in bold. On the right, the average result of all scenes is given. Here, the ground truth flows are referred to as 'GT', the ProFlow baseline flows as 'PF', and RAFT flows as 'R'. On the right, the average of all scenes is given for each splatting strategy.

has not the better result. Overall, the VMAF metric does not always yield better results for the ground truth flows. Due to this, we are going to only focus on the SSIM and RMSE again.

Generally, we can see that using a ground truth flow does not give us much better results. In cases such as *Alley 1, Sleeping 2* and *Temple 2*, it actually yields worse results than we have seen so far with the ProFlow baseline flow. To investigate this further we examine the scenes where the ground truth flow provided worse results. Since the scenes *Alley 1* and *Sleeping 2* have quite a small motion, the resulting interpolated images do not provide many differences. Because of this, from now on we focus only on the scene *Temple 2*, which provides a much larger motion with bigger differences between the used flows. Based on this scene we investigate in the following section, in which areas using a ground truth flow yields such bad results and what the reason for this could be.

4.5 Investigation of Ground Truth Results

In this section, we focus on the reason why the interpolated images created with a ground truth flow yield worse results than with the other flow estimations. As mentioned before, we focus only on the scene *Temple 2* throughout this investigation, since it provides the largest motion. In this scene, we only consider the parts of the interpolated images that showcase the biggest differences. Figure 4.18 showcases these most interesting areas of this scene which are investigated in the following sections.

In the investigation, we first examine the resulting interpolated images to see what errors are made with the ground truth flow. To investigate further, we then try to smooth the outlines in a ground truth flow with a Gaussian filter. As we have seen in Section 4.1, the main difference between RAFT and the ProFlow baseline is that in a ProFlow baseline flow the outlines appear

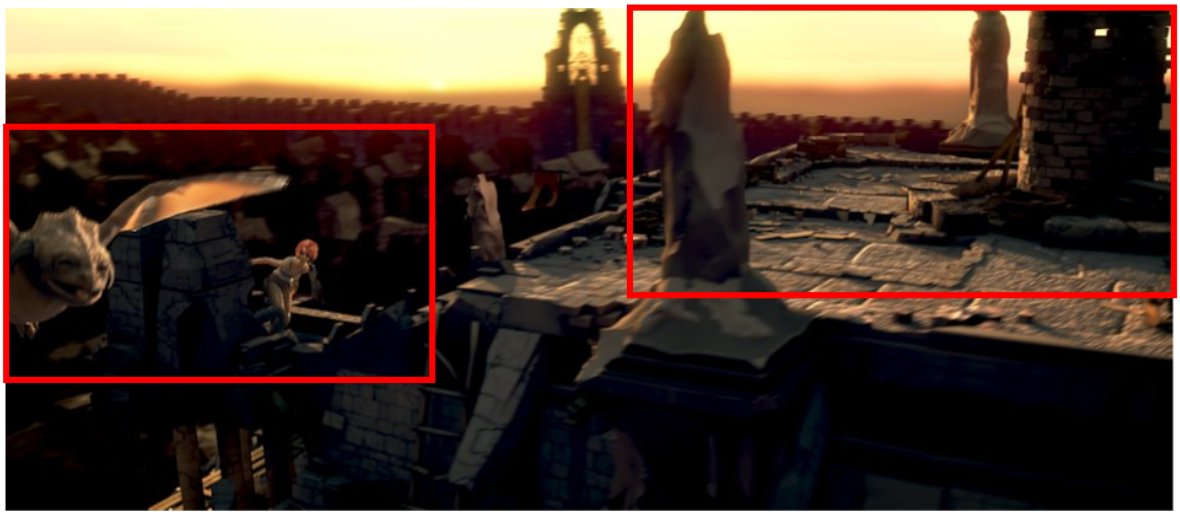


Figure 4.18: The ground truth image of *Temple 2*, showcasing the cropped areas (marked red) which are investigated in the following sections. Image source: [5].

more blurred. Since the ProFlow baseline yields better interpolation results, an idea would be to see whether smoothing the ground truth flow with a Gaussian filter can have a similar effect and yield better interpolation results. Following this, we examine whether the used inpainting method might be the reason causing such errors. For this, we perform the forward warping interpolation without an inpainting method and compare the results. Lastly, we make use of the disocclusion maps shown in Section 3.7. With these, we calculate the SSIM and RMSE results only in areas that are not disoccluded to see whether this yields better ground truth results.

4.5.1 General Problem

First, we examine the interpolated images generated with a forward warping. Since the 4N weight strategy yields the best results, we focus on the results generated with this strategy. The cropped interpolated images showcasing the most interesting aspects can be seen in Figure 4.19. In (a) we can see the image generated with the ground truth flow, in (b) the one created with a ProFlow baseline flow, and the RAFT counterpart is seen in (c). In Subsection 4.3.3, we have already seen that images created with a RAFT flow tend to have doubling effects due to not removing the old position of objects from frame I_0. Such an effect is even more visible in the images generated with the ground truth flow. Here, we can clearly see an overlaying of two different positions in the dragon, shown in the bottom image, or the towers, shown in the top image. As was the case with RAFT, here, the old position of the moving objects does not

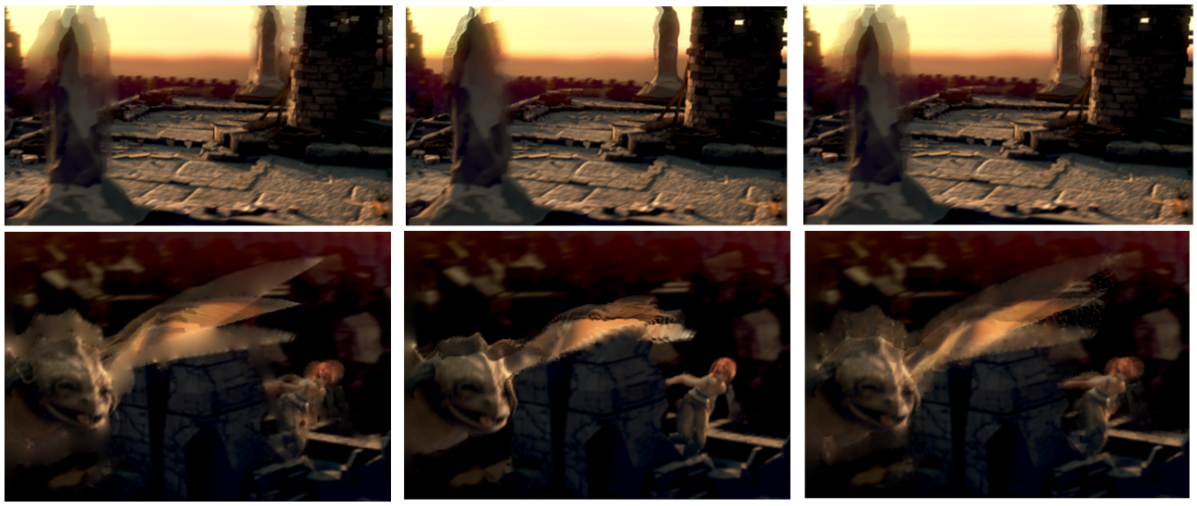
(a) Ground Truth (b) ProFlow Baseline (c) RAFT

Figure 4.19: Cropped resulting images of a 4N weight forward warping interpolation for *Temple 2*, generated with a ground truth flow (a), the ProFlow baseline flow (b), and a RAFT flow (c). In (a) and (c) doubling artifacts showcasing two different positions of the dragon and towers can be seen, which are less visible in (b).

get removed. The new position merely gets placed over it. While such an effect can be seen in some places in the image generated with the ProFlow baseline, such as the statue in the top image, these effects are much less visible. In the case of the right tower shown in the top image, such an effect cannot be seen at all.

This effect can be seen in the forward forward warping results as well. In Figure 4.20, we can see these results generated with the ground truth flow (a), the ProFlow baseline flow (b), and RAFT flow (c). Here, a 4N weight strategy was used as well. Since a forward forward warping warps both frames I_0 and I_1, here, we have three different positions of the moving objects that can be seen in (a) and (c). These are the initial positions of frames I_0 and I_1, and the resulting position of the interpolated image. This is clearly visible in the left tower of the top image or the wings of the dragon in the bottom image.

Generally, we can see that the interpolated images created with a ground truth flow exhibit the same phenomenon as the ones generated with RAFT. In these images, the old positions of moving objects do not get removed, but the new positions are laid on top. Such a phenomenon is almost non-existent in the images generated with the ProFlow baseline flow. As we have seen before, the main difference between these flows are the outlines of moving objects. In a ProFlow baseline flow these outlines appear more blurred and blurred than in a RAFT or ground truth flow. Since the interpolated images generated with the ProFlow baseline flows do not contain the overlaying effects, an idea to fix these effects is to imitate this



(a) Ground Truth (b) ProFlow Baseline (c) RAFT

Figure 4.20: Cropped resulting images of a 4N weight forward forward warping interpolation for *Temple 2*, generated with a ground truth flow (a), the ProFlow baseline flow (b), and a RAFT flow (c). In (a) and (c) doubling artifacts showcasing three different positions of the dragon and towers can be seen, which are less visible in (b).

smoothing of outlines in the ground truth flows. This is done in the following with a Gaussian filter.

4.5.2 Smoother Outlines with a Gaussian Filter

The reason for the doubling effect with the better flows can be that in case the sharp outlines of moving objects do not perfectly align with the outlines in the flow, these areas might be 'dragged' to a new position in the interpolation. In case the cut of the outlines is not as sharp and the outlines of the flows do not match with the outlines of the moving objects, it may not cause them to be moved as drastically. Hence why the more blurred outlines in a ProFlow baseline flow might not have such doubling effects as the sharp outlines in a ground truth or RAFT flow. To test this theory we try emulate this smoothing effect of the outlines in the ground truth flows by adding a Gaussian filter to them. For this, we use a Gaussian filter with $\sigma = 1$ and $\sigma = 5$ and perform a forward warping method with these smoothed flows. Since the main focus was on the 4N weight strategy so far because it provides the best results, in the following, we focus only on this strategy as well.

In Figure 4.21 we see the cropped interpolated images using the ground truth flow without a Gaussian filter (a), with an applied Gaussian filter with $\sigma = 1$ (b), and $\sigma = 5$ (c). As

Figure 4.21: Cropped resulting images of a 4N weight forward warping interpolation for *Temple 2*, generated with a ground truth flow without a Gaussian filter (a), with a Gaussian filter with $\sigma = 1$ (b), and $\sigma = 5$ (c). With (a) showing doubling artifacts on the dragon and towers which are less visible in (b) and not visible at all in (c).

		Forward Warping - Ground Truth Flow		
		Temple 2		
		Without Gaussian Filter	Gaussian Filter with $\sigma = 1$	Gaussian Filter with $\sigma = 5$
4N W	SSIM	**0.850**	0.336	0.281
	RMSE	**28.184**	43.982	49.846

Table 4.17: SSIM and RMSE results of a 4N weight forward warping interpolation of *Temple 2*, using a ground truth flow without a Gaussian filter, with a Gaussian filter and $\sigma = 1$ and $\sigma = 5$.

we can see, the doubling effects do in fact disappear completely when a Gaussian filter is used. Even with $\sigma = 1$, a large reduction of these doubling effects can already be seen. With $\sigma = 5$, these effects are completely gone. So the above mentioned theory does hold. However, we also want to achieve good SSIM and RMSE results. Hence why we evaluate each shown interpolated image with the SSIM and RMSE.

This is shown in Table 4.17. Looking at the results of the applied Gaussian filter, we can see a drastic decline in the SSIM and RMSE results. The results for both $\sigma = 1$ and $\sigma = 5$ are not good. Thus, using a Gaussian filter is not a solution.

As we have seen, while the Gaussian filter eliminates the doubling effects, it causes the SSIM and RMSE results to suffer. What happens here is similar to what we have seen in Section 4.3.3. While we have fewer doubling effects with the Gaussian filter, the motion of the

(a) Ground Truth (b) ProFlow Baseline (c) RAFT

Figure 4.22: Cropped resulting images of a 4N weight forward warping interpolation for *Temple 2* without an inpainting step, generated with a ground truth flow (a), the ProFlow baseline flow (b), and a RAFT flow (c). The holes are marked red.

objects is not estimated as accurately. Meaning the moving objects in the scene are not moved far enough in the interpolated image. Instead, they mostly remain in the position of the first frame I_0. This causes bad SSIM and RMSE results since the motion was not estimated well in the interpolation. Thus, a smoothing of the ground truth flow does not work for our problem. An alternative is to inspect at the underlying inpainting method. To test this, we leave out the inpainting step and compare the interpolated images containing holes. This is done in the following.

4.5.3 Interpolation Without Inpainting

One possibility for causing the doubling effects in the interpolated images generated with the ground truth and RAFT flow can be the underlying inpainting method, discussed in Section 3.4. If we apply a forward warping without an inpainting step at the end and then visualize the resulting holes with a red color, we get the following images shown in Figure 4.22. The shown cropped interpolated images were generated with a 4N weight strategy.

As we can see, the visualized holes in the images correspond to the old positions that do not get removed in the interpolation. Hence why the underlying inpainting method may be the reason why the old positions do not get removed. To test this, we calculate the SSIM and RMSE for an interpolated image containing holes. The results for this are shown in Table 4.18. Additionally, the amount of holes for each interpolated image is given as well.

		Forward Warping Without Inpainting		
		Temple 2		
		Ground Truth	ProFlow	RAFT
1N U	SSIM	0.739	0.750	**0.753**
	RMSE	63.070	59.413	**58.938**
	Amount of Holes	28278	24495	**23593**
1N W	SSIM	0.739	0.750	**0.753**
	RMSE	63.185	59.299	**58.950**
	Amount of Holes	28278	24495	**23593**
1N WW	SSIM	0.739	0.749	**0.751**
	RMSE	64.323	**59.472**	60.050
	Amount of Holes	28278	24495	**23593**
4N U	SSIM	**0.842**	0.822	0.808
	RMSE	54.235	**43.790**	48.766
	Amount of Holes	19126	**10749**	13716
4N W	SSIM	0.842	**0.851**	0.807
	RMSE	54.047	**43.623**	48.612
	Amount of Holes	19126	**10749**	13716
4N WW	SSIM	0.815	**0.823**	0.807
	RMSE	55.882	**44.215**	50.368
	Amount of Holes	19126	**10749**	13716

Table 4.18: SSIM and RMSE results of a forward warping interpolation of *Temple 2* without an inpainting step, using a ground truth flow, the ProFlow baseline flow, and RAFT flow. For each splatting strategy, the better result is marked bold. Additionally, the amount of holes is given for each strategy as well.

Looking at the SSIM and RMSE results, we can see that the images generated with the ground truth flow still do not provide better results. Although a 4N uniform strategy results in the the ground truth flow having the best SSIM results, for the rest of the strategies, the SSIM is still not better with a ground truth flow. If we compare the number of holes, we see that the images created with the ground truth flow contain the highest amount of holes. This may explain why the SSIM and RMSE results are not as good.

Overall, not using an inpainting method does not yield better results for the images generated with the ground truth flow. In fact, they yield even worse results due to the high amount of holes. An alternative would be to investigate other inpainting solutions, however, this is outside the scope of this thesis. As we have seen, the holes appear on the areas of the 'old' position of the moving objects. Such positions are areas that get disoccluded in the interpolated frame. Hence why in the following we examine these areas a little further.

4.5.4 Disoccluded Areas

We have seen that the main problem of the doubling effect is that the moving objects at the old positions in frame I_0 do not get removed in the interpolated frame. Parts of these positions

in the first frame I_0 get disoccluded in the second frame, meaning in our interpolated frame. Hence why we do not have a valid optical flow estimation for these areas. In the following, we test whether ignoring these disoccluded areas in the SSIM and RMSE evaluation yields better results for the images generated with the ground truth flow. We have already discussed how to generate occlusion and disocclusion maps in Section 3.7. Now we make use of such maps to assess which areas get disoccluded in the interpolated image, meaning in the second frame. Since we are only interested in the disoccluded areas we only use the disocclusion map. For this, Algorithm 3.6 was used with the threshold $T = 1$. Since we want to obtain a disocclusion map for each of the flow estimations separately, the forward-backward check was used for the ground truth flows, the ProFlow baseline flows and RAFT flows separately. For the disocclusion map of the ground truth flow, the forward flow ranging from the first frame to the second frame, and the backward flow ranging from the second frame to the first frame was used. For RAFT and the ProFlow baseline, the forward flow ranging from the first frame to the third frame and the backward flow ranging from the third frame to the first frame was used.

Figure 4.23 shows these disocclusion maps generated with the ProFlow baseline flows (a), RAFT flows (b) and the ground truth flows (c). As we can see in (c), the white parts representing the disoccluded areas correspond to the 'old' positions from frame I_0 that do not get removed in the interpolated image with the ground truth flow. We now we take the interpolated images generated with a forward warping and calculate the SSIM and RMSE only for the pixels that are not disoccluded. For this, the used SSIM method has to be modified in order to calculate the SSIM for each pixel. This was already done in Subsection 4.3.1. After calculating the SSIM for each pixel, we only consider the pixels that are not disoccluded and average over their SSIMs to obtain one resulting SSIM value. Analogously, for the RMSE we only calculate the squared difference between the pixels that are not disoccluded.

Table 4.19 shows the SSIM and RMSE results for each splatting strategy. Overall, the SSIM and especially the RMSE results are much better when the disoccluded areas are not considered in the evaluation. However, as we can see, the images generated with the ProFlow baseline flows still provide better results than the ones generated with the ground truth flows. Meaning even when the 'old' positions are removed in the interpolated image, the ProFlow baseline images are still better according to the SSIM and RMSE.

Overall, using disocclusion maps to leave out disocclusions in the interpolated images, provided much better results. However, these results still indicate that the interpolated images generated with the ground truth flows are not as good as the ones generated with the ProFlow baseline. Meaning that even if we use an accurate flow estimation, the results do not always reflect this. Thus, for the conducted experiments, a better flow quality did not automatically correspond to better interpolation results.

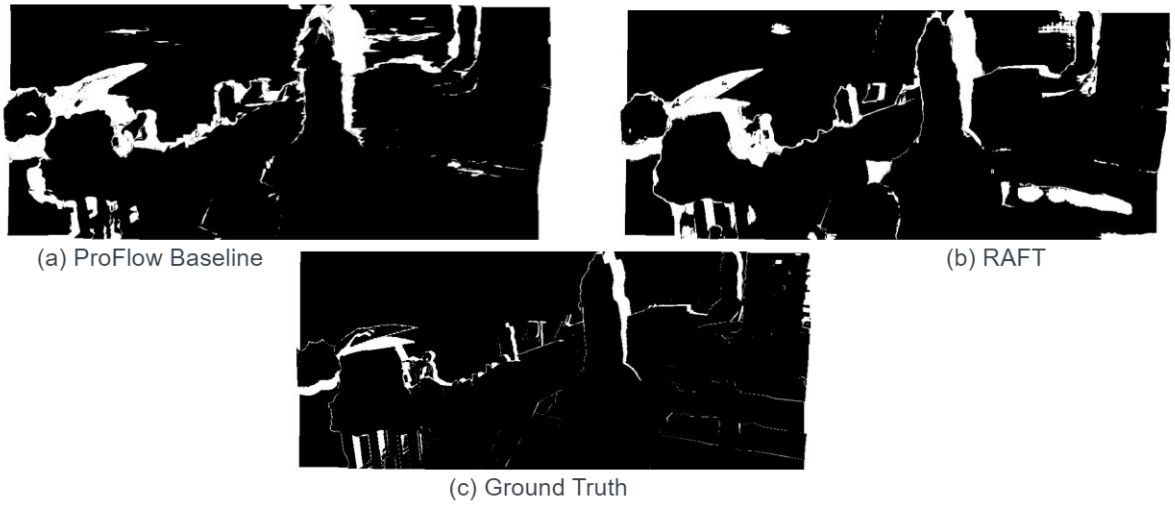


Figure 4.23: Disocclusion maps for *Temple 2*, generated with the ProFlow baseline flows (a), RAFT flows (b), and ground truth flows (c).

		Forward Warping Without Disocclusions		
		Temple 2		
		Ground Truth	ProFlow	RAFT
1N U	SSIM	0.854	**0.878**	0.860
	RMSE	10.253	**8.559**	9.922
1N W	SSIM	0.854	**0.879**	0.860
	RMSE	10.483	**8.396**	10.043
1N WW	SSIM	0.854	**0.877**	0.858
	RMSE	12.378	**8.637**	11.938
4N U	SSIM	0.844	**0.865**	0.849
	RMSE	9.932	**8.433**	9.747
4N W	SSIM	0.877	**0.901**	0.883
	RMSE	9.565	**8.030**	9.434
4N WW	SSIM	0.854	**0.877**	0.857
	RMSE	12.295	**8.603**	11.999

Table 4.19: SSIM and RMSE results of a forward warping interpolation of *Temple 2* calculated only on areas without disocclusions, using a ground truth flow, the ProFlow baseline flow, and RAFT flow. For each splatting strategy, the better result is marked bold.

4.6 Summary of Experiments

Throughout this chapter, we have conducted several experiments to determine whether the quality of an optical flow estimation played a role in the quality of interpolation results. To recapitulate on the conducted results, we summarize the main points of these experiments.

First, we compared the visualization of optical flows estimated with RAFT and the ProFlow baseline in order to determine their main differences. Here, we have seen that the biggest difference were the outlines of moving objects. While RAFT captures very clean-cut and sharp outlines of moving objects, the ProFlow baseline flows have more blurred outlines. On the other hand, the ProFlow baseline seems to capture more textured areas such as trees, a little better. Overall, an investigation of the EPE showed that RAFT provided generally better optical flow estimations. Here, we also assessed each pixel individually and found that the RAFT flows had better EPE results for the outlines of moving objects.

The next step was to compare the interpolation results using a ProFlow baseline and RAFT flow. Here, we looked at the 'winner' of various interpolation methods, with the focus being on the forward warping and forward forward warping. While the Middlebury dataset had overall more RAFT 'winners', the ProFlow baseline did a better job for the KITTI and Sintel datasets. The main difference in these datasets is that Middlebury captures primarily moving objects in a static scene, while KITTI contains mostly camera movement. From this, we conducted that RAFT seems to work better for sequences with larger motion and without any camera movement. The ProFlow baseline flows, on the other hand, provided better interpolation results for scenes with textured areas where the outlines are not as easily differentiated from the background. This corresponded to what was seen in the optical flow estimations. The Sintel dataset is mainly synthetic. For scenes with a very large motion, RAFT worked better, while for scenes with a small motion, the ProFlow baseline had better results. However, here, the optical flow quality did not correspond to the quality of the interpolated images. Most RAFT optical flows were better for the Sintel dataset, yet the ProFlow baseline had overall more 'winners' in the interpolation.

This was further investigated by calculating the SSIM for each pixel individually and examining which areas seemed to be erroneous. Here, we have seen that while the RAFT optical flows provided much better EPE results for the outlines of moving objects, the SSIM results of the interpolated images were actually worse for these same areas. This led to an investigation of the used error metric SSIM, where we found instances where the RMSE actually worked better for the perceived quality. Due to this, we also explored an alternative metric, the VMAF, to test whether this leads to better RAFT results. Overall, this metric did not provide any more insight and usually corresponded to the RMSE. Hence why the focus then remained on the SSIM and RMSE despite its shortcomings. As a last comparison of the RAFT and the ProFlow baseline interpolation results, we then examined the most prominent errors occurring in the

interpolated images. Here, we found that RAFT consists of many doubling artifacts. Instead of properly removing the moving objects depicted in the initial position of frame I_0, these old positions are kept in the interpolated image. The objects depicted in the 'new' position are then just layered on top, creating these doubling effects. Such a phenomenon was not seen in the interpolated images generated with a ProFlow baseline flow. While this was visually more pleasing, sometimes it also caused the objects to not move at all. Meaning, the objects in the interpolated image remained in the same position as in frame I_0. This was further showcased by interpolating at a farther distance.

After comparing ProFlow and RAFT results we then took the ground truth flows for the Sintel dataset and conducted an oracle experiment. Here, we considered the forward warping and forward forward warping interpolation methods. While most scenes had better results when the ground truth flow was used, the actual SSIM and RMSE values were not as good. Furthermore, for some scenes, the ProFlow baseline results were actually better. Even the VMAF metric did not provide any better results.

Because of this, the last part of the experiments consisted of an investigation of possible causes. Here, we first discovered that the doubling effects visible in the RAFT results were also seen in the images generated with a ground truth flow. Since both flows depicted outlines much sharper than the ProFlow baseline flows, the idea was to then smooth the ground truth flow with a Gaussian filter. With these smoothed flows we then applied a forward warping interpolation. While this helped to get rid of the doubling effects, here, we had the problem that the moving objects were not moved in the interpolated images, thus resulting in bad SSIM and RMSE values. Another idea was to perform an interpolation without an inpainting step to see whether the underlying inpainting method caused the problems. However, here, the images generated with a ground truth flow had overall more holes than with RAFT or the ProFlow baseline, hence why the results did not improve. The last attempt to improve the results was to use disocclusion maps to find the disoccluded areas that did not have an accurate flow. Here, we calculated the SSIM and RMSE only for pixels that were not disoccluded. Although the overall results were better, the images generated with a ProFlow baseline flow still yielded better results than the ones generated with the ground truth flow.

After conducting these experiments, the main insight we have gathered here is that the quality of the optical flow quality does not always correspond to the quality of interpolated images using these flows.

5 Conclusion

To answer the question of whether the quality of the used optical flow correlates to the quality of the interpolation results, we investigated two different optical flow estimations, RAFT and ProFlow baseline, and performed several interpolation experiments with them.

Examining the EPE and visualizations of these flows, we have seen that generally, RAFT yields more accurate optical flow estimations with sharper outlines of moving objects. However, usually, these estimations tend to smooth over textured areas, which are thus better depicted with ProFlow baseline. Using these flows in interpolation experiments, we could see that for scenes with less camera movement and with objects that have well visible outlines, the RAFT flow works better. On the other hand, ProFlow baseline flows works better for more camera movement. An expected outcome of the interpolation was that a better flow would result in better interpolation results. However, the majority of the Sintel dataset had better interpolation results with a ProFlow baseline flow, despite RAFT providing more accurate optical flow estimations for most scenes. Thus a 'better' flow did not generate 'better' interpolated images in all cases. Surprisingly, specific areas of a scene where the optical flow estimation was better with RAFT, contained more errors in the resulting interpolated images than the ones generated with ProFlow baseline. This was mainly seen in the outlines of objects, where the SSIM was worse for images generated with RAFT. While the SSIM metric did not always correspond to the perceived quality, it was not the sole reason for these bad results. The main problem that could be seen in these cases are doubling effects around moving objects. This is the result of not removing old the position of the moving object but rather just overlaying the new one on top.

Since RAFT flows have their inaccuracies as well, we wanted to see how well an interpolation can perform with a mostly accurate flow estimation. For this, we used the ground truth flows for the Sintel dataset to conduct an oracle experiment. Here, we have seen that in some cases, the interpolation results are still better when a ProFlow baseline flow was used. Similar to the case of using a RAFT flow, images that were generated with a ground truth flow showcase major doubling effects around moving objects. Both RAFT and the ground truth flows display much sharper outlines of moving objects in the flow visualization. From this, we concluded that the sharper the outlines in an optical flow estimation, the more doubling effects are visible in the interpolation. While smoothing these outlines with a Gaussian filter removed doubling effects, it caused the objects of the scene to not move far enough in the interpolated image, resulting in bad SSIM and RMSE results. By leaving out an inpainting step we could see

that the ‘old’ positions that do not get removed in the interpolation, correspond with the holes of the image. However, calculating the SSIM and RMSE of the images containing holes led to no improvement due to the high amount of holes. These ’old’ positions also correspond to the disocclusions in the second frame. Nonetheless, calculating the SSIM and RMSE for pixels that are not disoccluded did not result in better ground truth results than with ProFlow baseline. This lead to the conclusion that for some cases, a more ‘accurate’ flow does not provide better results in a forward warping or forward warping interpolation.

Future Work

We conducted various experiments to analyze why a ‘better’ optical flow did not always correspond to better interpolation results. For these experiments, we used different sequences from three different benchmarks. While the used datasets contained a variety of sequences, it would be beneficial to look into additional scenes or even other benchmarks to get more data. For the datasets that were used in this thesis, we have seen that in some cases, used error metric did not always correspond to the perceived quality. For this, it might be helpful to examine further error metrics that may provide a more accurate estimation of the perceived quality.

Nonetheless, we have seen that the main problem for the suprising results was not the error metric. To understand why the interpolated images are not better even when a ground truth flow is used, we looked at different approaches, yet there are still many possibilities that can be investigated. In the experiment chapter, we have already touched briefly on the option of using a different inpainting strategy. As we have seen, the ‘old’ positions of image I_0 which do not get removed in the interpolated image, correspond to the holes that have to be inpainted. Using a more suited inpainting method could prevent these holes to be filled with the area of the ‘old’ position and instead fill them with the right information. In this thesis, we only looked at an isotropic, diffusion based approach. The alternative is to look at anisotropic inpainting methods instead [57, 68]. Moreover, the used inpainting method is based on solving a PDE. Here, it would be beneficial to look at alternatives such as convolution filter based methods [69, 70] or synthesis based methods [55, 56] or other alternatives. Furthermore, we only looked at two different ways for splatting and three different strategies that deal with a collision. An option is to use other approaches that were not discussed in this thesis. Additionally, while using a forward warping and forward forward warping interpolation method yielded such worse results, other methods might provide different results. In this thesis, we briefly discussed various other approaches that were not further investigated with the ground truth flows. Conducting the same oracle experiments with these methods could possibly yield other results. Furthermore, using other methods that were not discussed in this thesis might provide more insight as well. All interpolation methods that were discussed use only two frames. Here, it might be useful to utilize methods that look at more than just two frames. Additionally, we

only looked at linear interpolation. In Section 3.6, we saw what challenges this can cause. An alternative would be to look at other approaches such as quadratic interpolation [61] or a bicubic interpolation [71] or other possible methods.

Lastly, we have tried comparing the interpolation results only of areas where we should possess an accurate ground truth flow, namely areas without disocclusion. However, it might be possible that the ground truth flows in these areas are not accurate enough. Here, an idea could be to investigate whether the flow is estimated well for each area of the scene or whether there are some errors that need to be fixed.

Bibliography

[1] Z. Teed and J. Dang, "RAFT: recurrent all pairs field transforms for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 402–419, Springer, 2020. (Cited on pages 3, 5, 10, 11, 16, 18, 19 and 59)

[2] D. Maurer and A. Bruhn, "Proflow: learning to predict optical flow," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018. (Cited on pages 3, 5, 10, 11, 16, 17, 54 and 59)

[3] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *International Journal of Computer Vision*, vol. 92, pp. 1–31, 2011. (Cited on pages 3, 5, 10, 11, 14, 15, 54, 55, 56, 82, 83 and 85)

[4] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015. (Cited on pages 3, 5, 11, 54 and 57)

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)* (A. Fitzgibbon, ed.), Part IV, LNCS 7577, pp. 611–625, Springer-Verlag, 2012. (Cited on pages 3, 5, 11, 20, 21, 46, 48, 54, 58, 80, 81 and 88)

[6] C.-Y. Wu, N. Singhal, and P. Krähenbühl, "Video compression through image interpolation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 416–431, September 2018. (Cited on page 9)

[7] H. Jiang, D. Sun, V. Jampani, M. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9000–9008, 2018. (Cited on page 9)

[8] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, "Jump: Virtual reality video," in *ACM Transactions on Graphics*, vol. 35, pp. 1–13, Association for Computing Machinery, 2016. (Cited on page 9)

[9] A. Karargyris and N. Bourbakis, "Three-dimensional reconstruction of the digestive wall in capsule endoscopy videos using elastic video interpolation," in *IEEE Transactions on Medical Imaging*, vol. 30, pp. 957–971, 2011. (Cited on page 9)

[10] S. Niklaus and F. Liu, “Context-aware synthesis for video frame interpolation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1701–1710, 2018. (Cited on page 9)

[11] T. Thaipanich, P.-H. Wu, and C.-C. J. Kuo, “Low complexity algorithm for robust video frame rate up-conversion (fruc) technique,” in *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 220–228, 2009. (Cited on page 9)

[12] N. B. Senn, *Zwischenbildinterpolation mit Optischem Fluss*. Universität Stuttgart, 2020. http://dx.doi.org/10.18419/opus-11150. (Cited on pages 11, 27, 29, 30, 31, 32, 34, 40, 42 and 54)

[13] J. J. Gibson, *The Perception of the Visual World*. Houghton Mifflin, 1950. (Cited on page 13)

[14] S. Shah and X. Xuezhi, “Traditional and modern strategies for optical flow: an investigation,” *SN Applied Sciences*, vol. 3, 2021. 289. (Cited on page 13)

[15] M. Black, “Computing optical flow: The "good parts" version.” Machine Learning Summer School (MLSS), Tübingen, 2013. (Cited on page 14)

[16] S. Manandhar, “3d flow field visualization for 3d light-sheet microscopy image sequences,” Retrieved From: https://sandeepmanandhar.com/index.php/research/. (Cited on page 15)

[17] B. Li, J. Han, Y. Xu, and K. Rose, “Optical flow based co-located reference frame for video compression,” in *IEEE Transactions on Image Processing*, vol. 29, pp. 8303–8315, 2020. (Cited on page 14)

[18] G. Lu, W. Ouyang, X. Xu, D.and Zhang, C. Cai, and Z. Gao, “Dvc: An end-to-end deep video compression framework,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10998–11007, 2019. (Cited on page 14)

[19] J. Calandre, P. Péteri, and L. Mascarilla, “Optical flow singularities for sports video annotation: Detection of strokes in table tennis,” in *MediaEval’19 Workshop*, 2019. (Cited on page 14)

[20] Y.-H. Tsai, M.-H. Yang, and M. J. Black, “Video segmentation via object flow,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3899–3908, 2016. (Cited on page 14)

[21] G. Desouza and A. Kak, “Vision for mobile robot navigation: a survey,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 237–267, 2002. (Cited on page 15)

[22] N. Ohnishi and A. Imiya, “Featureless robot navigation using optical flow,” in *Connection Science*, vol. 17, pp. 23–46, 2005. (Cited on page 15)

[23] H. Chao, Y. Gu, and M. Napolitano, "A survey of optical flow techniques for robotics navigation applications," in *J Intell Robot Syst*, vol. 73, p. 361–372, 2014. (Cited on page 15)

[24] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance.," in *Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 2, pp. 726–733, 2003. (Cited on page 15)

[25] M. Jain, H. Jégou, and P. Bouthemy, "Better exploiting motion for better action recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2555–2562, 2013. (Cited on page 15)

[26] A. Ranftl, F. Alonso-Fernandez, S. Karlsson, and J. Bigun, "Real-time adaboost cascade face tracker based on likelihood map and optical flow," in *IET Biometrics*, vol. 6, pp. 468–477, 2017. (Cited on page 15)

[27] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," in *Image and Vision Computing*, vol. 21, pp. 359–381, 2003. (Cited on page 15)

[28] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences," in *International Journal of Computer Vision*, vol. 35, p. 295–319, 1999. (Cited on page 15)

[29] G. Ayzel, M. Heistermann, and T. Winterrath, "Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0.1)," in *Geoscientific Model Development Discussions*, pp. 1–23, 09 2018. (Cited on page 15)

[30] J. L. Barron and A. Liptay, "Measuring 3-d plant growth using optical flow," in *Bioimaging*, vol. 5, pp. 82–86, 1997. (Cited on page 15)

[31] M. Michels, *A Beehive Monitoring System Incorporating Optical Flow as a Source of Information*. Freie Universität Berlin, 2011. http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/Betreute_Arbeiten/Bachelorarbeit-Michels.pdf. (Cited on page 15)

[32] M. Khalid, L. Pénard, and E. Mémin, "Application of optical flow for river velocimetry," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 6243–6246, 2017. (Cited on page 15)

[33] M. Abdel-Nasser, A. Moreno, H. Rashwan, and D. Puig, *Analyzing the evolution of breast tumors through flow fields and strain tensors*, vol. 93. 2017. Pattern Recognition Techniques in Data Mining. (Cited on page 15)

[34] T. Weibel, C. Daul, D. Wolf, R. Rösch, and F. Guillemin, "Graph based construction of textured large field of view mosaics for bladder cancer diagnosis," in *Pattern Recognition*, vol. 45, pp. 4138–4150, 2012. (Cited on page 15)

[35] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, p. 674–679, 1981. (Cited on page 15)

[36] A. Bruhn, J. Weickert, and C. Schnörr, "Combining the advantages of local and global optic flow methods," in *Pattern Recognition* (L. V. Goo, ed.), vol. 2449 of Lecture Notes in Computer Science, pp. 454–462, Springer: Berlin, 2002. (Cited on page 15)

[37] B. Horn and B. Schunck, "Determining optical flow," in *Artificial Intelligence*, vol. 17, pp. 185–203, 1981. (Cited on page 15)

[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766, 2015. (Cited on page 16)

[39] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2462–2470, 2017. (Cited on page 16)

[40] A. Ahmadi and I. Patras, "Unsupervised convolutional neural networks for motion estimation," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1629–1633, 2016. (Cited on page 16)

[41] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu, "Occlusion aware unsupervised learning of optical flow," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4884–4893, 2018. (Cited on pages 16 and 28)

[42] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1164–1172, 2015. (Cited on pages 17 and 18)

[43] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patch match for large displacement optical flow," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5704–5712, 2016. (Cited on page 17)

[44] Y. Hu, Y. Li, and R. Song, "Robust interpolation of correspondences for large displacement optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 481–489, 2017. (Cited on page 17)

[45] D. Maurer, M. Stoll, and A. Bruhn, "Order-adaptive and illumination-aware variational optical flow refinement," in *Proceedings of the British Machine Vision Conference*, p. 1–13, 2017. (Cited on page 18)

[46] K. Cho, J. Chung, C. Gulcehre, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning*, 2014. (Cited on page 19)

[47] F. Galton, *Composite Portraits, Made by Combining Those of Many Different Persons into a Single Resultant Figure*, vol. 8. Royal Anthropological Institute of Great Britain and Ireland, 1879. (Cited on page 25)

[48] D. Rowland and D. Perrett, "Manipulating facial appearance through shape and color," in *IEEE Computer Graphics and Applications*, vol. 15, pp. 70–76, 1995. (Cited on page 25)

[49] D. W. Thompson, *On Growth and Form*. Canto, Cambridge University Press, 1942. (Cited on page 25)

[50] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1106–1112, 1997. (Cited on page 25)

[51] Y. Tang and C. Suen, "Image transformation approach to nonlinear shape restoration," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 155–172, 1993. (Cited on page 25)

[52] M.-C. Chiang and T. Boult, "Efficient super-resolution via image warping," in *Image and Vision Computing*, vol. 18, pp. 761–771, 07 2000. (Cited on page 25)

[53] D. Aliaga, D. Yanovsky, T. Funkhouser, and I. Carlbom, "Interactive image-based rendering using feature globalization," in *Proceedings of the Symposium on Interactive 3D Graphics*, 2003. (Cited on page 25)

[54] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5437–5446, 2020. (Cited on page 28)

[55] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 1033–1038. (Cited on pages 40 and 100)

[56] L. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," p. 479–488, 2000. (Cited on pages 40 and 100)

[57] P. Perona and J. Malik, *Scale-space and edge detection using anisotropic diffusion*, vol. 12. 1990. (Cited on pages 40 and 100)

[58] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pp. 417–424, 2000. (Cited on page 40)

[59] L. Hoeltgen, A. Kleefeld, I. Harris, and M. Breuss, "Theoretical foundation of the weighted laplace inpainting," in *Applications of Mathematics*, vol. 64, pp. 281–300, 2019. (Cited on page 40)

[60] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer, "Edge-based compression of cartoon-like images with homogeneous diffusion," in *Pattern Recognition*, vol. 44, pp. 1859–1873, 2011. Computer Analysis of Images and Patterns. (Cited on page 40)

[61] X. Xu, L. Siyao, W. Sun, Q. Yin, and M. Yang, "Quadratic video interpolation," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. (Cited on pages 44 and 101)

[62] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), pp. 438–451, Springer Berlin Heidelberg, 2010. (Cited on page 44)

[63] M. Carnec, P. Le Callet, and D. Barba, "Full reference and reduced reference metrics for image quality assessment," in *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, vol. 1, pp. 477–480, 2003. (Cited on page 48)

[64] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, 2004. (Cited on page 49)

[65] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," in *Netflix TechBlog*, 2016. (Cited on page 51)

[66] Z. Li, "Vmaf: The journey continues." Streaming Media West, 2018, Retrieved From: https://www.youtube.com/watch?v=s-xUrqTztBQ&t=1006s. (Cited on page 52)

[67] Scikit-image library: https://scikit-image.org/docs/dev/api/skimage.html. (Cited on pages 54 and 72)

[68] A. Halim and B. R. Kumar, "An anisotropic pde model for image inpainting," in *Computers Mathematics with Applications*, vol. 79, pp. 2701–2721, 2020. (Cited on page 100)

[69] M. M. Hadhoud, K. Moustafa, and S. Shenoda, "Digital images inpainting using modified convolution based method," in *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 1, p. 1–10, 2008. (Cited on page 100)

[70] H. Noori, S. Saryazdi, and H. Nezamabadi-pour, "A convolution based image inpainting," 2010. (Cited on page 100)

[71] R. Keys, "Cubic convolution interpolation for digital image processing," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, pp. 1153–1160, 1981. (Cited on page 101)

All links were last followed on May 31, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature