Andrej Reichmann

Video Frame Interpolation for Smooth Slow Motion Based on Optical Flow

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dr. Thomas Pock

Institute for Computer Graphics and Vision

Advisor

Dipl.-Ing. Thomas Mauthner

Institute for Computer Graphics and Vision

Graz, Austria, March 2015

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the presented master's thesis dissertation.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

(place/date) (signature)

Abstract

Slow motion is a widely used technique, to unveil and analyze details of movements, which are too fast to be accurately perceived by the human visual system. This deceleration effect is typically achieved with the overcranking technique, where a video is recorded at a high frame rate and played back at a slower speed. In this thesis, concepts for the generation of the slow motion effect, which do not require a high frame rate camera, are investigated. The motion compensated frame interpolation algorithm is a sophisticated basis for the computation of the slow motion effect. It inserts novel frames between every frame pair of the recorded video. These intermediate frames are computed by interpolating moving objects along their apparent motion trajectories. Therefore, the resulting slow motion sequence features smooth movements. In this work, the variational Huber-L1 optical flow algorithm is utilized, to estimate the motion information. It was observed, that in regions where the motion estimation is erroneous, interpolation errors are very likely to become visible. These errors especially occur at very large displacements, small moving structures, occlusions and disocclusions, which all represent weak spots of the variational optical flow model. To improve these deficiencies, a two step artifact removal strategy was developed, which addresses the correction of the optical flow errors, to reduce the visible artifacts. The optical flow result was enhanced, by combining the variational motion estimation model with a patch-based approach. Based on this enhancement, the wrongly interpolated objects are relocated at the proper position, and inpainted with the proposed depth-order preserving algorithm. As shown in the evaluation, the slow motion algorithm yields promising results for a wide range of different scenes and movements. Furthermore, the artifact removal strategy is able to significantly reduce the amount of interpolation errors.

Keywords: Slow Motion, Motion Compensated Frame Interpolation, Bidirectional Interpolation, Optical Flow, Disocclusion-Handling, Inpainting

Acknowledgments

First of all, I would like to express my gratitude to my family for their endless support throughout my whole study. Many thanks go to my thesis advisor Thomas Mauthner for his guidance, numerous problem-solving discussions and for proof-reading my thesis. Furthermore, I am grateful to my supervisor Thomas Pock for his support and inspirational ideas. I also have to thank the whole GPU4Vision group for providing the optical flow library, and especially René Ranftl for explaining details of the parameterization to me. Many thanks also go to Timon, Lara and Helena for helping me to record the ground truth dataset. I would also like to thank my office colleagues Mahdi, Patrick, Robert and Michael for the great time we spent together.

Finally, I would like to thank Daniela for her love and support.

Contents

List of Figures

Chapter 1

Introduction

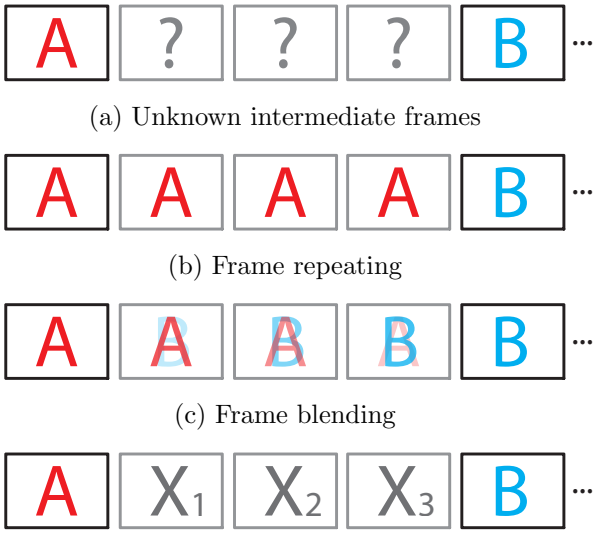
1.1 Motivation and Problem Statement

Slow motion is a commonly used technique, to slow down a certain video and unveil details of an initially fast movement. It is often used in the film industry for instant replays in the field of sports broadcasting or as analytic tool for fast motion sequences, like for example car crash-tests.

For professional applications, the slow motion effect is in general achieved by recording a video at a very high frame rate, and playing it back at a lower speed. This so called *overcranking* approach has no major computational challenges, and is very fast to compute. However, it requires a high speed camera, which is capable of recording videos at the desired frame rate, and handling the high amount of data within short time. These devices are significantly more expensive than regular cameras with recording frame rates of 24 - 30 frames per second (fps), depending on their highest possible recording frame rate and video resolution. Furthermore, they reach physical limits regarding the exposure time, and hence need additional light sources for high slowdowns, to get well exposed frames. Therefore, it is necessary, to find alternative ways for the computation of this slow motion effect, without the need for high speed cameras.

The primary goal of this thesis is to develop an algorithm, which computes a visually attractive slow motion effect, without utilizing the explained overcranking technique. We put special emphasis on smooth decelerated object movements and artifact-free results, to reach a comparable video quality, like the high speed camera approach. In the following sections, we give a short introduction to existing techniques and discuss their visual performance, to determine where and how they can be improved.

Several different concepts exist, to achieve a slow motion effect [25]. The simplest technique is to repeat each recorded frame multiple times. This approach is on the one hand very easy to implement and also fast to compute, but on the other hand it yields very juddery object movements. A smoother result is gained with the *frame blending* approach. Instead of repeating each frame multiple times, novel intermediate images are generated by cross-fading two consecutive frames. This method yields smooth transitions for moderate slowdowns. However, notable artifacts arise at the occurrence of fast movements, because the moving objects are visible twice during blending. The most sophisticated approach is the *motion compensated frame interpolation* [17]. A significant improvement over other methods is gained, by taking the object motion during the computation of novel frames into account. This allows the algorithm to propagate pixel intensities along their corresponding motion vectors, and properly interpolate the intermediate frames. The benefit of this approach are the very smooth decelerated object movements in the resulting video. The required motion information is provided by an optical flow algorithm. Therefore, this method has the highest computational expense and complexity. An illustration of the three presented slow motion approaches is shown in Figure 1.1.



(a) Unknown intermediate frames

(b) Frame repeating

(c) Frame blending

(d) Motion compensated frame interpolation

FIGURE 1.1: The figure shows a comparison on different slow motion concepts. (a) To achieve a video slowdown by the factor of 4, we have to insert 3 intermediate frames between every frame pair (in this example between frame A and B). (b) In the simple approach, the original frames are duplicated multiple times, which leads to a jaggy motion during playback. (c) A smoother result can be achieved by cross-fading frame A and frame B in the intermediate frames. However, for fast moving objects or high slowdowns, this technique still yields very poor results. (d) The smoothest slow motion is gained with the motion compensated frame interpolation approach, where the intermediate frames are generated by propagating pixel intensities along their corresponding motion vectors.

The motion compensated frame interpolation requires a dense motion field, to propagate every single pixel intensity linear along the estimated displacement vector, as shown in Figure 1.2. This is accomplished by applying a variational optical flow approach, like the proposed model of Horn and Schunck [22]. The benefit of their optical flow algorithm is the fill-in effect in weakly textured homogeneous regions, where a local optical flow computation is not feasible. This approach was among others further improved by Zach et al. [43] (Total Variation-L1), to allow sharp discontinuities in the flow field, and by Werlberger et al. [41] (Huber-L1), to gain a smooth solution in weakly textured regions.

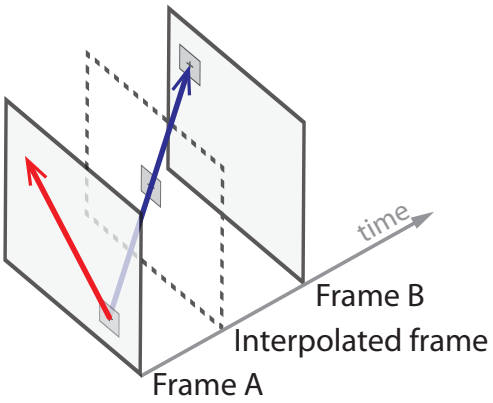
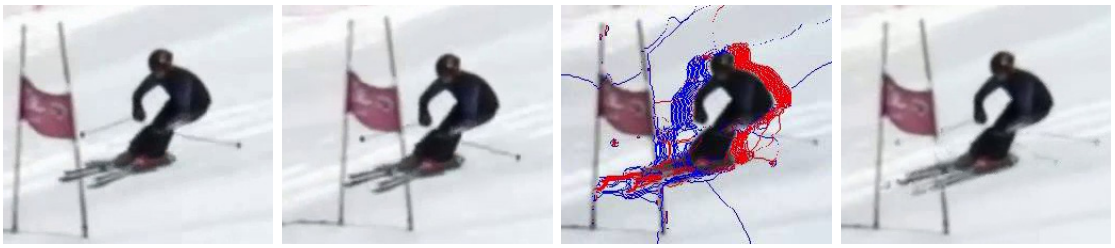


FIGURE 1.2: Linear propagation of a pixel intensity, along the corresponding motion vector. The red vector indicates the displacement of a pixel from frame A to B, visualized on the two-dimensional plane of frame A. The pixel displacement in the three-dimensional space-time volume, spanned by frame A and B, is illustrated by the blue vector. The point, where the blue vector intersects with the plane of the intermediate frame, represents the interpolated position of the moving pixel.

The optical flow is the foundation of the motion compensated frame interpolation. Therefore, the resulting slow motion video strongly depends on the quality and the limitations of the optical flow model. This is clearly notable in the *forward interpolation* approach, where only the forward flow, which is computed from frame A to B, is utilized for the interpolation algorithm. Due to the lack of motion information in disoccluded regions, visible holes emerge in the interpolated frames. This is illustrated in Figure 1.3, where the occlusions and disocclusions are highlighted. A solution to this problem is the *bidirectional interpolation* approach, where beside the forward also the backward flow is taken into account. The holes can be properly filled, because a disocclusion in forward direction is an occlusion in backward direction, and vice versa. This improvement comes with an almost doubled computational expense, due to the additional backward flow estimation.

With the constraint, that the motion estimation algorithm provides a correct optical flow, the bidirectional interpolation approach yields visually pleasant intermediate frames and a smooth slow motion. However, if the optical flow is erroneous, visible artifacts arise in the interpolated frames. To increase the visual quality, we focus on



(a) Frame A (b) Frame B (c) Disocclusion (d) Bidir. interpolation

FIGURE 1.3: Visualization of occlusions (blue) and disocclusions (red), based on the forward flow, estimated from frame A to B. For the red colored pixels, no intensity information is available, due to the lack of motion vectors. In the improved bidirectional interpolation method, these holes are filled with vectors of the opposite flow direction, which leads to the correctly interpolated intermediate frame shown in (d).

the reduction of these occurring errors. In the first place, an error measure needs to be defined, to locate wrongly interpolated pixels. These errors have to be removed and inpainted at the correct location, to preserve the temporal continuity of the slow motion video.

Our proposed solution is aimed at the correction of wrongly estimated displacement vectors, since they cause the occurring artifacts. We detect and mask perceivable interpolation errors, with the warp error measure. Identified erroneous pixels, which are geometrically connected, are treated as individual patches. Based on these patches, we try to enhance the optical flow, with a patch-based motion estimation approach. With the corrected displacement vectors, we are able to determine the proper position of the wrongly interpolated patch, where it needs to be inpainted. The proposed patch-based inpainting algorithm, is able to preserve the correct depth order of the scene, which is necessary for a sophisticated occlusion and disocclusion handling. As the evaluation results show, our proposed slow motion algorithm yields visually attractive results, and is able to significantly reduce the amount of artifacts, in comparison to the bidirectional interpolation approach.

1.2 Related Work

A frame interpolation approach, for the generation of slow motion videos, was proposed by Dudek et al. [17]. They utilize the motion vector field for the image registration, to warp an image into the other. Based on this warping process, they generate novel intermediate frames between every frame pair. The following key aspects are listed as requirements for the motion estimation: support of large displacements, dense motion vector field, fast and robust computation. Therefore, they use the variational approach of Horn and Schunck [22], to estimate the optical flow. To support large displacements, the

optical flow estimation is iteratively performed in an image pyramid, from the coarsest to the finest level. The novel intermediate frames are generated, by propagating the pixel intensities linear along the displacement vectors of the forward flow. The proposed algorithm yields reasonable results, as shown in Figure 1.4. However, since the backward flow is not considered in the frame interpolation, no sophisticated disocclusion handling is achieved.

FIGURE 1.4: Comparison of the simple frame blending approach and the forward frame interpolation approach. The upper row shows the two consecutive input images of the frame interpolation algorithm. In the lower left, the visually poor result of the frame blending approach is visible. The lower right image shows the interpolated result of the frame interpolation algorithm, proposed by Dudek et al. [17]. The images are taken from [17].

Werlberger et al. [42] proposed a framework, which is beside the bidirectional frame interpolation also aimed at video restoration, denoising and inpainting. To gain a robust occlusion and disocclusion handling, the forward and backward flow are taken into account. A slightly modified TV-L1 denoising model is utilized in a spatio-temporal volume, which is spanned by the input images. The temporal derivatives are computed along previously estimated motion vectors, to incorporate the movement of the pixels. For the motion estimation they use the variational optical flow model, presented in [41]. As shown in Figure 1.5, their approach yields properly interpolated intermediate frames, with good results in occluded and disoccluded regions. They state large displacements, fast movements, small-scaled structures and complex occlusions as major difficulties.

An alternative frame interpolation approach, based on a modified TV-L1 optical flow algorithm, was proposed by Rakêt et al. [31]. The data term of the TV-L1 energy functional was re-parameterized, with the assumption of a symmetric optical flow. This requires the unknown intermediate frame, to be located exactly in the middle between the two input images, as shown in Figure 1.2. The solution of the modified energy functional yields an optical flow, which points from the intermediate frame to the input

FIGURE 1.5: The leftmost and rightmost column contain the input images and the center columns the interpolated images. As can be seen in the enlarged frames in the bottom row, the bidirectional frame interpolation algorithm of Werlberger et al. [42] yields visually attractive results. The head movement and the mouth closing are interpolated without visible artifacts. Furthermore, also the occluded region in the background is handled correctly. The shown images are taken from [42].

images, instead of from the first input image to the other. Therefore, the flow warping step, which is necessary for the forward and bidirectional interpolation, becomes obsolete. In their evaluation, the symmetrical approach outperforms the forward and backward interpolation, as shown in Figure 1.6.

FIGURE 1.6: Comparison of different frame interpolation approaches. From left to right column: ground truth images, forward interpolation, backward interpolation, symmetrical approach by Rakêt et al. [31]. It can be observed, that the symmetrical approach yields a visually good result, with less artifacts in comparison to the other two approaches. The images are taken from [31].

1.3 Synopsis

In this section we want to give a brief overview on the structure of the thesis. In chapter 2 we give a detailed introduction to optical flow. We present three different optical flow concepts in section 2.2, and focus on gradient-based models in section 2.3.

After introducing widely used constraints, we discuss local and global approaches. In section 2.3.4 we in detail explain the well performing variational optical flow optimization. Algorithms for motion compensated frame interpolation are presented in chapter 3. After giving an overview on image and video inpainting algorithms in section 3.4, we present our two-step artifact removal strategy in section 3.5. In chapter 4 we explain the evaluation of the implemented algorithm. The utilized optical flow library and its parameterization is presented in section 4.1. The final evaluation results of the Skiline and Ground Truth datasets are presented in section 4.2 and section 4.3. A final conclusion on the presented work and an outlook for further research are summarized in chapter 5.

Chapter 2

Optical Flow

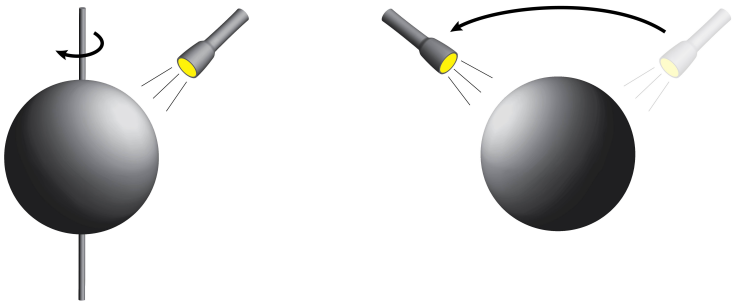
Optical flow is an important topic in the field of computer vision, with applications in video compression, segmentation, 3D reconstruction, object detection and motion compensated frame interpolation. It describes the apparent motion of objects in a sequence of images. Usually two consecutive images of a scene are considered for the computation, where the main goal is to find for each pixel in frame A, a vector pointing to the corresponding pixel in frame B. The resulting displacement vector field is called optical flow.

2.1 Apparent Motion in Image Sequences

In general, motion of a rigid object is understood as geometric displacement of the object itself, the observer, or both. Assuming the observer being a camera, which is capturing the scene, we get an image sequence with certain object displacements. This image-capturing process is a projection of the continuous three-dimensional (3D) space, onto a discrete two-dimensional (2D) image plane, where information loss is inevitable. The remaining information left for motion analysis, are pixel intensities and their changes over time. This is not sufficient to compute the 2D motion field, which is defined as the projection of 3D velocities on the image plane.

Since we have to stick to pixel intensities, we are only able to compute the motion, that can be perceived in the image sequence. This apparent motion is called optical flow and differs from the 2D motion field especially at the occurrence of illumination changes, reflections, untextured objects, shadows, and translucent objects. For example, the 2D motion field of an untextured rotating sphere is non-zero, but the optical flow, because the rotation cannot be perceived in the image, is zero. This example is illustrated in

Figure 2.1. A another problem with moving objects in image sequences are occlusions and dis-occlusions, where the challenge is to find the optical flow for an object of the current frame, which isn't visible or only partly visible in the next frame.



(a) Rotating sphere (b) Static sphere

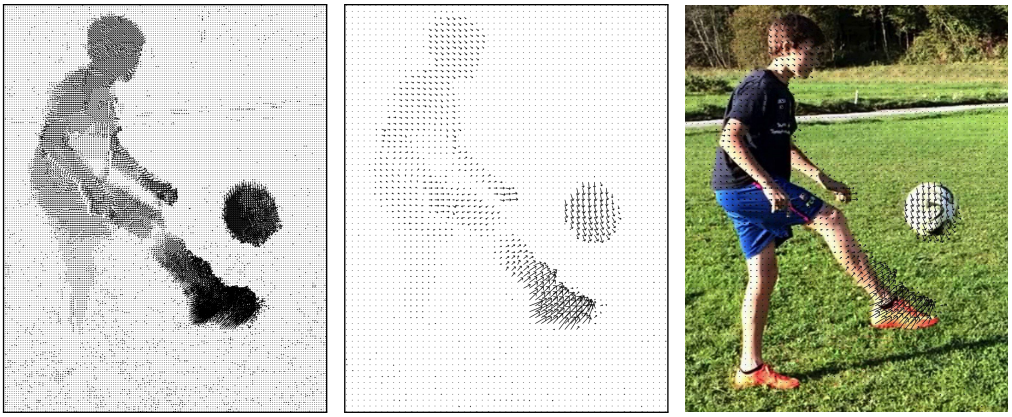
FIGURE 2.1: (a) An untextured sphere is rotating around its own axis and is lit by a static light source. The motion field is non-zero due to the rotation of the sphere, but the optical flow is zero, since the rotation cannot be perceived. (b) A static sphere is lit by a moving light source. The motion field of a static sphere is zero, but the optical flow is non-zero, because of the moving shadow on the sphere.

Despite these difficulties, the aim of optical flow algorithms is to get a result which is as close as possible to the 2D motion field.

2.1.1 Visualization

Optical flow is defined by two components per pixel, therefore its visualization is not a trivial task. The naive approach is to plot the motion vector as an arrow for every pixel in the image. For large images or large motion this method mostly ends up in chaos, since the arrows will overlap with their neighbors. By sub-sampling the vector field, the number of arrows to plot can be significantly reduced. This on one hand improves the visibility, but on the other hand hides some motion information because many vectors are discarded. An alternative approach is, to plot a separate image for each of the two optical flow components, and indicate their magnitude with brightness values. The most common approach for dense optical flow visualization is the color-coded plot, where every pixel gets a color assigned, based on the corresponding flow vector and a two dimensional color map. The direction of the vector is coded by hue, and the length by saturation. Each vector can be roughly estimated, by also plotting the color map. With this approach it is very easy to evaluate the behavior of an algorithm in particular regions, like along edges or in areas with weak texture. A comparison of the arrow-plot and the color-coded plot is given in Figure 2.2. For color-coded plots we use the color-scheme proposed by Baker et al. [2].

(a) Input image (b) Color-coded optical flow (c) Color map



(d) Arrow plot (e) Subsampled arrow plot (f) Mixed

FIGURE 2.2: Several different techniques exist to visualize the optical flow. (a) A cropped input image of the *Football* video. (b) The most common technique is the color-coded plot, where the vector direction is coded by hue and the length by saturation. (c) The color map for the color-coded scheme used in the entire work. A dash on the axes denotes a vector length of one pixel. (d) In the sub-sampled arrow plot every n^{th} pixel gets an arrow assigned, which represents the motion vector. If n is too small, most arrows will overlap, like in this example where $n = 3$. (e) The visibility can be improved, by increasing n to $n = 8$, with the drawback of loosing detail information. (f) To bring the arrow plot in context with objects of the image, it can be overlayed over the input image.

2.2 Optical Flow Concepts

As summarized by Aubert and Kornprobst [1] optical flow algorithms can be divided into three main categories, based on their concept:

1. **Correlation-based**

   Correlation-based methods extract patches from one image and try to find the corresponding patch in the next image, based on a similarity measure like sum of squared differences (SSD) or sum of absolute differences (SAD). If a matching pair of patches is found, the displacement vector is easily computed. The matching

process is based on the intensity representation of a patch, which implies that the intensities of a certain object, should remain constant in both images. Noise, shadows and illumination changes are events, where correlation based methods are very likely to fail, due to the violation of the constant intensities constraint. Furthermore also the optical flow of repetitive structures or homogeneous regions is error prone, due to multiple occurrences of a single patch. An example for a correlation-based algorithm is the method of Barnard and Thompson [3].

2. **Feature-based**
   Correlation-based approaches estimate the correspondence for each pixel in the image, which often leads to false matches in homogeneous regions, where not enough structure is present for reliable patch matching. This can be avoided if only salient points like corners, are considered for the matching process, which is the concept of feature-based methods. These distinct points are located with a feature detector like *Speeded Up Robust Features* (SURF [5]), and are represented by a feature descriptor like *Scale Invariant Feature Transform* (SIFT [26]). Depending on the application and its requirements, the feature detector and descriptor is chosen properly. When using a descriptor, which is robust to illumination changes, then the influence of light and shadows can be minimized. By matching feature descriptors of points from both images, the displacements for those points can be computed. The resulting optical flow is sparse, due to lack of distinct points in the image. Brox et al. [9] presented a sophisticated approach, to convert a sparse to a dense optical flow. By combining the sparse result of a feature-based method and a variational energy minimization, they get a dense optical flow also in homogeneous regions.

3. **Gradient-based**
   Gradient-based (differential) approaches use spatial and temporal derivatives to estimate the optical flow. They are among the most common techniques, and have in comparison to others the best performance (Bruhn et al. [10]). Based on their additional assumption on how to solve the aperture problem (see section 2.3.2), they can be separated in local and global methods. Local methods as presented by Lucas and Kanade [27] assume, that the optical flow is constant in a local neighborhood of a pixel. In contrast to that, global approaches as the method of Horn and Schunck [22], introduce a global smoothness constraint for the optical flow vector field.

Due to the good performance of gradient-based global approaches (Baker et al. [2], Bruhn et al. [10], Weickert et al. [39]) and the ability to compute a dense flow field, we use a global method to compute the optical flow in our project. Therefore we concentrate on

the category of gradient-based optical flow algorithms in section 2.3, where we give an introduction to local approaches and then mainly focus on global methods.

2.3 Gradient-Based Optical Flow

2.3.1 Optical Flow Constraint

A common assumption for the optical flow computation is, that intensities of an object stay constant in the image sequence. Also if the object moves across the image, its gray values stay the same. This assumption is also called *brightness constancy*

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{2.1}$$

where $I(x, y, t)$ is the pixel brightness at position (x, y) and at time t, $(\Delta x, \Delta y)$ is the displacement vector, and Δt refers to the change in time. Considering real world examples, this assumption may be easily violated by for example non-uniform illumination, noise or camera exposure inaccuracy. Nevertheless, approaches based on this assumption in practice have a good overall performance. The reason for that is basically, that most algorithms use two consecutive images with small object displacements or short image capturing intervals, where illumination changes have less impact.

Based on the work of Fleet and Weiss [19], we will at first derive the displacement estimation for the one-dimensional (1D) case. Let $I_1(x)$ and $I_2(x)$ be two consecutive 1D images, where $I_2(x) = I_1(x - d)$, which means that $I_2(x)$ is translated by d from $I_1(x)$, as shown in Figure 2.3. Assuming that the occurring displacement d is small, we can derive an estimator for d by linearising $I_1(x - d)$ at the point x, with the first order Taylor expansion and ignoring higher order terms.

$$I_1(x - d) \approx I_1(x) - dI_1'(x), \tag{2.2}$$

where $I_1'(x) = dI_1(x)/dx$ is the slope of $I_1(x)$. Based on equation (2.2) we get an estimator $\hat{d}$ for the real displacement d

$$\hat{d} = \frac{I_1(x) - I_2(x)}{I_1'(x)}, \tag{2.3}$$

which yields an exact result for linear and an approximation for non-linear signals, as shown in Figure 2.3.

The same approximation with the first order Taylor expansion, valid for small displacements, is applied to 2D images. By expanding the right hand side of the equation (2.1),

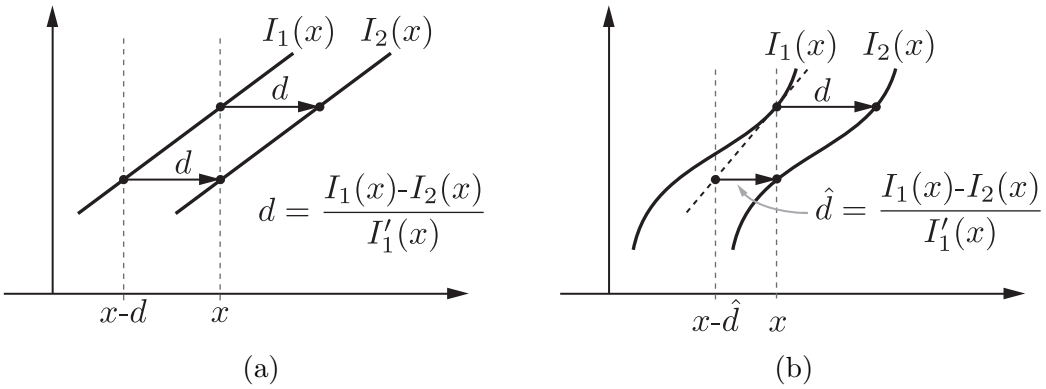


FIGURE 2.3: Approximation of the displacement d between two 1D images $I_1(x)$ and $I_2(x)$, by dividing the temporal difference $I_1(x) - I_2(x)$ with the spatial derivative (slope) $I_1'(x)$. (a) The calculation of d is exact for linear signals. (b) For non-linear signals we get an approximation $\hat{d}$, which is close to d for small displacements . (Fleet and Weiss [19])

we get

$$I(x, y, t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t, \tag{2.4}$$

where $\partial I/\partial x$ and $\partial I/\partial y$ are the spatial derivatives in x and y direction, and $\partial I/\partial t$ is the temporal derivative. When subtracting $I(x, y, t)$ and dividing equation (2.4) by Δt, we according to Horn and Schunck [22] get

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{2.5}$$

By using I_x and I_y for the spatial and I_t for the temporal derivatives, and setting $u = dx/dt$ and $v = dy/dt$ as the displacement components, we get the so-called *optical flow constraint (OFC)* equation

$$I_x u + I_y v + I_t = 0. \tag{2.6}$$

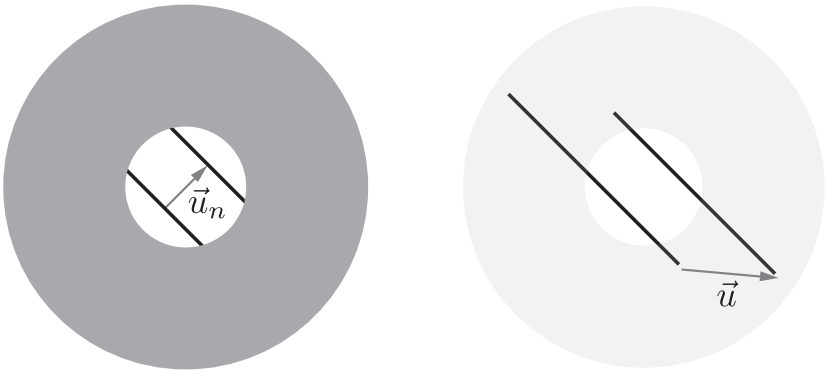
This equation is not sufficient to compute both unknown optical flow components u and v, which is often referred as *aperture problem*.

2.3.2 Aperture Problem

Equation (2.6) represents an under-determined system with two unknowns and just one equation. Without additional constraints this only allows to compute the *normal flow*,

$$\vec{u}_n = \frac{-I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|}, \tag{2.7}$$

which has the same direction as the brightness gradient $\nabla I = (I_x, I_y)^\intercal$ (Beauchemin and Barron [6]). The aperture problem can be visualized by a straight line, which is moving in an arbitrary direction. The movement of the line is observed through a small aperture, which is the reason why only the motion perpendicular to the edge can be perceived. The second component of the motion vector, which is parallel to the moving line, cannot be determined unless the aperture is enlarged, and the edge-corners become visible. This problem is illustrated in Figure 2.4.



(a) (b)

FIGURE 2.4: (a) When observing a moving edge through a small aperture, then only the component of the motion vector, which is perpendicular to the edge, can be perceived. This motion vector $\vec{u}_n$ is also called normal flow. (b) If the edge-corners become visible, also the second component of $\vec{u}$ can be measured. (O'Donovan [28])

Several additional assumptions were introduced, to solve the ill-posed[1] problem in equation (2.6). Depending on whether these assumptions affect the local neighborhood of a pixel or the whole image, the solutions are separated in local (section 2.3.3) and global approaches (section 2.3.4).

2.3.3 Local Approach

Lucas and Kanade [27] introduced an additional constraint, to solve equation (2.6). Their proposed method assumes, that the optical flow is locally smooth. The neighborhood of a pixel $\vec{x}$ is expressed as a patch $\mathcal{N}(\vec{x})$ of the size $n \times n$. For all pixels in $\mathcal{N}(\vec{x})$, the displacement vector is assumed to be constant. This results in an over-constrained system with $n \times n$ equations and two unknown optical flow components $\vec{u} = (u, v)^\intercal$. The system can be written as a weighted least-squares estimator

$$E(\vec{u}) = \sum_{\mathcal{N}(\vec{x})} g(\vec{x})[\vec{u} \cdot \nabla I(\vec{x}, t) + I_t(\vec{x}, t)]^2, \tag{2.8}$$

[1] A ill-posed problem, according to Hadamard [21], is a problem which violates any of the following statements: a solution exists; the solution is unique; the solution depends continuously on the input data.

where $g(\vec{x})$ is a weighting function, which gives close pixels a higher and distant pixels a lower weight, since nearby neighbors are more likely to have the same displacement vector (Fleet and Weiss [19]). By minimizing $E(\vec{u})$ we get the least-squares optical flow estimate $\vec{u}$. Since least-squares problems are convex, the global minimum lies at the point where its slope (derivative of $E(\vec{u})$) is equal to zero:

$$\frac{\partial E(u,v)}{\partial u} = \sum_{\mathcal{N}(\vec{x})} g(\vec{x}) \; [uI_x^2 + vI_xI_y + I_xI_t] = 0 \tag{2.9}$$

$$\frac{\partial E(u,v)}{\partial v} = \sum_{\mathcal{N}(\vec{x})} g(\vec{x}) \; [vI_y^2 + uI_xI_y + I_yI_t] = 0. \tag{2.10}$$

When combining the equations (2.9) and (2.10) and rewriting them in matrix form, we get the system $\mathbf{M}\vec{u} = \vec{b}$, where

$$\mathbf{M} = \begin{bmatrix} \sum gI_x^2 & \sum gI_xI_y \\ \sum gI_xI_y & \sum gI_y^2 \end{bmatrix}, \quad \vec{b} = -\begin{pmatrix} \sum gI_xI_t \\ \sum gI_yI_t \end{pmatrix}. \tag{2.11}$$

Only if $\mathbf{M}$ has rank 2, the displacement estimate $\hat{u}$ can be calculated with $\hat{u} = \mathbf{M}^{-1}\vec{b}$. This depends on the image structure within the local neighborhood $\mathcal{N}(\vec{x})$. Corners and well textured regions yield good results, but uniform areas or a single edge will result in a rank deficient $\mathbf{M}$, which then is singular and thus not invertible. By increasing the size of $\mathcal{N}(\vec{x})$ this problem may vanish, but for too large patches it is more likely that the assumption of constant optical flow will be violated. A common solution is that the size of $\mathcal{N}(\vec{x})$ is kept low, with the drawback of getting a sparse optical flow.

Due to the low computational cost, the local approach is an attractive algorithm for applications where sparse results are sufficient. If a dense optical flow is required, then a method based on the global approach as described in the following section 2.3.4 should be preferred.

2.3.4 Global Approach

With the additional assumption that optical flow vectors change smoothly over the image, global methods introduce a global smoothness constraint. Horn and Schunck [22] first proposed such a variational approach in 1981. Several extensions and further developments of this method were proposed in past years, which are among the best performing algorithms in the optical flow benchmark database of Baker et al. [2]. We will discuss the approach of Horn and Schunck, and then present some improved methods.

2.3.4.1 Horn and Schunk

Instead of solving the optical flow constraint on a local patch, Horn and Schunck [22] presented a method where also the global neighborhood has an impact on the computation of an optical flow vector. Their assumption is, that nearby points have similar velocities and therefore the optical flow field changes smoothly over the image. This constraint is expressed with the smoothness term

$$E_{smooth} = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 = |\nabla \vec{u}(x,y)|^2, \tag{2.12}$$

which measures the change of u and v in both coordinate directions. This term penalizes optical flow discontinuities and supports smooth motion changes. In combination with the optical flow constraint as data term this gives the error to be minimized

$$E_{HS} = \int_\Omega \big(\underbrace{(\vec{u} \cdot \nabla I + I_t)^2}_{\text{data term}} + \underbrace{\lambda |\nabla \vec{u}(x,y)|^2}_{\text{regularization term}} \big) d\vec{x}, \tag{2.13}$$

where $\lambda > 0$ is the regularization parameter, which defines the balance between smoothness (regularization term) and data fidelity. With this equation the error over the whole image Ω regarding each pixels optical flow vector is computed. This error consists of a data term, where the deviation of the flow vector from the spatial and temporal gradient is measured, and a smoothness term, where the amount of smoothness in the pixel-neighborhood is computed. In well-structured regions the data term is dominant, whereas in uniform regions a smooth solution with respect to neighbor vectors is preferred. The propagation of optical flow estimates over such homogeneous regions leads to a dense optical flow field. This automatic *fill-in* effect in poor textured areas is one of the major advantages over local approaches.

To find the optical flow vector $\vec{u}$, we have to derive the energy functional in equation (2.13) and set it equal to zero. By estimating the Laplacian with the difference of u and the weighted average of neighbor-values $\Delta u = u - u_{avg}$, Horn and Schunk developed a system of two equations for each pixel

$$(\lambda + I_x^2 + I_y^2)(u - u_{avg}) = -I_x(I_x u_{avg} + I_y v_{avg} + I_t) \tag{2.14}$$

$$(\lambda + I_x^2 + I_y^2)(v - v_{avg}) = -I_y(I_x u_{avg} + I_y v_{avg} + I_t). \tag{2.15}$$

To solve this large system, Horn and Schunck proposed an iterative scheme, since u_{avg} and v_{avg} depend on their neighbor values which also depend on the current estimate of $\vec{u}$. The optical flow estimate $\vec{u}^{n+1} = (u^{n+1}, v^{n+1})$ at iteration $(n+1)$ can be computed

with

$$u^{n+1} = u^n_{avg} - \frac{I_x(I_x u^n_{avg} + I_y v^n_{avg} + I_t)}{\lambda + I_x^2 + I_y^2} \tag{2.16}$$

$$v^{n+1} = v^n_{avg} - \frac{I_y(I_x u^n_{avg} + I_y v^n_{avg} + I_t)}{\lambda + I_x^2 + I_y^2} \tag{2.17}$$

where (u^n_{avg}, v^n_{avg}) are the weighted averages of neighbor estimates at iteration n (Horn and Schunck [22]). In image regions with brightness gradients close to zero, the estimate will simply be the average of neighbor estimates. If such regions are large, then the *fill-in* is accomplished over multiple iterations. Therefore the number of iterations among others, depends on the size of the biggest homogeneous region to be filled in. If this size is not known in advance, then Horn and Schunck suggest to use the cross-section of the whole image as iteration estimate.

The energy functional in equation (2.13) penalizes deviations of the smoothness constraint in a quadratic way, which results in over-smoothed solutions with no discontinuities in the optical flow vector field. This is a crucial drawback, because discontinuities occur at object boundaries and define the boarders between objects. Also the the data term has a quadratic penalization, which disables to perform a robust outlier handling. These outliers arise if the OFC is violated with illumination changes, shadows, occlusions etc. (Aubert and Kornprobst [1]).

Several different approaches have been proposed to overcome these disadvantages, by modifying or replacing the data and the regularization term. A detailed survey on such global variational approaches with focus on the analysis of data and regularization terms has been done by Weickert et al. [39]. In the following chapters we will focus on two improved methods, which address the major deficiency of the approach by Horn and Schunck [22].

2.3.4.2 TV-L1

The TV-L1 approach assumes constant intensities along motion vectors (brightness constancy) and uses the OFC equation as data term. As regularization term the total variation (TV) norm of the optical flow field is applied

$$TV(\vec{u}) = \int_\Omega |\nabla \vec{u}| d\vec{x} = \int_\Omega \sqrt{u^2 + v^2} d\vec{x}. \tag{2.18}$$

Instead of a quadratic penalization, the L1-norm is applied on both terms, to allow discontinuities in the flow field and perform robust OFC outlier handling. With this

setting we get the TV-L1 energy functional to minimize

$$E_{TV\text{-}L1} = \int_\Omega \Big(|\nabla \vec{u}(x,y)| + \lambda |\vec{u} \cdot \nabla I + I_t| \Big) d\vec{x}. \tag{2.19}$$

Since the data and the regularization term in (2.19) are not continuously differentiable, Chan et al. [14] proposed a numerical optimization with differentiable approximations of both terms. Another approach to efficiently solve the TV-L1 optimization was proposed by Zach et al. [43], and is based on a dual formulation of the TV energy. They introduce a coupling term and split the problem in two sub-problems which are alternately solved. The energy functional with the additional term is

$$E = \int_\Omega \Big(\sum_{d=1}^{2} |\nabla u_d| + \sum_{d=1}^{2} \frac{1}{2\theta} (u_d - v_d)^2 + \lambda |\rho(\vec{v})| \Big) \, d\vec{x}, \tag{2.20}$$

where $\vec{v}$ is an auxiliary variable and $\rho(\vec{v}) = \vec{v} \cdot \nabla I + I_t$ represents the OFC constraint. Zach et al. [43] split the energy functional and perform the optimization in two separate steps:

- 1) For every d and fixed v_d solve

$$\min_{u_d} \int_\Omega \Big(|\nabla u_d| + \frac{1}{2\theta} (u_d - v_d)^2 \Big) \, d\vec{x}. \tag{2.21}$$

- 2) With fixed u, solve

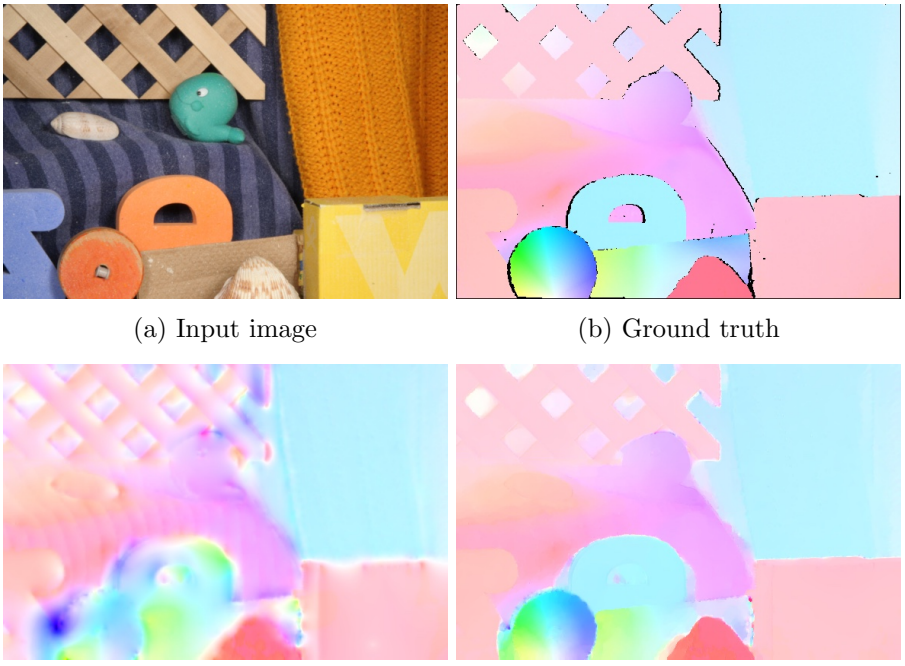
$$\min_{\vec{v}} \sum_{d=1}^{2} \frac{1}{2\theta} (u_d - v_d)^2 + \lambda |\rho(\vec{v})|. \tag{2.22}$$

The minimization of (2.21) is the TV-based de-noising model of Rudin et al. [32], which was efficiently solved by Chambolle [11]. The second step (2.22) boils down to a point-wise thresholding step. For a detailed solution of both problems we refer to the work of Zach et al. [43]. With the support of a parallel graphics processing unit (GPU) implementation, they achieved realtime (30fps) performance on 320×240 pixel images.

To make the improvements of the TV-L1 model over Horn and Schunck visible, we compare both color coded optical flow results, based on the *Rubberwhale* dataset of the Middlebury optical flow benchmark[1] [2]. They provide multiple datasets with image sequences and the corresponding ground truth data. The comparison of both models is shown in Figure 2.5. We can clearly see that the resulting optical flow of the Horn and Schunck approach allows no sharp discontinuities. Edges are strongly over-smoothed,

[1] http://vision.middlebury.edu/flow/

which is caused by the quadratic penalization. In comparison to that, the TV-L1 approach has sharp object boundaries and is very close to the ground truth, which can be explained with the robust outlier handling and L1 penalization.



(a) Input image (b) Ground truth

(c) Horn and Schunck TV-L1

FIGURE 2.5: Comparison of color-coded optical flow results with focus on the behavior along edges. Images taken from Werlberger [40]. (a) One of the two input images of the Middlebury *Rubberwhale* dataset with (b) the corresponding ground truth optical flow. (c) The result of the Horn & Schunck approach is strongly over-smoothed, which leads to blurred edges. (d) We can clearly see that the TV-L1 model yields a more accurate result and allows flow discontinuities.

The TV-L1 approach of Zach et al. [43] was extended by Wedel et al. [38] to reduce the impact of illumination changes, which cause violations of the OFC. In a preprocessing step they perform a structure-texture decomposition on the input images, where the structure represents the main large objects and the texture the smaller scaled details. The structural part is calculated by denoising the gray-value image with the TV based model of Rudin et al. [32] (see equation 2.21). The texture image is computed, by subtracting the structural part and the original image. As their results show, the texture image has significantly less shadows and shading reflections. By using this as input image for the optical flow computation, they can increase the robustness against illumination changes.

2.3.4.3 Anisotropic Huber-L1

TV is a popular choice as regularization term, since it allows sharp discontinuities in the optical flow. Nevertheless it still has some drawbacks in regions with weak texture. As pointed out by Werlberger [40], the fill-in effect of the TV-L1 regularizer yields piecewise constant instead of smooth solutions. This so-called *staircasing* effect can be reduced by using an image-driven (anisotropic) regularization, which uses a quadratic penalization for regions with low gradient magnitudes and linear penalization for high gradient magnitudes. This behavior is achieved with the Huber-norm (Huber [23]) denoted by $|\cdot|_\varepsilon$

$$|s|_\varepsilon = \begin{cases} \frac{|s|^2}{2\varepsilon} & |s| \leq \varepsilon \\ |s| - \frac{\varepsilon}{2} & \text{else} \end{cases}, \tag{2.23}$$

where $\varepsilon > 0$ is the threshold for the decision between the quadratic and the linear penalization. By replacing the L1-norm of the regularization term in (2.19) with the Huber-norm $|\nabla u|_\varepsilon$ we get the Huber-L1 energy functional

$$E_{H\text{-}L1} = \int_\Omega |\nabla u|_\varepsilon d\vec{x} + \lambda \int_\Omega |\rho(\vec{u})| d\vec{x}, \tag{2.24}$$

where $\rho(\vec{u}) = \vec{u} \cdot \nabla I + I_t$ is the OFC constraint. With $||a||_p = \left(\sum_{j=1}^{N} \sum_{i=1}^{M} |a_{i,j}|^p \right)^{1/p}$ and the discrete Huber-norm $||\nabla u||_\varepsilon$ we can rewrite the optimization problem as

$$\min_{u \in Y} ||\nabla u||_\varepsilon + \lambda ||\rho(u)||_1 \quad , \qquad Y := \mathbb{R} \cup \{\infty\}. \tag{2.25}$$

Werlberger [40] applied the primal-dual algorithm ([13] [12]), in order to optimize (2.25). Therefore, the convex conjugate of the Huber function $F(u) = ||\nabla u||_\varepsilon$ must be determined

$$F^*(p) = \sup_{u \in Y} \{\langle \nabla u, p \rangle_Z - ||\nabla u||_\varepsilon\}. \tag{2.26}$$

By considering the two cases of the Huber-norm (2.23) in the computation of the supremum, we according to [40] get

$$F^*(p) = \frac{\varepsilon}{2} ||p||_2^2 + \delta_P(p) \qquad \text{with} \qquad \delta_P(p) = \begin{cases} 0 & \text{if } \; p \in P \\ \infty & \text{else} \end{cases}, \tag{2.27}$$

where $P = \{p \in Z : ||p||_\infty \leq 1\}$. The Huber-L1 energy functional (2.24) can now be formulated as primal-dual saddle-point problem

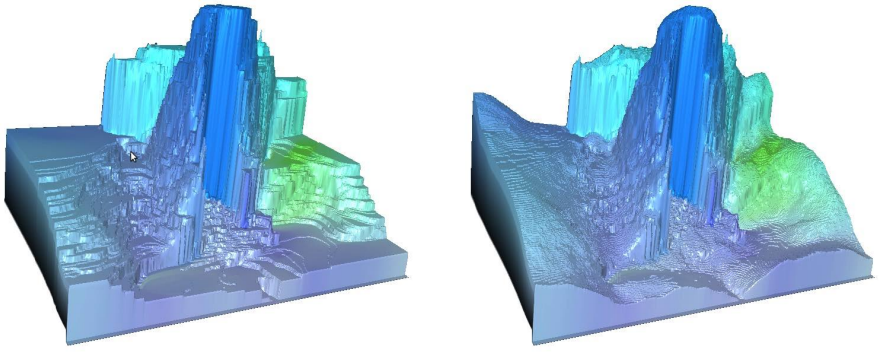
$$\min_{u \in Y} \max_{p \in Z} \langle \nabla u, p \rangle_Z + G(u) - F^*(p) \qquad \text{with} \qquad G(u) = \lambda ||\rho(u)||_1, \tag{2.28}$$

where u denotes the primal variable and p its dual. The solution of minimizing u and maximizing p in (2.28) is given by an iterative algorithm, with the following iteration updates

$$\begin{aligned} p^{n+1} &= \text{prox}_P\left(\frac{p^n + \sigma\nabla\bar{u}^n}{1+\sigma\varepsilon}\right) \\ u^{n+1} &= \text{shrink}\Big(u^n - \tau\text{div}p^{n+1}\Big) \\ \bar{u}^{n+1} &= 2u^{n+1} - u^n, \end{aligned} \tag{2.29}$$

where shrink($\hat{u}$) refers to a soft-thresholding step, regarding the OFC data term. The thesholding step and a detailed solution of this optimization is given in [40].

To compare the the Huber-norm with the TV regularization, Werlberger [40] plotted the u_1 component of the resulting flow as height-field. As can be seen in Figure 2.6 the TV regularization yields piecewise constant levels. Much smoother transitions are achieved with the Huber regularization due to quadratic penalization in regions with weak texture and linear penalization else.



(a) (b)

FIGURE 2.6: By visualizing the height-field of the u_1-component, we can evaluate the presence of the *staircasing* effect (Werlberger [40]). The Middlebury *Dimetrodon* dataset was used as input sequence. (a) With the TV as regularization term, the piecewise constant levels in the height-field are clearly visible. (b) By changing the regularization penalization to the Huber-norm, the result is significantly smoother.

2.3.5 Large Displacements

All in section 2.3 discussed optical flow algorithms are based on the linearized OFC equation (2.6). The linearization limits the methods to only be capable of estimating small displacements (see Figure 2.3). A common extension to enable the estimation of large motion, is the *coarse-to-fine warping* technique. The basic idea is to downsample the input images on a smaller image size, to decrease the motion distance. This is done by

computing an image-pyramid for both input images. The scale-factor for downsampling and the total number of pyramid levels depend on the size of the original image and the largest supported displacement.

As illustrated in Figure 2.7 the first step is to initialize the displacement vectors with $u = 0$ & $v = 0$, and then compute the optical flow with e.g. the anisotropic Huber-L1 algorithm on the coarsest pyramid level. In the next step the moving image is warped towards the fixed image, by propagating the pixel intensities along the estimated motion vectors, to reduce the displacements between the images. Depending on the requirements of the application and computational resources this *solve* - *warp* step may be repeated several times at the same pyramid level. The last step is to prolongate the resulting vectors to the next finer level and use them as initialization. This process is repeated on all pyramid levels, till the original image at the highest level is reached. (Fleet and Weiss [19], Werlberger [40])

Due to consecutive solving on different pyramid levels, this technique estimates the final motion vectors in a step-by-step fashion. A disadvantage of this scheme is, that large displacements of tiny objects cannot be computed. The problem is, that these objects vanish, when downsampling the original image to a coarser level. Therefore their displacement is not computed, since they are not present at coarse levels.

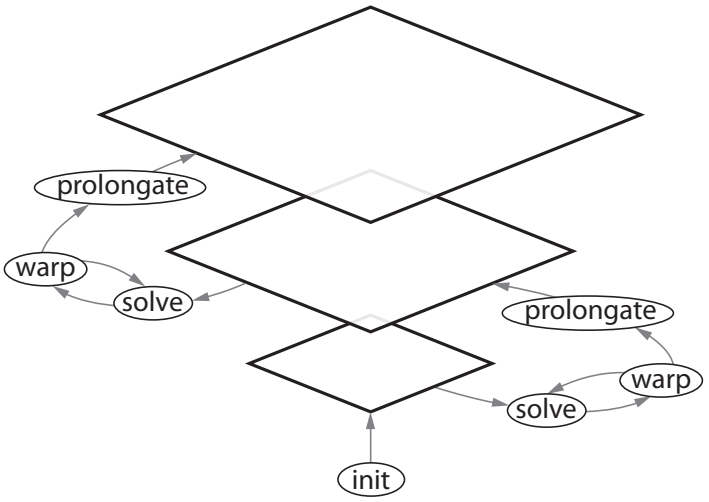


FIGURE 2.7: The coarse-to-fine warping scheme enables optical flow algorithms, which use the linearized OFC, to support large displacements. The algorithm starts to calculate the optical flow at the coarsest level and initializes the next level with the resulting displacement vectors, till it reaches the original image at the highest level.

Chapter 3

Video Frame Interpolation by Motion Compensation

Motion compensated frame interpolation, often just referred as *frame interpolation* [31], is the process of computing one or multiple synthetic intermediate frames, based on a sequence of input images. It is used in the field of video compression [24], video restoration [42], and frame-rate adjustment [15]. The latter has its applications mostly in converting frame rates for different video standards (e.g. PAL: 25 frames-per-second (fps), NTSC: 29.97 fps). By upsampling the frame rate with an frame interpolation method, and playing the video at a lower frame rate, the well known slow motion effect can be achieved.

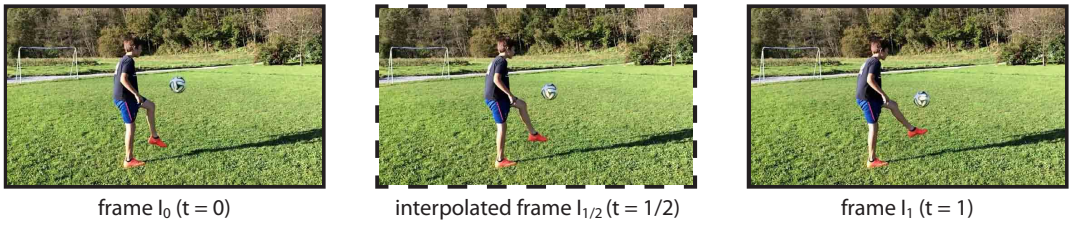


FIGURE 3.1: With the frame interpolation algorithm, novel intermediate frames between the input images I_0 and I_1 are generated. The interpolation is based on object motion information derived from the two known frames.

To interpolate the position of e.g. the ball in the intermediate frame (Figure 3.1), its motion from frame I_0 to frame I_1 must be known. As discussed in section 2.1, the 2D motion field of a captured scene cannot be computed based on image data without additional information. The motion field that can be computed, is the apparent motion called optical flow, which describes the perceivable movement of objects in the image. An essential requirement of the optical flow algorithm is its capability to estimate

a dense displacement field. This is crucial for frame interpolation, since every pixel in the intermediate frame gets interpolated. Therefore, variational global optical flow algorithms are best suited for this task, as pointed out in section 2.3.4.

3.1 Linear Motion

By computing the optical flow field $\mathbf{u}_f$, the warping of a pixel in image I_0 to its position in image I_1 is estimated. The missing information we do not have is how the pixel moved from I_0 to I_1. The object motion between to frames is commonly assumed to be linear. This may be sufficient for straight moving objects. Problems with this assumption arise if an object moves on a nonlinear path, as depicted in Figure 3.2. The motion information gets lost, due to the too low sampling frequency for the fast object motion. The accuracy of the linear approximation increases with a higher recording framerate.

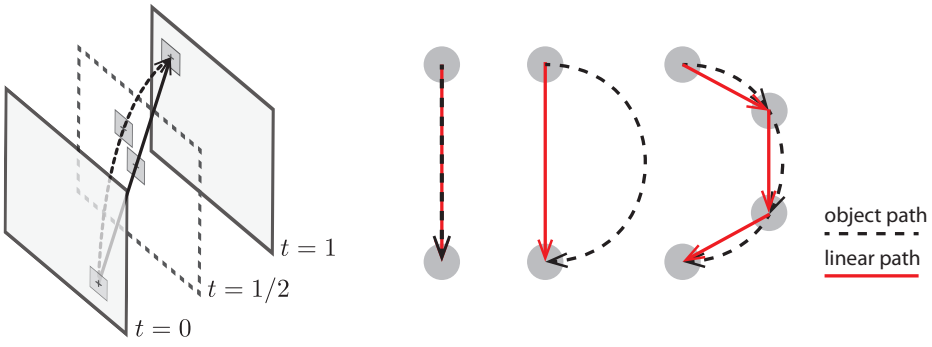


FIGURE 3.2: (a) The assumption of linear object motion is sufficient for straight moving objects. For non-linear movements, this constraint is just an estimate, which causes a misplacement of the pixel in the intermediate frame (image adapted from Werlberger [40]). (b) An increased sampling frequency may minimize this error.

The basic idea of frame interpolation is to propagate the pixel intensities linear along their optical flow vectors. The scale factor $t_i \in (0, 1)$ for the motion vectors depends on the intermediate position of the interpolated frame, on the timeline between frame I_0 and I_1. The relative time factor t_i represents the time between the two known frames, where $t_i = 0$ is the position at frame I_0 and $t_i = 1$ at frame I_1. The intermediate positions t_i are computed, depending on the total number of interpolated frames N, with

$$t_i = \frac{i}{N+1}, \tag{3.1}$$

where i starts at $i = 1$ for the first interpolated frame.

3.2 Forward Interpolation

Since the scaled flow vectors do not point to exact pixel coordinates, most methods perform *flow-warping* with the optical flow field $\mathbf{u}_f$ (Werlberger et al. [42], Baker et al. [2], Rakêt et al. [31]). In the flow warping step $\mathbf{u}_f$ is propagated towards $\mathbf{u}_f^{t_i}$, which represents the warped flow field at the intermediate position t_i

$$\mathbf{u}_f^{t_i}(\lfloor \vec{x} + t_i \mathbf{u}_f(\vec{x}) + 0.5 \rfloor) = \mathbf{u}_f(\vec{x}). \tag{3.2}$$

With this equation every flow vector of $\mathbf{u}_f$ is written on the intermediate position $\text{round}(\vec{x} + t_i \mathbf{u}_f(\vec{x}))$ of $\mathbf{u}_f^{t_i}$. This allows us to compute the interpolated frame i at time t_i with

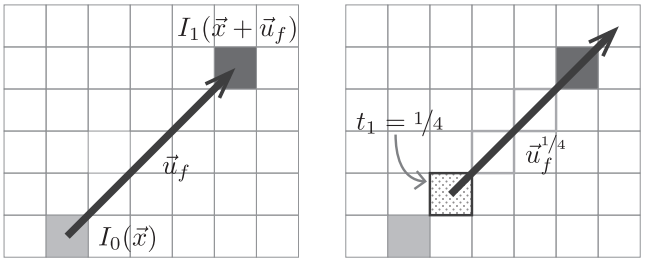
$$I_{t_i}(\vec{x}) = I_0(\vec{x} - t_i \mathbf{u}_f^{t_i}(\vec{x})). \tag{3.3}$$

The interpolation described in equation (3.3) is a simple approach, where pixel intensities from image I_0 are propagated towards the image at t_i, with the scaled motion vectors of the warped flow $\mathbf{u}_f^{t_i}$. As shown in Figure 3.3, the optical flow vector points from the pixel intensity in image I_0 to the intensity in image I_1. Since the pixel information of image I_1 is not taken into account in the simple approach, this may lead to rough transitions in video playback, especially in areas where the optical flow constraint (OFC) is violated. To get a smoother solution, both intensities from image I_0 and I_1 are considered as weighted average, in the *forward interpolation* approach

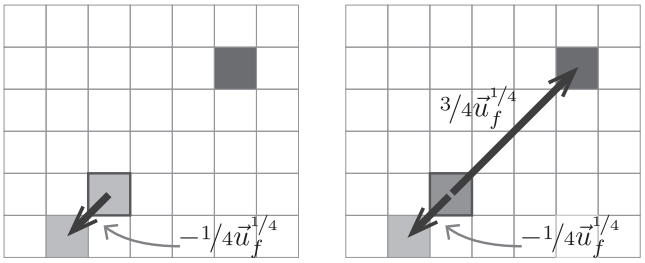
$$I_{t_i}^F(\vec{x}) = (1 - t_i) I_0(\vec{x} - t_i \mathbf{u}_f^{t_i}(\vec{x})) + t_i I_1(\vec{x} + (1 - t_i) \mathbf{u}_f^{t_i}(\vec{x})). \tag{3.4}$$

This approach blends the intensities linearly, with the weights depending on where the intermediate frame is positioned on the timeline between image I_0 and I_1. For $t_i < 1/2$ (closer to I_0), I_0 gets a higher weight, and vice versa. The flow warping step and both interpolation approaches are depicted in Figure 3.3.

Problems with this approach arise from the occurrence of object occlusions and disocclusions, as can be seen in Figure 3.4. Occlusions typically cause the scenario, where multiple optical flow vectors point to the same pixel location in image I_1. We can explain this by analyzing an object occlusion. If an object covers another object, then some pixels in image I_0 lose their corresponding pixels in image I_1, since they get occluded. Therefore they are forced to map to another pixel in image I_1, which obviously causes the error. A common solution to the problem of multiple candidates for a single pixel, is to pick the vector with the best data fidelity. This can be measured with the absolute difference of both pixel intensities $|I_0(\vec{x}) - I_1(\vec{x} + \vec{u}_f)|$.



(a) Flow vector (b) Warped flow



(c) Simple approach (d) Forward interpolation

FIGURE 3.3: Example of the flow warping and frame interpolation process, with $N = 3$ and $i = 1$. All images (I_0,I_1 and $I_{1/4}$) are overlaid. (a) Visualization of the flow vector $\vec{u}_f$, pointing from image $I_0(\vec{x})$ to $I_1(\vec{x}+\vec{u}_f)$. (b) In the flow warping step $\vec{u}_f$ is written on the position of the first intermediate frame t_1 . The other two intermediate positions $t_2 = 1/2$ and $t_3 = 3/4$ are marked with a bold border. (c) Simple frame interpolation approach which propagates the intensity $I_0(\vec{x})$. (d) A smoother transition is gained by blending the intensities of $I_0(\vec{x})$ and $I_1(\vec{x}+\vec{u}_f)$.

Disocclusions cause the inverse problem, which is not as easy to solve. By uncovering an object, novel pixels in image I_1 become visible. They do not have a reference to image I_0, since they initially appear in image I_1. Therefore, no motion vectors point from image I_0 to these disoccluded pixels. This leads to holes in the warped flow $\mathbf{u}_f^{t_i}$, and hence also in the interpolated image. Baker et al. [2] and Rakêt et al. [31] suggest to solve this problem by applying an outside-in strategy on holes in the warped flow field. This is a quite simple approach to fill the holes with neighbor intensities. The impact of occlusions and disocclusions on the flow field is visualized in Figure 3.7.

A resulting intermediate frame of the forward interpolation and the occurring occlusions and disocclusions are shown in Figure 3.5. As can be seen, the interpolation itself and the occlusion handling yield acceptable results, whereas the fill strategy for holes introduces clearly visible artifacts. They are caused by filling the red marked holes, which represent the disoccluded background, with motion vectors of the skier. Therefore the disocclusion handling is still in need for improvement. These deficiencies are addressed with the *bidirectional interpolation* approach.

(a) I_0 (b) $I_{1/3}$ (c) $I_{2/3}$ (d) I_1

FIGURE 3.4: (a,d) The cropped input images of the *Skiline* dataset. (b,c) Forward interpolated frames without occlusion/disocclusion handling are prone to image errors. The black region behind the skier depicts pixels with no intensity information, which originates from the disocclusion. To remove such artifacts, we have to handle these regions by filling in motion vectors.

(a) I_0 (b) I_1

(c) Occlusion/disocclusion (d) $I_{1/2}$: forward (e) $I_{1/2}$: bidirectional

FIGURE 3.5: (a,b) The cropped input images of the *Skiline* dataset. (c) A blue pixel indicates that multiple motion vectors point to it, which is caused by occlusion. Pixels that are not hit by a vector are colored red. These holes are the result of object disocclusions. (d) The forward interpolation scheme yields an intermediate frame with visible artifacts. Multiple candidates for a single pixel are handled by choosing the best fitting vector, which yields a reasonable result. Holes in the flow field are filled with neighbor values. This causes the dark trails, visible behind the skier. (e) A better disocclusion handling is performed in the bidirectional interpolation, where no interpolation artifacts in the disoccluded region are perceivable.

3.3 Bidirectional Interpolation

The main problem of the forward interpolation scheme is the lack of a proper disocclusion handling technique. Disocclusions cause holes in the warped flow field, which means that no information is available on how to color such pixels. A simple approach to fill these holes is to propagate the surrounding neighbor values inside the hole. However, this approach introduces artifacts as depicted in Figure 3.5. A better method to gain vectors for hole-filling, is to additionally compute the optical flow in reverse direction. This is simply done by swapping the input images and estimating the motion field again. The optical flow computed from image I_0 to I_1 is called forward flow, and from I_1 to I_0 backward flow. A color-coded optical flow visualization of the forward and backward flow is shown in Figure 3.6.

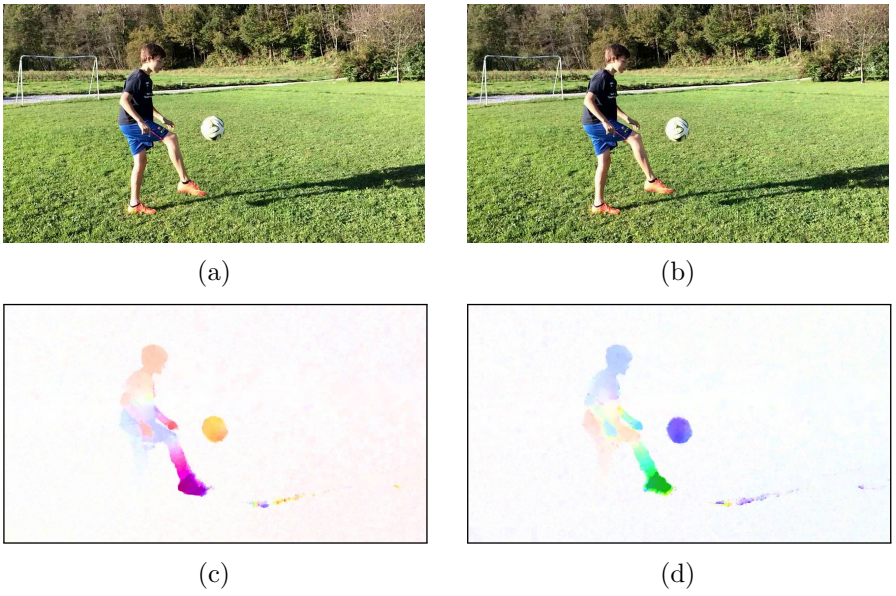
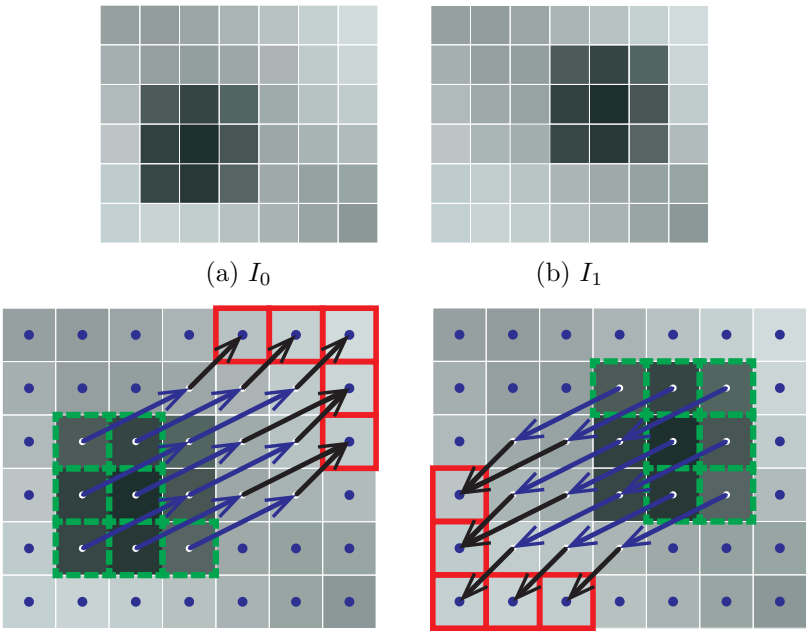


FIGURE 3.6: Color-coded (c) forward and (d) backward flow visualization of the *Football* sequence.

By comparing the resulting displacement field of both directions we can observe that an occlusion in the forward flow is a disocclusion in the backward flow and vice versa. This is also the reason why the optical flow is asymmetric, concerning its computing direction. We can use this as advantage, since the holes of the forward scheme can be filled with the information of the backward interpolation. This forward/backward behavior is visualized and explained with a simple example in Figure 3.7.

Like the forward scheme, also the bidirectional interpolation needs a flow warping step, to be able to generate the intermediate frames. This is performed separately for each of the two flow directions. The warped forward flow is computed as shown in equation



(a) I_0 (b) I_1

(c) Forward flow (d) Backward flow

FIGURE 3.7: A simple example to analyze the behavior of occlusions and disocclusions in forward and backward computed optical flow. (a,b) Two 7×6 pixel input images, with a static background and a dark moving 3×3 box. (c) *Forward flow*: blue vectors indicate the movement of the dark box from the lower left to the upper right corner. Static background pixels with zero motion vectors are drawn as blue dots. These vectors point on the same position where they originate from. More interesting is the movement of pixels, with black motion vectors. Due to the fact that their corresponding pixels in image I_1 are occluded, they are forced to point to wrong locations. This causes the situation, where multiple vectors point to a single pixel, which is indicated by a red bold frame. Disoccluded pixels are marked with a green dashed border. No flow vector points to such pixels, since the initially appear in image I_1. (d) *Backward flow*: in general we can observe the same behavior as seen in forward flow, but since the direction is reversed, also the occlusions and disocclusions are reversed. This allows us to fill the holes of the backward flow with the forward flow and vice versa.

(3.2). Analogous to this, the backward flow $\mathbf{u}_b$ is warped with

$$\mathbf{u}_b^{t_i}(\lfloor \vec{x} + t_i \mathbf{u}_b(\vec{x}) + 0.5 \rfloor) = \mathbf{u}_b(\vec{x}). \tag{3.5}$$

Both, the forward and the backward interpolation use the same relative time factor t_i, which indicates the intermediate frame position between I_0 and I_1. Therefore the convention that $t_i = 0$ is at image I_0 and $t_i = 1$ at I_1 is valid for both. With the warped backward flow $\mathbf{u}_b^{t_i}$, we are able to compute the backward interpolated frame as a weighted average of both pixel intensities

$$I_{t_i}^B(\vec{x}) = (1 - t_i) I_0(\vec{x} - t_i \mathbf{u}_b^{t_i}(\vec{x})) + t_i I_1(\vec{x} + (1 - t_i)\mathbf{u}_b^{t_i}(\vec{x})). \tag{3.6}$$

The final bidirectional interpolated frame is the average of the forward and backward computed frames $I_{t_i} = 1/2(I_{t_i}^F + I_{t_i}^B)$, according to Rakêt et al. [31]. If holes occur in

one of the two images, then only the color information of the filled image is taken into account. In case that a hole of image I_0 overlaps with a hole in image I_1, then no color information for the interpolated image is available. We handle this by applying the already mentioned outside-in strategy on the overlapping region in both images. In practice this scenario appears only over a very small range of pixels, and therefore has not much impact on the final result.

As compared in Figure 3.5, the bidirectional interpolation yields more accurate results than the forward scheme, since a more sophisticated occlusion and disocclusion handling is performed. The drawback of this technique is that it is computationally more expensive. The estimation of the optical flow, which already is the most demanding process, has to be executed two times per image pair, to get the forward and backward flow.

Results of the bidirectional interpolation method are shown in Figure 3.8. We can observe, that the overall quality of the intermediate frames is very satisfactory, however some artifacts are still visible. They are not caused by the interpolation method, but have its origin in the optical flow estimation. The quality of the interpolated results strongly depends on the accuracy of the optical flow. Therefore errors in the displacement field are in general also perceivable in the intermediate frame. Small scaled objects, strong shadows, illumination changes, etc. are events which cause problems for the flow estimation and thus also for the frame interpolation. We are able to identify some of these errors, by analyzing the bidirectional optical flow and the corresponding pixel intensities.

3.3.1 Optical Flow Consistency

A common error measure for the bidirectional optical flow is the vector consistency. It measures the similarity of the forward and the backward flow, by simply calculating the euclidean distance between the corresponding vectors

$$e_{dist} = \sqrt{(x_{f_x} - x_{bx})^2 + (x_{f_y} - x_{by})^2}, \tag{3.7}$$

where $\vec{x}_f = \vec{x} + \mathbf{u}_f(\vec{x})$ and $\vec{x}_b = \vec{x} - \mathbf{u}_b(\vec{x} + \mathbf{u}_f(\vec{x}))$, as shown in Figure 3.9. Large error distances indicate, that the computation of the forward flow yields a different result than the backward flow. In the best case the corresponding backward vector points on the origin position of the forward vector, with the distance between the vectors being close to zero.

This error measure denotes, for which regions in the image the flow estimation is dependent on the order of the input images. As can be observed in Figure 3.7, such regions are

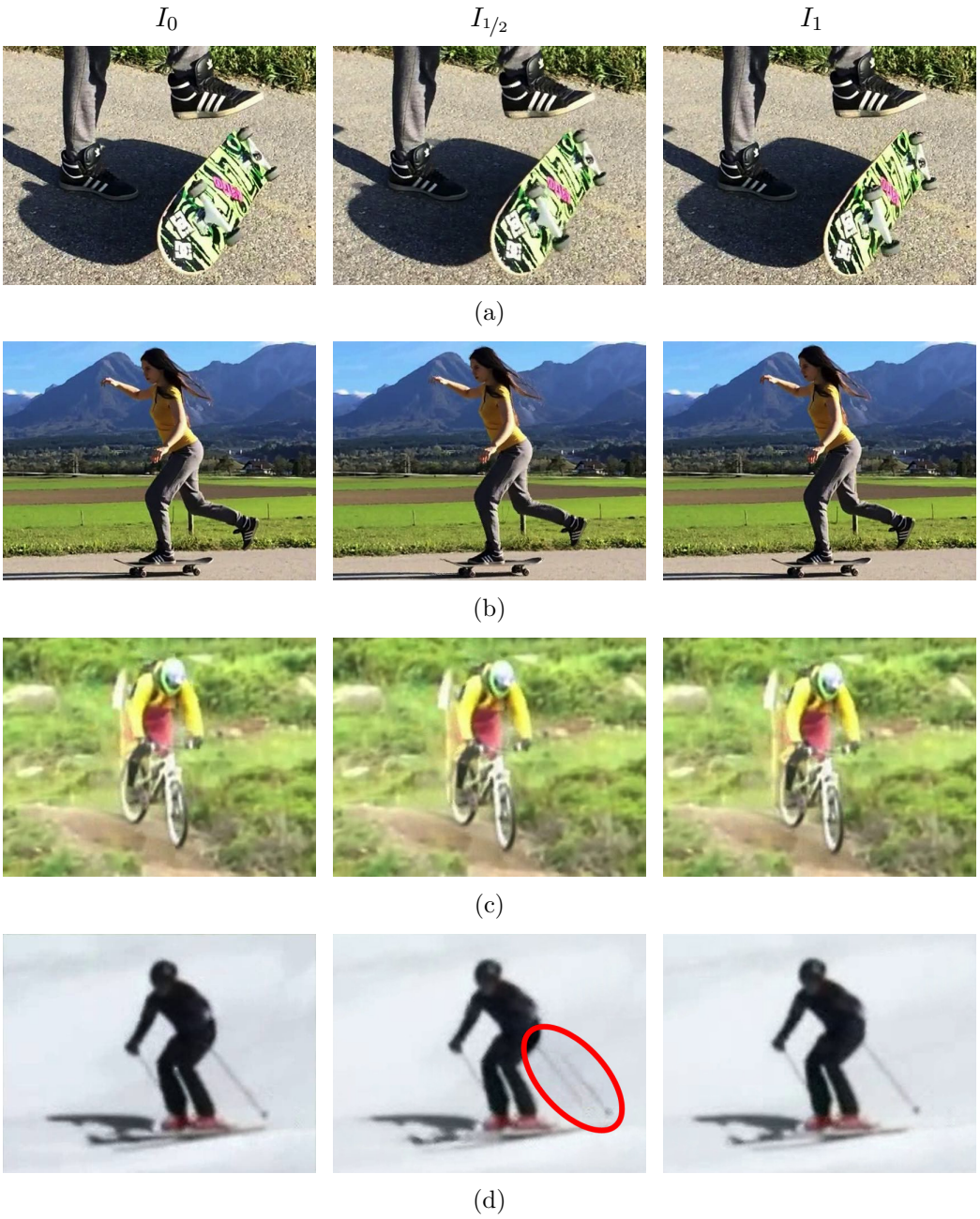


FIGURE 3.8: The left and right column contain the cropped input images and the center column the bidirectional interpolated frame. The object displacements are better visible if the image sequence is played back as video, but in a printed work we have to align them side by side. Different kinds of environments and movements are shown in these four scenes. The bidirectional interpolation yields a satisfying performance in all of them, since no noticeable interpolation artifacts are visible. Errors which originate from the optical flow estimation, are also propagated to the interpolation process. An example for this is given in (d), where a skiing-stick is visible twice. Such small objects are prone to errors, since they are not visible in the coarser levels of the coarse-to-fine warping image pyramid, as explained in section 2.3.5. A similar error is visible in (b), where a pole of the fence (visible between the legs) is also drawn two times.

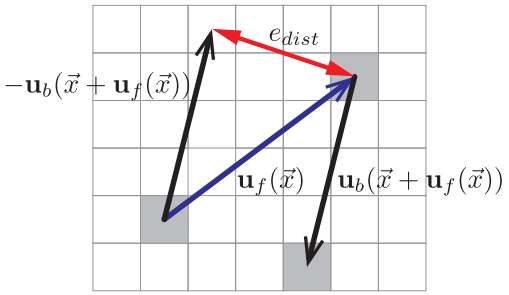
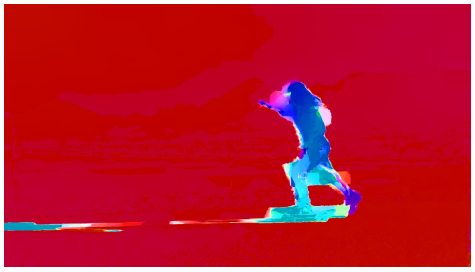
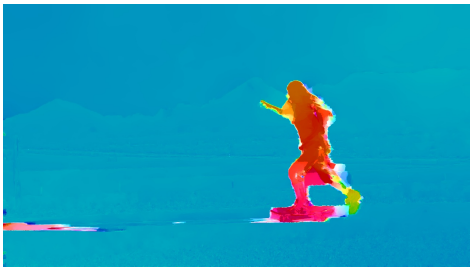


FIGURE 3.9: The flow consistency is measured between the forward vector $\mathbf{u}_f(\vec{x})$ and the backward vector $\mathbf{u}_b(\vec{x}+\mathbf{u}_f(\vec{x}))$. Since the backward flow is computed in reverse direction, the vector $\mathbf{u}_b(\vec{x}+\mathbf{u}_f(\vec{x}))$ is inverted and shifted onto the origin of the forward vector, to compute the euclidean distance e_{dist}.

very likely to be occlusions or disocclusions. Other image parts, especially well textured regions, typically yield low consistency error values. This is also verified in Figure 3.10, where the consistency of the *Skateboard* dataset is visualized.



(a) Forward flow



(b) Backward flow

$1.5 \leq e_{dist} < 5$	gray
$5 \leq e_{dist} < 10$	white
$10 \leq e_{dist} < 20$	green
$20 \leq e_{dist} < 30$	blue
$e_{dist} \geq 30$	red

(c) Color-coded flow consistency error

FIGURE 3.10: (a,b) Color-coded representation of the forward and backward flow, calculated on the *Skateboard* dataset. In these visualizations some rough differences in the flow fields are visible, for example at the knee of the skateboarder. (c) By calculating the euclidean distance e_{dist} between the corresponding forward and backward flow vectors, the error is visualized according to the color map. It is clearly visible that the consistency error is large at the outer boundary of the moving object, where the occlusions and disocclusions occur.

The optical flow consistency error is well suited to analyze the result of the bidirectional flow estimation, and to determine the inconsistent regions in the image. However, it is not able to identify the perceivable errors in the resulting interpolated image, since it

takes only the flow vectors into account and not the pixel intensities. If for example the displacement of a small object is wrongly estimated in the forward and backward direction, then this error will not be indicated by the consistency error. Also artifacts, caused by too large object displacements, are not detected with this error measure. These errors occur in forward and backward direction, and thus cause no flow inconsistency. Therefore, an error measure has to be introduced, which also takes the pixel intensities into account, to detect the perceivable artifacts.

3.3.2 Warp Error

In general the variational optical flow algorithm, yields a good overall performance with reliable displacement vectors. However, due to the coarse-to-fine warping technique, small objects and large displacements often pose a problem. Due to the smoothness term of the variational optical flow model, the displacement of such objects is obtained from their neighboring vectors. This causes disruptive artifacts in the interpolated image, as already observed in Figure 3.8 (d). To minimize such image errors, we have to locate them and then try to restore the correct content. A salient property of erroneous vectors is their poor data fidelity. This can be used as an error measure, which is based on flow vectors and pixel values. We define the warp error, as the difference of the two corresponding pixel intensities of a displacement vector $\mathbf{u}(\vec{x})$

$$e^f_{warp} = I_0(\vec{x}) - I_1(\vec{x} + \mathbf{u}_f(\vec{x})) \quad \text{and} \tag{3.8}$$

$$e^b_{warp} = I_1(\vec{x}) - I_0(\vec{x} + \mathbf{u}_b(\vec{x})), \tag{3.9}$$

where e^f_{warp} is for the forward and e^b_{warp} for the backward flow, as depicted in Figure 3.11. This error measure indicates vectors which establish a correspondence between two pixels

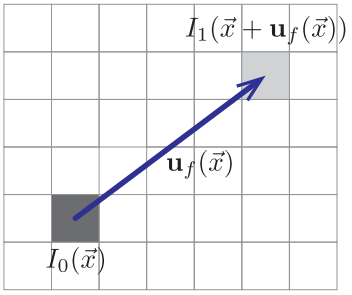


FIGURE 3.11: The warp error e^f_{warp} is calculated by subtracting the intensity $I_0(\vec{x})$ and its corresponding pixel $I_1(\vec{x} + \mathbf{u}_f(\vec{x}))$. It measures how well the two intensities of the displacement vector $\mathbf{u}_f(\vec{x})$ match, which is required for a reliable flow vector. The backward warp error e^b_{warp} is computed analogous to this, with the vectors of the backward flow.

with unequal intensities. Such vectors obviously point to wrong locations, since the

brightness constancy assumption of the optical flow estimation is violated. However, this assumption is rarely fulfilled, especially in real world scenarios where sensor noise of the camera is existent and illumination changes in the scene may occur. To be robust to such minor allowed variations, we perform a thresholding step after calculating the warp error. This yields a binary-mask as result, which denotes vectors with not matching pixel intensities. The error mask is calculated separately for vectors of the forward and backward flow. As shown in Figure 3.12, we use this error measure to identify image errors, which originate from the optical flow estimation process. It is also important to mention, that only those errors are recognized by this error measure, which have a large color deviation. This means that a wrong displacement vector will not be marked as erroneous, if the pixel it points to has nearly the same intensity as the pixel at the correct position. In most cases such errors will not be perceived in the interpolated image, since the color-difference is minimal. This can be also observed in Figure 3.12, where the snow in the background has incorrect motion vectors.

3.4 Inpainting

The previously explained bidirectional interpolation approach generates intermediate frames, based on a pair of input images. It propagates image intensities along scaled motion trajectories, estimated with an optical flow algorithm. Occlusions and disocclusions are handled robustly by filling holes with information of the reverse flow respectively. As can be seen in Figure 3.8, the overall performance of this interpolation approach is mostly sufficient. Nevertheless, there are still some visible artifacts, that occur in certain situations, like shown in Figure 3.12. The primal reason for such image errors are incorrect motion vectors, which are mainly caused by too large displacements or too small objects. As explained in section 3.3.2, we are able to measure these in the intermediate frame perceivable artifacts with the warp error. In this section, we will mainly focus on how to handle such image errors and how to reconstruct the proper content.

3.4.1 Image Inpainting

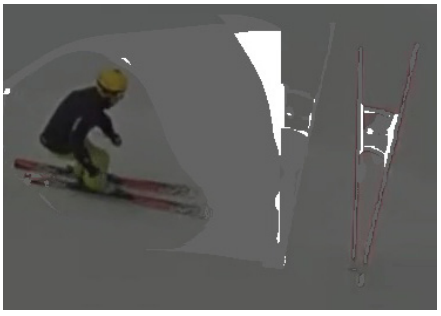
Image inpainting is a widely used technique to perform object removal or restoration of damaged images. The basic idea is to fill a predefined region (target region) with content derived from the local pixel neighborhood or the whole image, with the main goal of achieving a visually pleasant result. Several different approaches were proposed in past years, which can be divided into three main categories:

(a) I_0 & I_1

(b) $I_{1/4}$

(c) $I_{1/2}$

(d) $I_{3/4}$



(e) Consistency error

(f) Warp error

FIGURE 3.12: (a) The cropped input image pair of the *Skiline* dataset, with a large movement of the skier and the camera in the same direction. Therefore the skier is nearly static, whereas the ski-gate has a large displacement on the image plane. The ski-gate pixels have an estimated optical flow of nearly zero, wherefore it is drawn twice. Once at the position where it appears in I_0 and once where it appears in I_1. We can explain the wrong estimation of the ski-gate optical flow, by its type of loose construction and the large displacement of over 70 pixels. The correct interpolated position of the ski-gate however, should be between the two drawn gates. (e) Flow inconsistency is measured between the skier an the ski-gate. (f) All pixels of both gates are accurately recognized as warp error and marked with red color, since their motion vectors are clearly incorrect. The displacement of the snow in the background is also estimated wrongly. This region is not identified as error, because the color deviation of the wrong vectors is too small. Due to the blending effect of the bidirectional interpolation, this misbehavior is not perceived as disturbing artifact, when played back as video.

- **Structural inpainting**

  Structural inpainting methods try to preserve the geometric structure of the image, in the target region. Bertalmio et al. [7] for example proposed a method, which linearly propagates isophote lines through the target region. As shown in Figure 3.13, the algorithm is able to successfully inpaint the superimposed text, to restore the whole image.

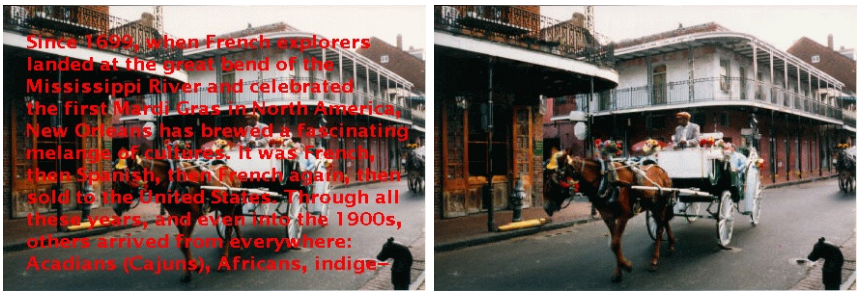


FIGURE 3.13: Result of the structural inpainting algorithm. The images are taken from Bertalmio et al. [7].

Inpainting methods based on structure propagation introduce perceptible blur when applied on too broad regions. The reason for this is that no textural information is considered in the inpainting process. Therefore, such algorithms are mainly used to inpaint thin regions like dust particles, scratches or text.

- **Textural inpainting**

  Texture synthesis based inpainting methods perform a texture analysis in the image, and try to replicate it in the target region. Due to these repetitive texture patterns, larger areas in comparison to the structural approach can be inpainted, without significant loss of details. A representative method for pure textural inpainting is the approach of Efros and Leung [18], shown in Figure 3.14. Textural inpainting has its advantages in being able to inpaint large regions, however it is not suitable to propagate structural elements like edges.

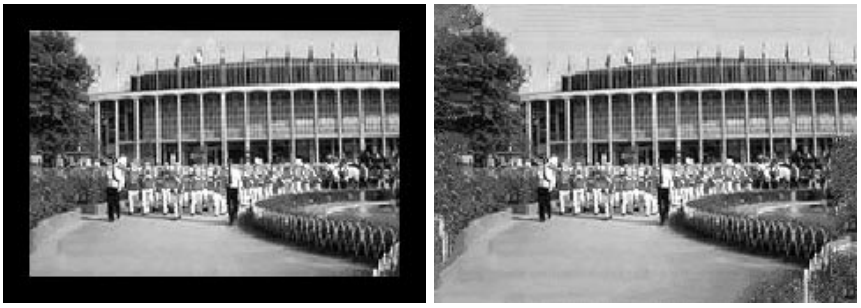
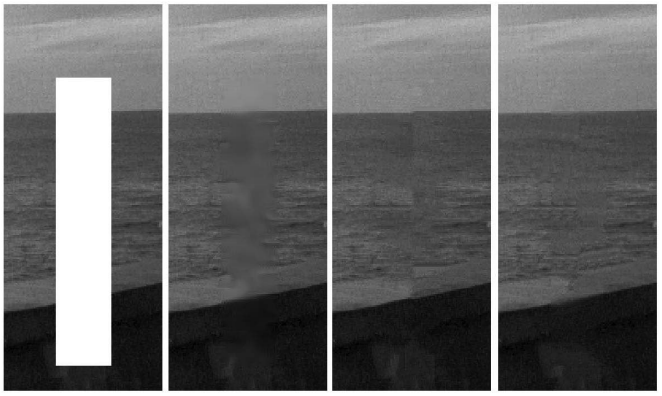


FIGURE 3.14: Textural inpainting performed in the black region of the left input image. Whereas some texture synthesis based algorithms require user-interaction to specify which texture has to be replicated in the target region, this approach of Efros and Leung [18] does this automatically. These images are taken from. [18].

- **Combined structural and textural inpainting**

  To overcome the disadvantages of textural and structural inpainting, Bertalmio et al. [8] proposed a method which combines both approaches. They decompose the image in a structure and texture part, and inpaint the target region in both parts separately. To get the final image, as last step both sub-images are aggregated. The comparison between a pure structural, a pure textural and the combined approach is shown in Figure 3.15, where the advantages of the latter are clearly visible.



(a) (b) (c) (d)

FIGURE 3.15: Comparison of inpainting approaches based on the white target region in input image (a). The images are taken from [8]. (b) The structural approach yields a blurred solution with correct edge propagation. (c) A more detailed solution is gained by applying the textural approach with the drawback of inaccurate edge reconstruction. (d) Bertalmio et al. [8] combine both methods to get a detailed textured solution and proper edge paths. Due to replicating the texture from the source into the target region, some repetitive patterns are noticeable in the region of actually random waves.

  A different approach to combine the advantages of structural and textural inpainting was proposed by Criminisi et al. [16]. They perform large object removal with an exemplar-based inpainting algorithm. By replicating similar patches from the source image into the target region, their method is capable to preserve structural and textural information, as can be seen in Figure 3.16. Furthermore, they also discuss the importance of the fill order and present an edge-driven ordering as alternative to the commonly used *onion peel* strategy.

The above mentioned inpainting algorithms are designed to be applied on still images. In most cases user-interaction is required to mark the desired inpaint region in the image. To inpaint this region, the algorithms take information of the spatial pixel neighborhood into account. In the field of video processing where a sequence of frames is available, this is not sufficient to get a pleasing result. Therefore, additional constraints like temporal consistency have to be considered for the inpainting process.

FIGURE 3.16: Object removal performed by the exemplar-based inpainting algorithm of Criminisi et al. [16]. Both, structure and texture are preserved in the large target region. The images are taken from [16].

3.4.2 Video Inpainting

Inpainting is often applied on image sequences to perform restoration of historic video material or object removal. A naive approach is to utilize the same algorithms of image inpainting for every single frame of the video. Since the pixel neighborhood of the target region will change from frame to frame, the inpainted results will vary over time. This causes perceivable artifacts (ghost shadows) if the image sequence is played back. To minimize such image errors, pure spatial inpainting algorithms have to be extended to also take temporal continuity into account.

Patwardhan et al. [29] proposed a basic video inpainting approach for image sequences recorded with a static camera. Their algorithm is capable to remove moving objects which occlude stationary background. This turns out to be a simple task for videos with a stationary camera. Furthermore, their algorithm also supports inpainting of occluded moving objects, which is a more demanding task. For both scenarios spatio-temporal information is considered to maintain a time consistent solution. The proposed method divides the image in a moving foreground and stationary background part, based on motion information of the optical flow. To inpaint occluded moving objects, patches from undamaged frames are copied in the desired region of the current frame. This is a similar strategy as the example-based inpainting for images, with the enhancement that the search space is extended to a volume of multiple images. As shown in Figure 3.17, the algorithm yields visually pleasant results, with the limitation of only being applicable on videos with a static camera. To compensate this disadvantage Patwardhan et al. [30] further developed their algorithm to also support image sequences with a moving camera. However, they constrain the camera motion to be approximately parallel to the plane of image projection. Furthermore, they also assume to have a stationary background scene and moving foreground object.

FIGURE 3.17: The video inpainting algorithm of Patwardhan et al. [29] takes spatial and temporal information into account to fill the target region. The method is limited to image sequences with a stationary camera. These images are taken from [29].

The exemplar-based video inpainting approach proposed by Shih et al. [33] has no restrictions regarding the camera motion. Their method propagates an user-selected target mask along optical flow vectors trough the image sequence. Therefore the selection of the desired inpaint region is just required for the first frame. With an improved patch matching strategy based on image gradients they inpaint this region with a temporal continuity constraint. This ensures that nearly no artifacts like ghost shadows are perceivable in the resulting video sequence. As their evaluations show, camera movements like tilting and scenes with a perspective view (non-planar targets) still cause noticeable artifacts.

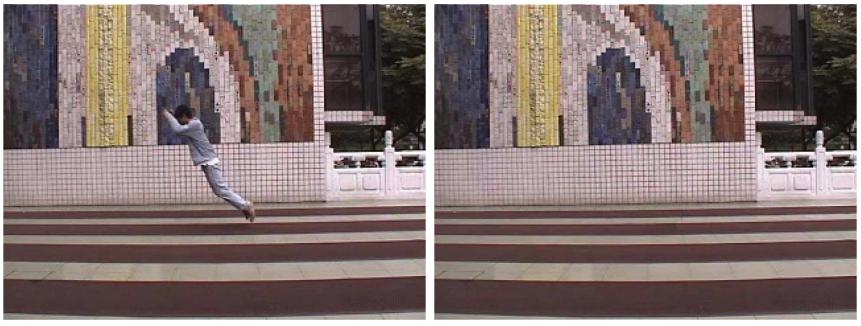


FIGURE 3.18: Successful object removal with the temporal consistent video inpainting algorithm of Shih et al. [33]. The images are taken from [33].

Granados et al. [20] proposed an algorithm, which is designed to handle dynamic objects and a free-moving camera. The algorithm assumes a static background and additionally requires a mask for the object to be removed and a mask for the dynamic object that remains in the scene. Both masks have to be user-defined for every single frame of the video. To inpaint the target region the algorithm aligns other frames in which the occluded region is visible. This process is based on the assumption that the scene is approximated with a piece-wise planar geometry. To align a frame-pair, the homography for each such planar region is estimated. This enables to copy disoccluded pixels from other frames in the target region to be filled. The proposed algorithm yields sophisticated results with no noticeable ghost shadows, as shown in Figure 3.19.

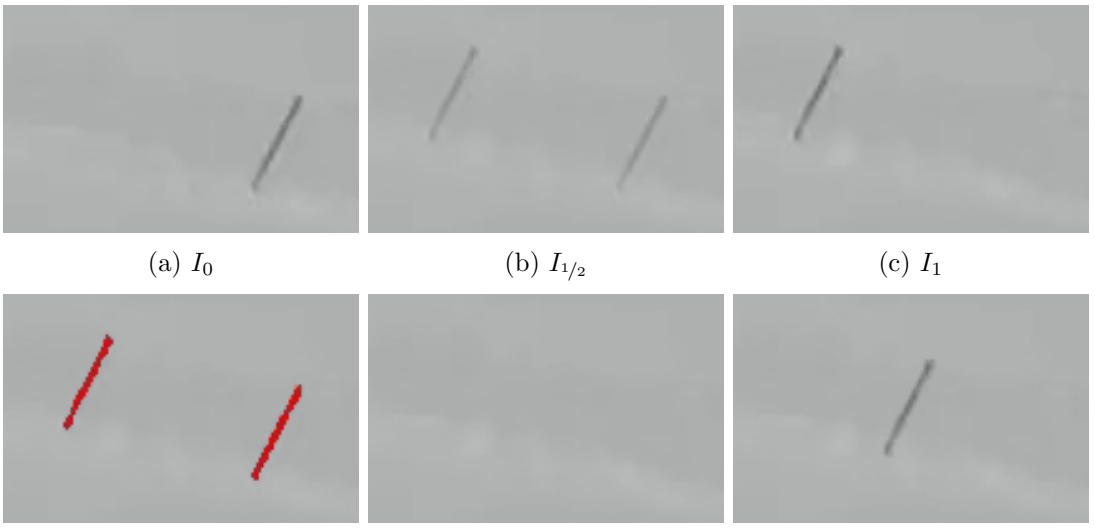
FIGURE 3.19: The video inpainting algorithm of Granados et al. [20] yields visually pleasing results, even for complex background scenes. A mask for objects to be removed and a mask for dynamic objects which remain in the image is required. The images are taken from [20].

The presented video inpainting algorithms in fact yield sufficiently good results, but require a certain scene, restricted camera motion or manual labeling of the target region. We can conclude that visually attractive video inpainting is possible in a constrained setup, but no general solution exists which covers all scenarios. The algorithms are basically designed to fulfill the requirements of a specific application environment. Therefore, we do not use an existing implementation, but develop an algorithm that perfectly fits our needs.

3.5 Artifact Removal Strategy

To develop a strategy on how to handle the remaining interpolation artifacts, we analyze the occurring errors and specify our application specific constraints. As discussed in section 3.3.2, small objects and too wide displacements of larger objects often pose a problem for the optical flow algorithm. They are not supported by the coarse-to-fine warping scheme which enables the estimation of large displacements (see section 2.3.5). Since the optical flow vectors are a fundamental part of the motion compensated frame interpolation, these mentioned objects are not interpolated properly. This leads to artifacts in the intermediate frame, which become clearly noticeable during video playback. A straight forward approach to get rid of these introduced artifacts is to apply an inpainting algorithm. The visible image errors are measurable with the warp error as shown in Figure 3.12. We use this warp error mask to define the target region for inpainting, instead of requiring additional user-interaction. However, this does not solve the problem of getting the correct intermediate frame, as explained in Figure 3.20.

To get a visually pleasing and temporal consistent interpolation result, we need a solution which handles the detected image errors properly, and additionally recovers wrong positioned objects at the correct intermediate position. We present a method which solves this problem in two separate steps:



(a) I_0 (b) $I_{1/2}$ (c) I_1

(d) Warp error (e) Inpainted frame (f) Correct $I_{1/2}$

FIGURE 3.20: (a,c) The cropped input images showing a fence pole recorded with a moving camera. (b) Due to the large displacement of about 80 pixels the optical flow algorithm is not able to estimate the correct displacement. Therefore, the intermediate frame is interpolated with the pole at the wrong position. As can be seen in this example, the object is drawn two times, since the forward and backward motion vectors do not match. (d) With the warp error mask the wrong positioned poles in $I_{1/2}$ are correctly identified. (e) These artifacts can be removed by inpainting the warp error mask and retrieving the background. This on the one hand yields an artifact free image, but on the other is not enough to obtain a visually pleasant intermediate frame, as shown in (f). For a temporal consistent interpolation result a further step is needed, where the pole is inpainted at the correct intermediate position.

1. **Calculate object displacement**

   Determine the correct intermediate position of a wrongly interpolated object, by estimating its displacement from image I_0 to I_1.

2. **Inpaint object**

   Inpaint the object at the in step 1 determined intermediate position, with a patch based algorithm and depth ordering.

During the explanation of the proposed approach within the next chapters, we always refer to the forward flow, even though the method is analogously also applied on the backward flow. This means that we assume the images I_0 and I_1 and the optical flow computed from I_0 to I_1 as input for the algorithm. To apply the artifact removal approach on the backward flow, the image order is simply reversed.

3.5.1 Calculate Object Displacement

The warp error mask denotes the image error, which has to be removed by retrieving the background, but it does not indicate the intermediate position where the wrongly

interpolated object needs to be restored, as shown in Figure 3.20. By calculating the displacement from image I_0 to I_1 of such error prone objects, the correct intermediate position can be determined. Since optical flow algorithms, which are based on the coarse-to-fine warping scheme, fail to estimate the motion, we have to stick to another approach.

The objects which cause interpolation errors are easily extracted as patches, by filtering the input image with the warp error mask. Therefore, the correlation-based optical flow approach (see section 2.2) is very well suited to estimate the displacement of the patches. It matches an object patch of image I_0 with its corresponding representation in image I_1. The matching process is implemented with a sliding window approach, where the extracted patch is compared to all possible candidates in image I_1. As similarity measure we use the *mean absolute error* (MAE)

$$e_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^{n} |I_0(i) - I_1(i)|, \tag{3.10}$$

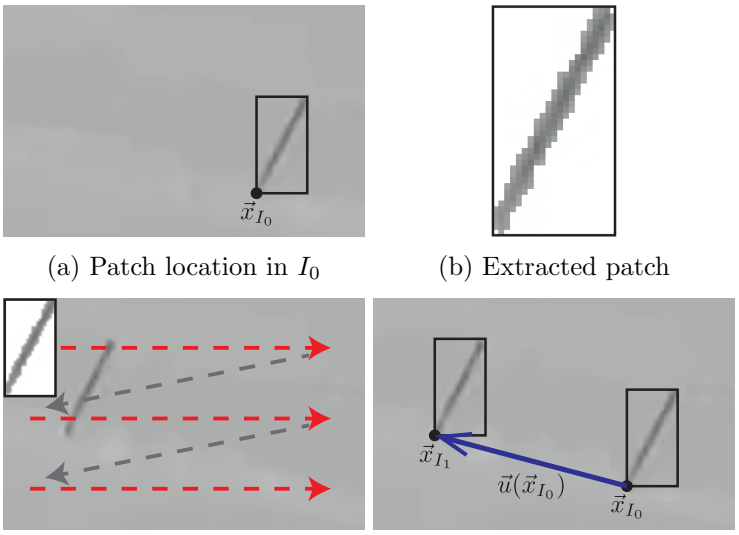
where $I_0(i)$ and $I_1(i)$ denote the i^{th} intensity value of the corresponding patch and n is the total number of patch pixels. We prefer the MAE over the *mean squared error* (MSE), because it is more robust to strong outliers. The sliding window position in image I_1 with the smallest error is set as the matching patch location. If the error value e_{MAE} of the best matching candidate does not satisfy the minimum matching distance threshold, then all pixels of the patch in image I_0 are marked as *not matched*. This in most cases happens if the object is occluded in image I_1 or if the the patch intensities change significantly due to illumination variations.

After the matching patch is found, the displacement vector $\vec{u}(\vec{x}_{I_0})$ is easily determined by computing the difference of the two bounding box origin points by

$$\vec{u}(\vec{x}_{I_0}) = \vec{x}_{I_1} - \vec{x}_{I_0}, \tag{3.11}$$

as shown in Figure 3.21. We use $\vec{u}(\vec{x}_{I_0})$ as displacement vector for every pixel of the extracted patch, instead of the previously computed optical flow.

With this method the result of the optical flow algorithm is enhanced, and extended to also handle large motions of small objects. If both matching objects lie within the image bounds, there is basically no limit for the maximal supported displacement distance. The approach is constrained on the visibility of the matching patches in both images. Therefore, we are for example not able to correct erroneous optical flow vectors, if the matching patch cannot be found in the corresponding image. This may be caused by occlusions, deformations or significant illumination changes.



(a) Patch location in I_0 (b) Extracted patch

(c) Sliding window approach (d) Displacement vector $\vec{u}(\vec{x}_{I_0})$

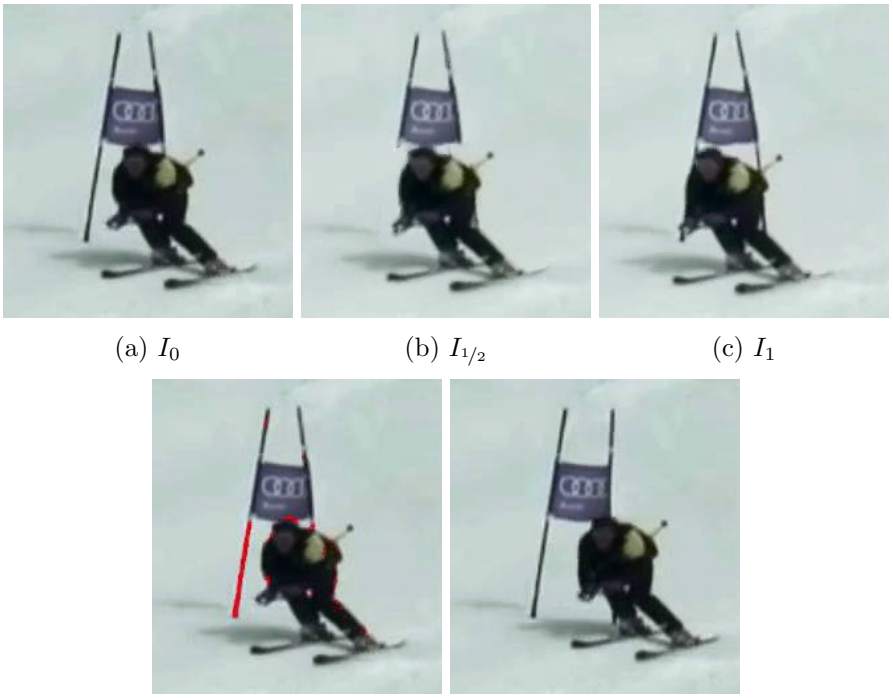
FIGURE 3.21: An example for the sliding window-based displacement computation of the wrong interpolated fence pole shown in Figure 3.20. (a) The bounding box of the object in image I_0 is defined by the warp error mask. (b) Based on this mask we can easily extract the patch. Only the non-white pixels are taken into account for further calculations. (c) With the sliding window approach we search for the matching object in image I_1. For every position of the patch, the error value e_{MAE} is computed. The matching object lies at the location with the lowest error value. (d) If a matching patch is found, the displacement vector $\vec{u}(\vec{x}_{I_0})$ is calculated with the difference of the bounding box origin coordinates.

To increase the robustness against such events, we make use of the temporal consistency assumption of motion vectors. Due to video recording frame-rates of 24 fps or higher, motion vectors change only marginally over the image sequence. Therefore we are able to approximate the displacement of an object from image I_0 to I_1 by calculating its motion from image I_{-1} to I_0. The advantage of this approach is that the target object which is occluded in image I_1 may be visible in image I_{-1}, as shown in the example in Figure 3.22.

The same sliding window approach as used before in image I_1 is utilized to search for the extracted patch of image I_0 in I_{-1}. If a matching patch is found the displacement vector describing the motion from image I_{-1} to I_0 is calculated in a similar way as shown in equation (3.11). But since we now want to predict the position of the object in the subsequent image I_1, as shown in figure Figure 3.23, we have to slightly modify it

$$\vec{u}(\vec{x}_{I_{-1}}) = \vec{x}_{I_0} - \vec{x}_{I_{-1}}. \tag{3.12}$$

With this approach we are able to estimate the position of an object in image I_1, although it is not visible or just visible in a significantly modified appearance in comparison to image I_0. The motion approximation is sufficient if the sampling rate is high enough



(a) I_0 (b) $I_{1/2}$ (c) I_1

(d) Warp error (e) I_{-1}

FIGURE 3.22: (a,c) The cropped input images with a moving skier. In the first image the pole of the ski-gate is visible, whereas in the second it gets occluded by the skier. Therefore the optical flow algorithm cannot estimate the correct displacement of the pole. (b) This leads to an intermediate frame with visible interpolation errors, due to the wrong motion vectors along the pole. (d) The occurring image errors are correctly indicated by the warp error mask. (e) Due to the temporal motion consistency, we can retrieve movement information from neighboring images, where the pole is visible. This is done by computing the pole displacement from image I_{-1} to I_0, which is an approximation of the motion vectors of the image pair I_0 and I_1.

(a) I_{-1} (b) I_0 (c) I_1 (d) Motion vector

FIGURE 3.23: With this example we want to demonstrate the temporal consistency of motion vectors. Based on the image pair I_0 and I_1, the optical flow for the left ski-gate pole cannot be estimated sufficiently, because it is only visible in the first of the two images. The position of the pole in image I_1 can be approximated, by also taking the neighboring frame I_{-1} into account. This is achieved by calculating the displacement vector $\vec{u}(\vec{x}_{-1})$ for the image pair I_{-1} and I_0 and utilizing it for the patch pixels in image I_0. This is illustrated in (d), where the three bounding boxes of the pole are overlayed on image I_1. The position of the occluded pole is given by the displacement vector $\vec{u}(\vec{x}_{-1})$.

in relation to the occurring motion changes. For object movements which significantly change from frame to frame, the predicted location will deviate from the correct location. The additional search in the neighboring image I_{-1} is only carried out, if no matching patch can be found in I_1. This ensures that in the first place the more accurate method is used to find the displacement. If no matching patch can found in both neighboring images I_{-1} and I_1, then the erroneous optical flow vectors cannot be repaired. Such a scenario occurs if the sought object is only visible in image I_0 or the appearance in both neighboring frames significantly differs.

In general, the patch-based search is prone to errors if the sought object underlies a geometric transformation like rotation or scaling. This can be improved by implementing a rotation and scale invariant matching algorithm, as proposed by Barnes et al. [4]. On the one hand, such a method increases the matching rate, but on the other it is also more likely to obtain false matches. Since this negatively affects the visual quality of the video, we stick to the standard sliding window approach. Furthermore, matching errors are also expected at the occurrence of repetitive structures, due to multiple similar candidate patches. With a high minimum matching distance threshold these false matches can be reduced. However, the threshold should not be too high, to allow minor outliers.

To reduce the computational expense, we limit the search region based on the largest expected displacement, as shown in Figure 3.24. This also decreases the probability of false matches, because less candidates are taken into account. The largest expected displacement is an application specific value, which depends on known parameters like recording frame-rate, sensor resolution, focal length and not predictable parameters like camera motion, object speed and distance of the object to the camera. We approximate this value individually for each dataset, according to an experimental evaluation on the image sequences.

(a) I_0 (b) I_1

FIGURE 3.24: By using a limited search region for the sliding window approach instead of the whole image, the computational expense can be reduced. Furthermore also the likelihood for false matches is decreased, because less candidates are considered in the matching process.

One of the major disadvantages of coarse-to-fine warping-based optical flow algorithms, is that small objects are not supported due to the downscaling in the image pyramid.

Furthermore, also object occlusions and disocclusions disable a reliable optical flow estimation. We address these deficiencies with the presented approach, which is used as an additional post-processing step. We can state the following improvements and strengths of our method:

- **Large displacements**
  Large displacements are handled very well with our approach, independently of the object size. However, we ignore patches with just a few pixels, due to the lack of matching reliability. In general, the maximum object displacement is only limited by the image bounds, but to increase the computational and matching performance we limit it to a predefined value.

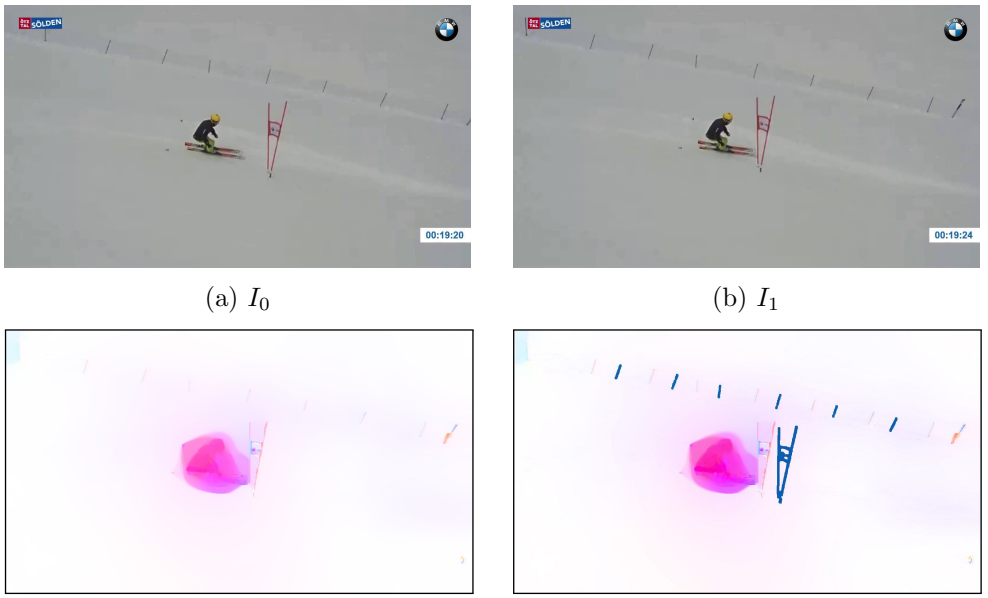
- **Occlusions and disocclusions**
  If an object gets occluded or significantly changes its appearance, we are still able to predict an approximated motion vector, if a matching patch can found in the other neighboring frame. In the worst case, if an object is just visible in one frame or has a significantly differing representation in all three frames, the displacement vector cannot be computed.

Our approach addresses optical flow errors, which cause perceivable artifacts in the interpolated image. Errors in homogeneous regions like for example snow, are not recognized by the warp error, but also not noticeable in the resulting interpolated frames. The optical flow enhancements with our post-processing step are shown in Figure 3.25.

With the corrected flow vectors we are able to compute the intermediate positions of the extracted patches. These objects have to be treated separately in the composition process of the intermediate frame. This is required, since we also approximate displacement vectors of objects, which are actually not visible in image I_1. Therefore, an patch-based video inpainting method is needed, which performs a depth check for such objects, to determine if they get occluded or stay in the foreground.

3.5.2 Inpaint Object

In general, objects with wrong optical flow vectors cause interpolation errors which become visible in the intermediate frame. This problem was addressed in the previous section, by recovering the correct flow vectors. In this section we focus on how to inpaint these wrongly interpolated objects at the proper position. The proposed displacement recovery method is also capable of approximating motion vectors of objects, which are occluded in image I_1 Therefore, the inpaint method must consider the correct depth



(a) I_0 (b) I_1

(c) Huber-L1 optical flow (d) Enhanced optical flow

FIGURE 3.25: An example for the optical flow improvements visualized with the color-coded representation, based on the image pair I_0 and I_1. With the Huber-L1 optical flow the displacements of all fence poles in the background, the ski-gate and also the snow are estimated wrongly, due to the fast camera movement. Beside the snow, which does not introduce perceivable artifacts, these errors are identified and handled by our approach. The improvements are clearly noticeable through the high color saturation at the mentioned objects.

order of objects. If for example a ski-gate gets occluded by a skier in image I_1, the depth order is: skier - foreground; ski-gate - background. This depth order has to be maintained during inpainting in the intermediate frames.

The position where the object needs to be inpainted is computed by linearly scaling the recovered optical flow vectors, as explained in section 3.1. By simply drawing the extracted patch at this position, the depth order may be violated. Therefore, we consider two cases for inpainting:

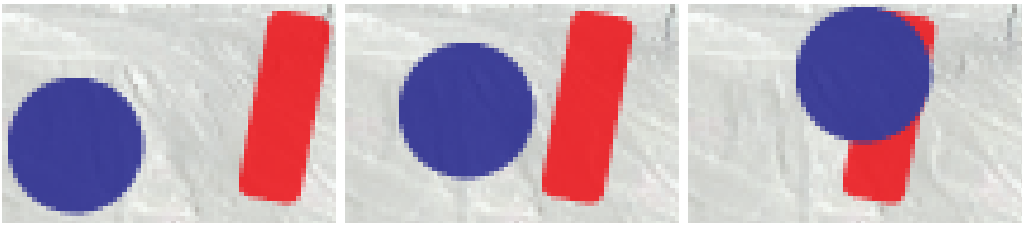
- **Object remains in the foreground**
  If for the patch extracted from image I_0 a matching patch in image I_1 is found, as shown in Figure 3.26, we can conclude that it is also in foreground in the intermediate frames. In this simple case, the whole patch is always drawn as foreground, without considering neighboring pixels during inpainting.

- **Object gets occluded**
  If for the extracted patch from I_0 no matching patch in image I_1 is found, we infer that it is occluded in image I_1. The patch moves from foreground to background in the depth order, as shown in Figure 3.27. This means that its appearance changes within the intermediate frames.

(a) I_0 (b) I_1

FIGURE 3.26: The red colored region represents the object to be inpainted. Since the whole red object is visible in both images, it is also in the foreground in the intermediate frames. Therefore, the extracted patch is inpainted at the intermediate position without any modifications.

We solve this problem by comparing the neighboring pixels of the patch in image I_0 with the underlying pixels of the interpolated image. With this comparison we can detect when and where the occluding object approaches to the extracted patch. This video inpainting algorithm is in detail explained with a simple example in Figure 3.28.

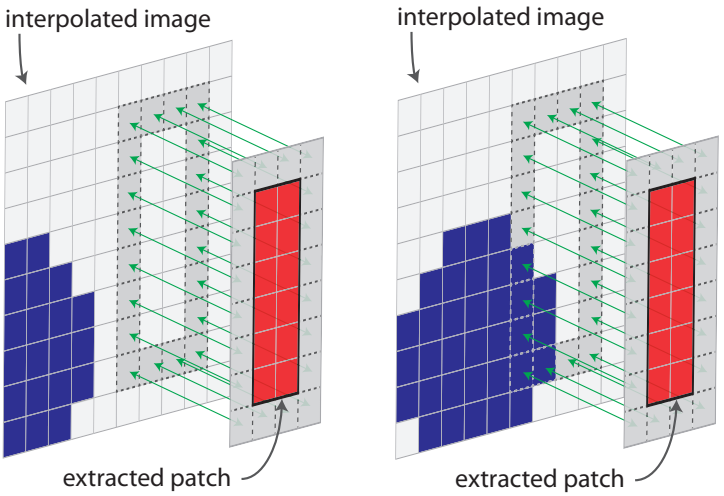


(a) I_{-1} (b) I_0 (c) I_1

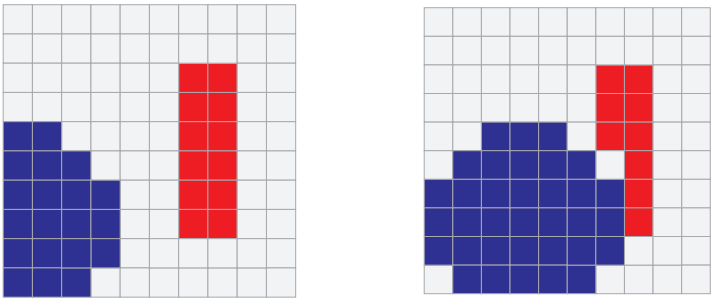
FIGURE 3.27: In this example the red target object is only partly visible in image I_1. This is not enough to be considered as a reliable match. Therefore, the optical flow is approximated by additionally taking the neighboring frame I_{-1} into account. The difficult task is now to inpaint the target region in the intermediate frames properly, so that the blue moving object is in front of the red object.

Our video inpainting algorithm is based on the visibility assumption, where an object is assumed to be visible in the intermediate frames if it is visible in both input images I_0 and I_1. As shown in Figure 3.29 this assumption is not reliable if the recording frame rate is too slow for the occurring object motions.

With the depth check we approximate the occlusion process in the intermediate frames. This method should give the viewer of the resulting video the feeling of proper depth handling. The approximated solution will not always be correct for every inpainted pixel. This is noticeable by taking a close look on single frames of the image sequence. However, when the sequence is played back as video these minor errors do not attract that much attention. As explained in section 3.4.2, temporal consistency is very important for video inpainting algorithms. We satisfy this criterion by using patches extracted from

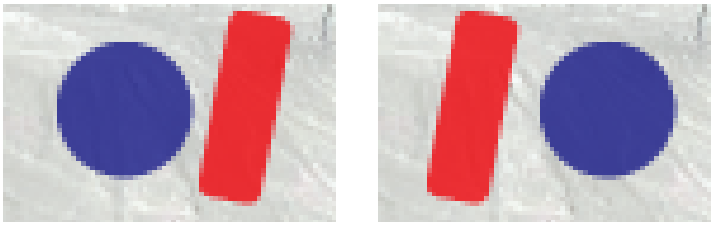


(a) Inpainting: $I_{1/3}$ (b) Inpainting: $I_{2/3}$



(c) Result: $I_{1/3}$ (d) Result: $I_{2/3}$

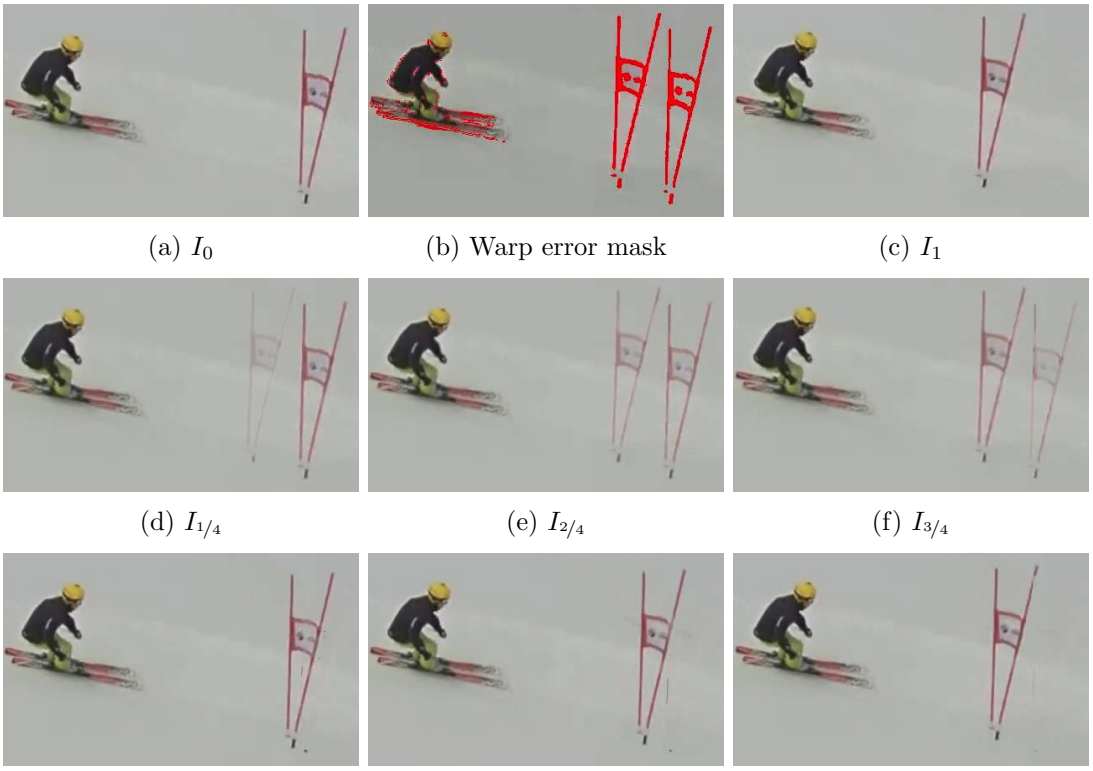
FIGURE 3.28: Interpolation of two intermediate frames with our inpainting approach, based on a similar example as shown in Figure 3.27, where the target object gets occluded. (a) Since the blue moving object is not overlapping with the neighbor pixels of the red target patch, both objects are rendered normally, as can be seen in the resulting interpolated frame (c). (b) The blue object approached the red region and partly occludes it. By comparing the neighbor pixels of the extracted patch with the underlying pixels of the interpolated image, we can detect where the blue object overlaps with the red region. This pixel-wise comparison is visualized with the green arrows. A red pixel is only drawn, if the closest neighbor pixel has a low comparison difference. If the closest neighbor pixel has a blue overlap (high comparison difference), then instead of the red pixel, the the underlying intensity is adopted. The result of this example is shown in (d), where the partly occlusion of the red region is visible.



(a) I_0 (b) I_1

FIGURE 3.29: The depth check becomes unreliable, if the sampling frequency is too low for the occurring displacements, as shown in the image sequence (a) and (b). Our algorithm assumes, that the red target object is in the foreground in the intermediate frames, since it is visible in both images. But in fact we cannot say which one of the two objects is occluding the other, based on the input images I_0 and I_1.

the input images, similar to the exemplar based inpainting approach. In general, the presented video inpainting algorithm yields satisfying results, as shown in Figure 3.30 and Figure 3.31.



(a) I_0 (b) Warp error mask (c) I_1

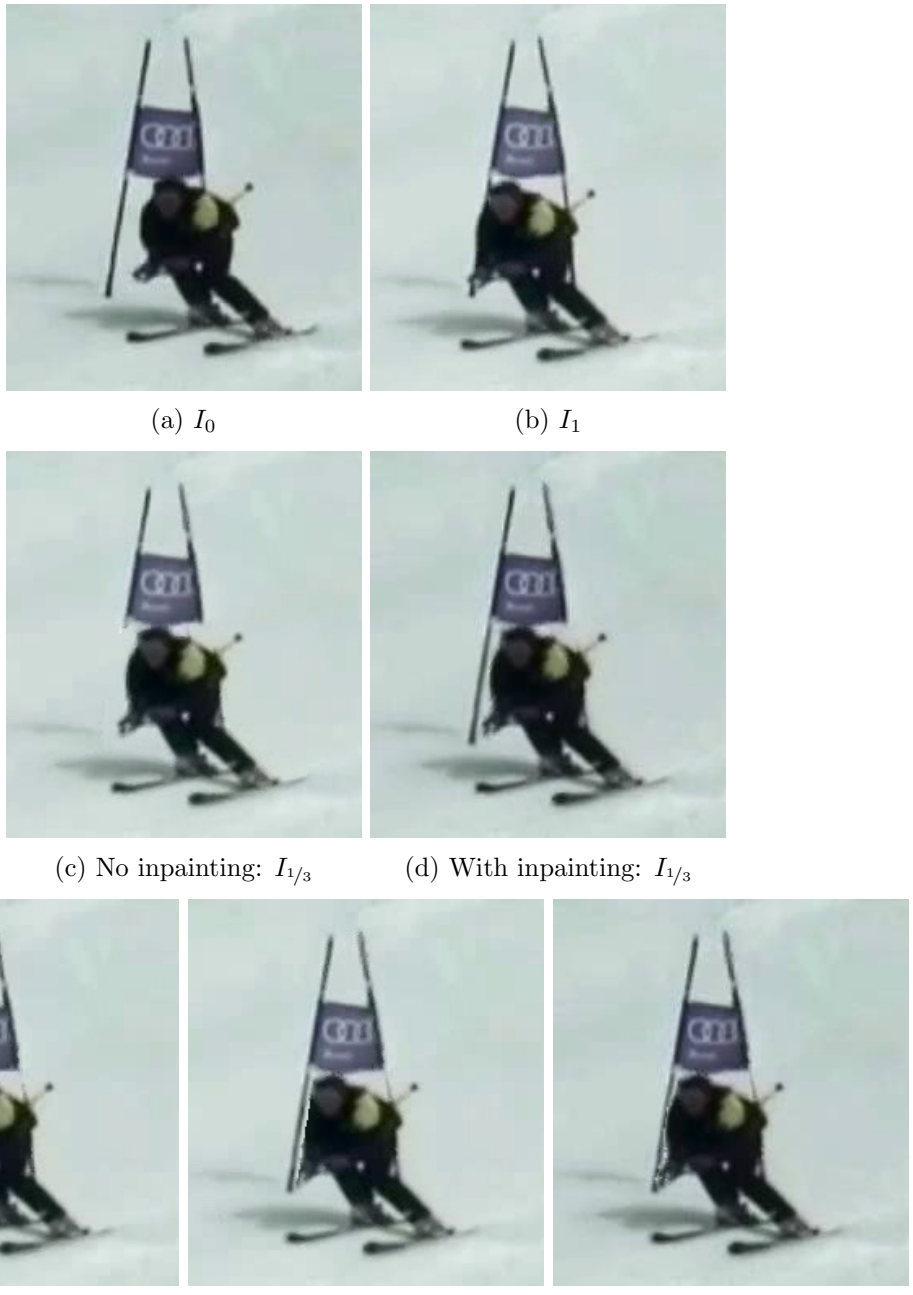
(d) $I_{1/4}$ (e) $I_{2/4}$ (f) $I_{3/4}$

(g) $I_{1/4}$ (h) $I_{2/4}$ (i) $I_{3/4}$

FIGURE 3.30: (a,c) The motion vectors of the ski-gate are not estimated properly, due to the fast moving camera. (b) This is correctly indicated by the warp error mask. (d,e,f) Therefore, the bilinear interpolation yields wrong intermediate frames, since the ski-gate is drawn two times. (g,h,i) With our video inpainting algorithm we are able to recover the displacement of the ski-gate from image I_0 to I_1. The extracted patch (ski-gate) is then inpainted at the three linearly interpolated intermediate positions. This yields a temporal consistent result, where the ski-gate is moving along its displacement vector.

3.6 Summary

To achieve the well known slow motion effect, synthetic intermediate frames are interpolated between every consecutive frame pair. This is done by propagating pixel intensities along motion vectors obtained from optical flow. To fill the holes, which arise due to object occlusions and disocclusions, the optical flow is computed in forward and backward direction. The intermediate frames are interpolated with the bidirectional approach, where the propagated intensities are linearly blended from I_0 to I_1. The bidirectional frame interpolation strongly depends on the optical flow. Therefore, errors in the flow field are visible as artifacts in the intermediate frame, which are measurable with the



(a) I_0 (b) I_1

(c) No inpainting: $I_{1/3}$ (d) With inpainting: $I_{1/3}$

(e) No inpainting: $I_{2/3}$ (f) No depth ordering: $I_{2/3}$ (g) With inpainting: $I_{2/3}$

FIGURE 3.31: (a,b) In this sequence the left pole of the ski-gate gets occluded in image I_1. Therefore its displacement is not estimated correctly. With our presented video inpainting approach, we are able to restore the proper motion vectors for the pole. (c,d) We can clearly see the difference between the bidirectional interpolated frame and the inpainted frame, where the pole gets recovered. (e,f,g) To emphasize the importance of the depth ordering-based inpainting approach we compare the bidirectional interpolated frame, the inpainted frame without considering the depth, and the depth ordered inpainting result. In (f) the pole is recovered at the correct position, which is an improvement over (e), but it is drawn over the skier. This gets fixed by considering the depth order in the inpainting process in (g).

warp error. Based on this error measure, we developed a two-step artifact removal strategy. In the first step, the corrupted optical flow is corrected with a patch-based method. The proposed optical flow enhancement has its strengths over variational algorithms especially at large object displacements, thin moving objects, occlusions and disocclusions. The latter two are achieved with the temporal consistency assumption for motion vectors, where the preceding and subsequent frames are also included in the computation. In the second step, the object-patch is inpainted at the in step one calculated position, by differentiating between two different cases:

- If the whole patch is visible in both input frames I_0 and I_1, then it is inpainted unchanged in the intermediate frame.
- If the patch is only visible in one of the two input images, then it moves from foreground to background or vice versa. We developed an inpainting method which determines where the patch gets occluded in the intermediate frame to maintain the correct depth order of the objects.

Our proposed algorithm improves the interpolation result, where the variational optical flow is not able to estimate the proper displacements.

Chapter 4

Evaluation and Results

The primary goal of this slow motion algorithm is to deliver visually attractive results for a large variety of input sequences. This is considered in the evaluation of our algorithm, where various test-videos with different scenes and movements are analyzed. A single intermediate frame $I_{1/2}$ is interpolated between every frame-pair of the test video. For a sequence of 100 frames with 99 consecutive frame pairs, the resulting slow motion video has 199 frames, which is a slowdown by the factor of approximately 2. Of course also higher slowdowns are feasible with our algorithm. However, the resulting slow motion sequence with multiple intermediate frames is difficult to visualize in this printed work. Depending on the recording frame rate, the test-videos are separated in two datasets, which are evaluated in different ways.

The *Skiline* dataset contains videos provided by the company *Skiline Media AG*. These test video have in common that they were recorded with pan-tilt-zoom (PTZ) cameras at a frame rate of 25 fps. Our frame interpolation algorithm is applied on every image pair of the input sequence. The intermediate frames and the resulting slow-motion video are visually inspected with the primary focus on disturbing artifacts and non-smooth movements.

The videos of the *Ground Truth* dataset were recorded with a hand-held smartphone, at a frame rate of 120 fps. This enables us to generate ground truth data for the intermediate frames, by leaving every second frame of the video out. With the comparison of the interpolated frame with the corresponding ground truth frame, we can measure their similarity and determine where interpolation errors occur.

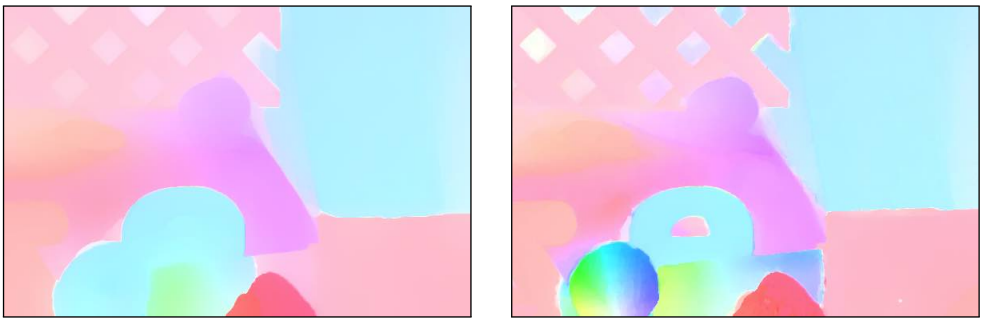
4.1 Optical Flow Library

To estimate the optical flow for each frame pair, we use the *FlowLib v3.0* library ([40], [41]), developed at the TUGraz by the GPU4Vision group. The library offers multiple variational optical flow models, which can be configured by a large set of parameters. It is implemented in C++ and CUDA, and can be executed in a massively parallel fashion on a GPU. This enormously speeds up the optical flow estimation. Binaries of the latest *FlowLib* version are available at the GPU4Vision website[1], which are free to use for academic purposes.

The main parameters of our *FlowLib* configuration used for evaluation of the test videos are listed below:

- **Model: HL1**
  We use the Huber-L1 model with the Huber norm in the regularization term and the L1 norm in the OFC-based data term. The Huber-L1 model is in detail explained in section 2.3.4.3.
- **Iterations: 70**
  The optical flow energy functional is minimized within 70 iterations per warp.
- **Warps: 10**
  At every level of the image pyramid, 10 warps of image I_0 towards image I_1 are performed. After each warp the optical flow is recalculated based on the previous result. This iterative warping leads to an optical flow refinement.
- **Scale-factor: 0.95**
  The two input images I_0 and I_1 are down-scaled to a coarser level in the image pyramid, with a factor of 0.95.
- **λ: 30**
  The balance between the regularization and data term of the energy functional, is controlled with the λ parameter. A low value favors smooth solutions, whereas a high value takes more details into account, as can be seen in Figure 4.1.
- **ϵ_u: 0.01**
  The parameter ϵ_u defines the threshold for the Huber norm. Below the threshold we obtain a quadratic and above a linear penalty.
- **Median Filter: ON**
  The resulting optical flow vectors of a coarse level in the image pyramid are up-scaled to a finer level by using a median filter.

[1] http://gpu4vision.icg.tugraz.at/



(a) $\lambda = 5.0$ (b) $\lambda = 50.0$

FIGURE 4.1: Influence of the balance parameter λ on the optical flow result. (a) A low value yields a smooth optical flow. (b) By increasing λ, also fine details are taken into account. However, a too high value increases the influence of noise on the result. Both images are taken from the FlowLib documentation.

4.2 Skiline Dataset

The company *Skiline Media AG* developed a system, where multiple stationary PTZ cameras track and record a skier along a defined racing track. The automatically composed video is stored online[1], where the skier is able to watch and download it. This application was also adapted for downhill mountain-biking and ski-/snowboard-jumping. With the slow motion effect applied on a short part of the image sequence, the company wants to enhance the viewing experience of the final composed videos. We chose 4 shortened sequences which cover most of the occurring scenarios like skiing, mountainbiking and jumping, to evaluate how well our algorithm performs on their videos.

The camera motion of the *Skiline* test videos is limited to pan, tilt and zoom, due to the type of installed camera-systems. Nevertheless, there are no restrictions regarding the object motion in the scene. All videos were recorded at a frame rate of 25 fps and with a video resolution of 1280 $\times$ 720 pixels. The chosen test sequences contain scene-backgrounds which range from homogeneous snow to highly textured grass. Due to the lack of ground truth data, the evaluation is done by visually inspecting the interpolated frames and the slow motion video. It is very important, that the slow motion movements look naturally, as if they were recorded at a higher frame rate. Therefore, we concentrate on the occurrence of disturbing artifacts, not correct interpolated objects or non-smooth movements. Also the optical flow consistency and warp error measures are taken into account within the evaluation.

[1] http://www.skiline.cc/skimovies/spot/

4.2.1 Test-Sequence: *Ski-Planai*

This test video shows a moving skier, passing a ski-gate in the front. The background is static, beside an another moving skier in the upper left. The video was recorded with a static camera, therefore, the optical flow of the background scenery is zero. Figure 4.4 shows the input images and resulting interpolated images side by side. The main difficulty of this video is the occlusion and disocclusion of the thin ski-gate poles by the skier. Figure 4.2 shows the consistency and warp error for these frames. The occurring errors cause disruptive artifacts in the intermediate frames of the bidirectional interpolation. By applying our proposed inpainting algorithm as post-processing step, the introduced artifacts are successfully removed, as shown in Figure 4.3.

Overall, the final interpolated frames in Figure 4.4 contain no major disturbing artifacts. By enlarging a single interpolated frame, some minor pixel-scale artifacts become visible, which vanish if the sequence is played back as video. The slow motion of the skier during playback is very smooth and natural.

(a) Consistency error

(b) Warp error

FIGURE 4.2: Both, the flow consistency and the warp error indicate that flow errors occur at the occlusion of the ski-gate poles. The two images on the left show the error measures for the occlusion of the first ski-gate pole, and the two images on the right for the second pole. While the consistency error also measures flow deviations in the homogeneous snow region, the warp error just focuses on the perceivable error along the ski-gate pole.

(a) Bidirectional frame interpolation

(b) Proposed frame interpolation approach with inpainting

FIGURE 4.3: Comparison of the resulting intermediate frames, which were calculated with and without our inpainting algorithm. (a) Parts of the ski-gate poles are drawn incorrectly in the bidirectional interpolated frame. This is caused by wrong optical flow vectors, due to the occlusion by the skier. (b) These disturbing artifacts are removed, by applying our proposed inpainting algorithm. Minor small-scaled artifacts are still visible in the enlarged frame. However, they do not attract that much attention during video playback with 25 fps.

4.2.2 Test-Sequence: *Mountain-bike*

The *Mountain-bike* sequence shows a cyclist jumping over a hill, as shown in Figure 4.6. The scene background is highly textured and dominated by grass, stones and dirt. Due to the fast camera panning in combination with the relatively long exposure time, a decent amount of motion blur is introduced. This eliminates fine details and leads to a poor visual quality of the test video. The thin elements of the mountain bike pose a problem for the optical flow algorithm. The part of the front wheel with the hidden rim is very hard to distinguish from the background. As in detail analyzed in Figure 4.5, this leads to interpolation errors caused by wrong motion vectors along the wheel. Since the upper part of the wheel is not clearly distinguishable from the background, also our inpainting algorithm fails to remove the introduced artifacts. Beside this error, which occurs in the last intermediate frame, no further disturbing artifacts are noticeable in the resulting slow motion sequence. The highly textured background and the cyclist are interpolated properly and have a smooth motion during video playback.

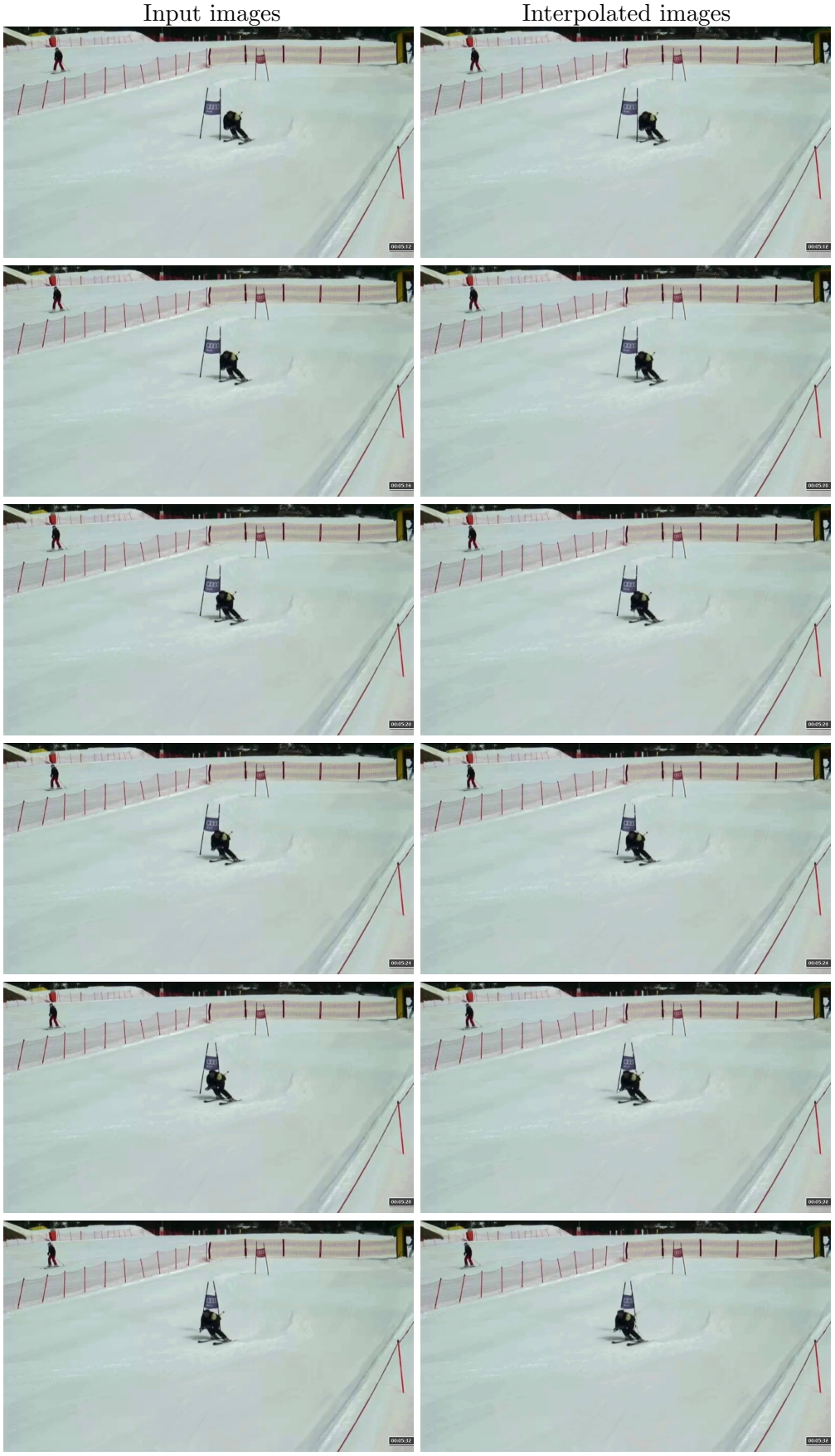


FIGURE 4.4: Resulting *Ski-Planai* sequence

(a) $I_{1/2}$

(b) I_0

(c) I_1

(d) Forward flow superimposed with I_0

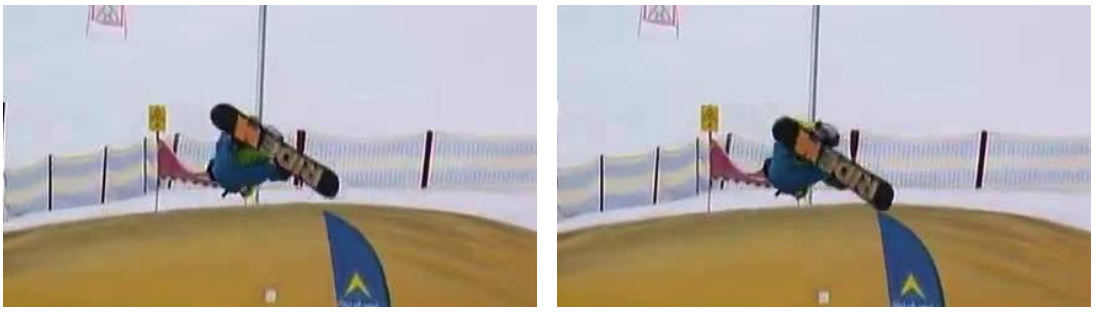
(e) Consistency error

FIGURE 4.5: (a) A noticeable interpolation error at the front wheel of the bike occurs in the last intermediate frame of the *Mountain-bike* sequence. (b,c) In image I_0 and I_1 the white rim is not clearly visible in the upper half of the front wheel, wherefore it is not distinguishable from the background. (d) This leads to a wrong estimated flow along the wheel. By superimposing the color coded forward flow with the input image I_0, we can clearly see that a part of the front wheel is covered with motion vectors of the background texture. (e) The occurring error is also indicated by the flow consistency measure.

FIGURE 4.6: Resulting *Mountain-bike* sequence

4.2.3 Test-Sequence: *Snowboard-Jump*

This test video shows a snowboarder, which is performing a back-flip, as depicted in Figure 4.8. He is rotating backward around his own axis. Under the boarder a landing platform, the so called *BigAirbag* is visible. Regarding the camera motion, a zoom-in is noticeable during the whole image sequence. A minor artifact is perceivable when the background of the snowboard changes from snow to the landing platform. This error is analyzed and shown enlarged in Figure 4.7. Except of this artifact, which has only a small impact on the final result, the intermediate frames are interpolated accurately. The slow motion video with the rotating snowboarder is smooth without any disturbing artifacts. The good slow motion result is explained with the moderate camera motion in combination with the compact type of moving object. In comparison to a skier with thin skies and sticks or a cyclist with thin wheels, the shown snowboarder is less prone to errors, regarding the optical flow estimation.



(a) I_0 (b) I_1

(c) $I_{1/2}$

FIGURE 4.7: At the transition of the snowboard from snow to the *BigAirbag*, a minor artifact is visible in the intermediate frame. Since the snowboard in I_1 occludes a part of the landing platform, the displacement vectors in this region are not reliable, which leads to the highlighted artifacts. However, due to its small size in comparison to the full image, this error does not attract attention during video playback.

FIGURE 4.8: Resulting *Snowboard-Jump* sequence

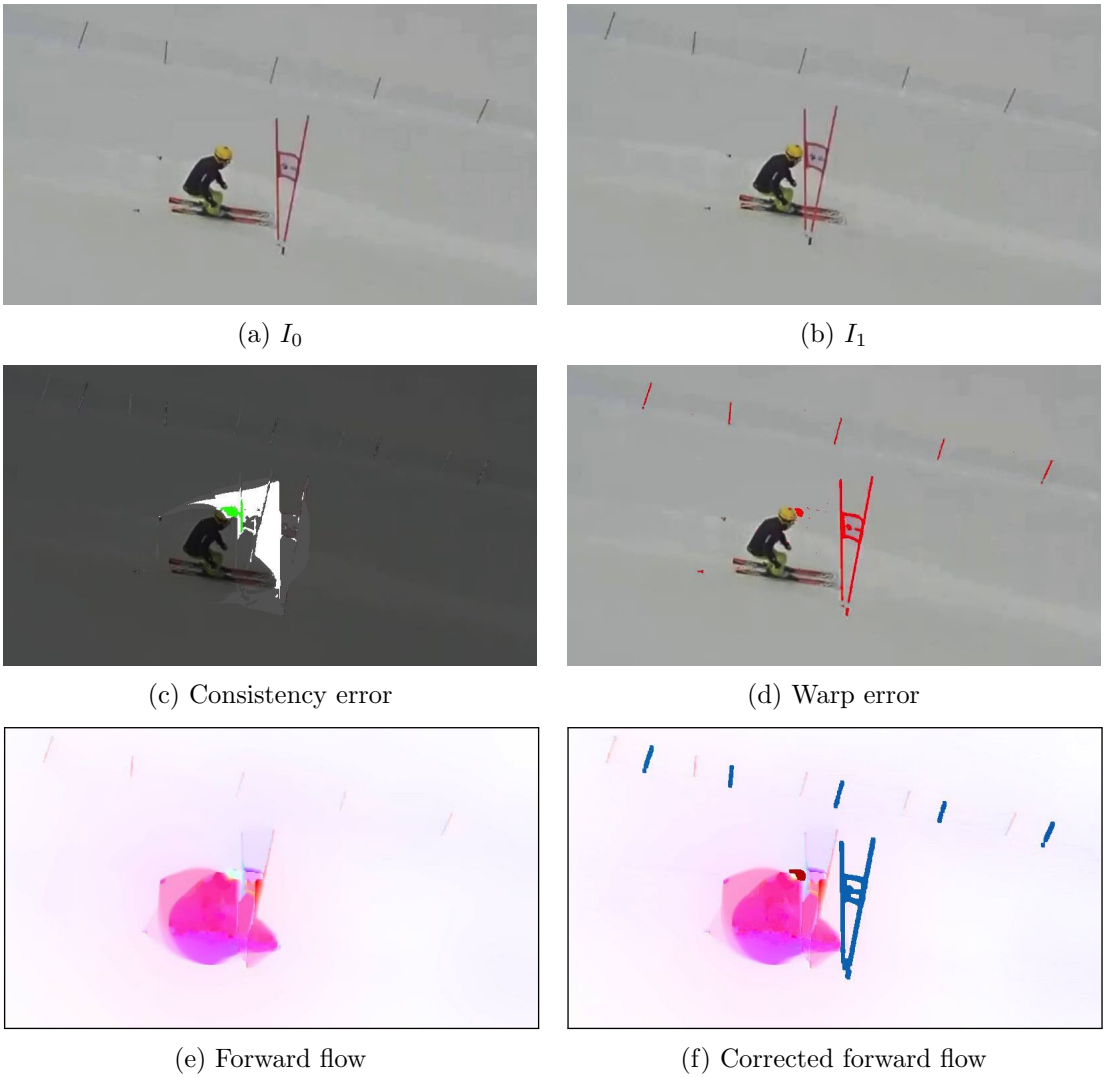
4.2.4 Test-Sequence: *Ski-Soelden*

In the *Ski-Soelden* video, a fast moving skier is passing a ski-gate in front of a poorly visible fence, as shown in the image sequence in Figure 4.11. Between these mentioned objects, weakly textured snow is visible everywhere in the image. The skier appears to be nearly stationary, due to the fast camera movement in the same direction. Therefore, the ski-gate and the fence poles are the apparently moving objects in the image, with displacements of about 70 pixels. This image sequence represents a very challenging problem for the variational optical flow approach. It is impossible to calculate a reliable optical flow for the homogeneous snow region, due to the poor data term. Furthermore, also the displacements of the ski-gate and fence poles cannot be estimated properly, since they are too thin to be supported by the coarse-to-fine warping scheme. Therefore, the snow region is filled-in with wrong neighboring flow vectors. Our patch-based algorithm, explained in section 3.5.1, is able to enhance the variational optical flow result, by correcting the displacement vectors of the patches masked by the warp error. The variational optical flow and the enhanced solution are compared in Figure 4.9. With the corrected optical flow, also the frame interpolation yields better results. A comparison of intermediate frames which were computed with and without our inpainting algorithm is shown in Figure 4.10. The improvements in the frame interpolation gained with our approach are clearly visible. The final slow motion sequence is plotted in Figure 4.11. Due to the corrected optical flow, the movements of the ski-gate and the fence poles appear to be very smooth. if the image sequence is played back as video.

4.3 Ground Truth Dataset

The *Skiline* dataset limits us to a subjective visual evaluation of the interpolated frames, due to the lack of ground truth data. Therefore, we decided to record test sequences, where for every interpolated frame a ground truth image is available. This was accomplished by using a smart-phone camera (Apple iPhone 5s), which is capable of recording videos at a frame rate of 120 fps. The test sequences are composed, by utilizing only every second frame of the initial 120 fps video. The skipped images are used as ground truth data for the interpolated frames $I_{1/2}$.

As test scenes we have chosen sport activities like football, jumping or skateboarding, where fast challenging movements occur. In contrast to the *Skiline* dataset, the main moving objects are closer to the camera, and therefore appear bigger on the recorded frames. This is an important consideration, since near objects in comparison to far objects have larger displacements on the image plane. Except of the *Jump* sequence, the



(a) I_0 (b) I_1

(c) Consistency error (d) Warp error

(e) Forward flow (f) Corrected forward flow

FIGURE 4.9: Optical flow correction based on the warp error mask. (e) Despite for the skier, the variational optical flow algorithm yields a completely wrong result for the input image pair I_0 and I_1. (c) Due to the incorrect forward and backward flow result, the flow consistency error cannot measure the occurring errors reliably. (d) The warp error mask provides the foundation for the optical flow correction, by indicating wrong flow vectors, which cause a perceivable interpolation error. (f) With our patch-based approach, the correct displacements for the indicated objects are regained. The snow still has erroneous optical flow vectors of nearly zero. However, these errors are not perceivable in the intermediate frame.

(a) Bidirectional frame interpolation

(b) Proposed frame interpolation approach with inpainting

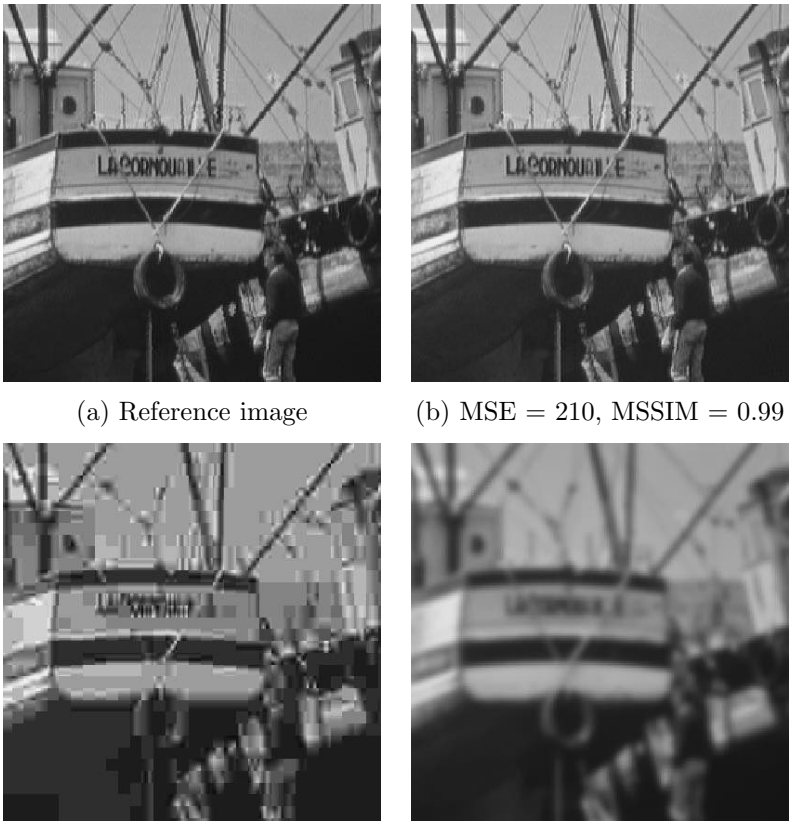
FIGURE 4.10: Comparison of the resulting intermediate frames, which were calculated with and without applying our inpainting approach. (a) Due to wrong optical flow vectors the bidirectional frame interpolation yields completely insufficient intermediate frames. (b) By applying our inpainting algorithm, the correct displacements of the ski-gate and the fence poles are regained. This leads to a clearly visible improvement of the interpolated intermediate frames. The depth order of the skier and the inpainted ski-gate is preserved, since the ski-gate is drawn on top of the skier. However, in the bottom right interpolated frame the helmet of the skier is visible through the flag of the ski-gate. This error occurs, because the white flag does not cause a measurable warp error, due to the white snow in the background. Therefore, the flag is not part of the warp error mask and thus is not inpainted.

smart-phone was hand held during video recording, wherefore the camera movements are not restricted. Due to the separated ground truth images, the frame rate of the input videos is reduced from 120 fps to 60 fps. Equal to the *Skiline* dataset, the video resolution amounts to 1280 $\times$ 720 pixels. However, the videos of the *Ground Truth* dataset feature much more details and sharper images.

Beside the subjective evaluation, with focus on disturbing artifacts and non-smooth movements, also a ground truth comparison is carried out for this dataset. With the difference of the interpolated intermediate frame and the corresponding ground truth frame, the pixel intensity-based interpolation errors are measured. Standard metrics to compare the distance between a reference frame and a modified frame are the mean squared error (MSE) or the peak signal-to-noise ratio (PSNR). The PSNR metric is mostly applied in the field of image or video compression, to measure the quality of compression codecs. It is appropriate for tasks where the pixel-wise similarity of two images is of interest. However, it does not correlate with the perceived visual quality

FIGURE 4.11: Resulting *Ski-Soelden* sequence

[35]. Wang et al. [37] proposed the mean structural similarity (MSSIM) index, which is a commonly applied metric for applications, where the subjective similarity of two images is more important then the pixel-wise comparison. By measuring the degradation of the structure instead of the pure pixel intensity error, the MSSIM index behaves similar to the human eye perception. It ranges from 0.0 (completely different images) to 1.0 (exactly the same images). A comparison of the MSE and the MSSIM metric is shown in Figure 4.12. It is clearly visible that the strongly degraded images yield low MSSIM values, whereas the MSE values stay constant. Many improved MSSIM-based metrics



(a) Reference image (b) MSE = 210, MSSIM = 0.99

(c) MSE = 210, MSSIM = 0.71 (d) MSE = 210, MSSIM = 0.69

FIGURE 4.12: Comparison of the MSE metric and the MSSIM index. The MSE is constant for all degraded images (b),(c) and (d). Similar to the human visual system the MSSIM index yields a high value for the similar looking image (b), and low values for the strongly degraded images (c) and (d). These images are taken from [37].

were developed in recent years ([36] [34]). An evaluation on these image similarity measurement approaches was done by Zhang et al. [45]. According to their results, the feature similarity (FSIM) metric of Zhang et al. [44] correlates the best with the human perception. It is based on the assumption that the human visual system understands an image mainly according to its low-level features. The features considered by the FSIM metric are the phase congruency and the gradient magnitude. Due to its robust performance and good results in the above mentioned evaluation, we use the FSIM metric to measure the interpolation quality of our algorithm.

4.3.1 Test-Sequence: *Football*

In this test video a person is dribbling a football, as shown in the shortened sequence in Figure 4.15. The whole test sequence has 84 input frames, which results in 167 frames in the final slow motion video. The football player is standing on a texture-rich turf and is surrounded by a forest in the background. Due to the direct sunlight, a lot of hard shadows are visible in the scene. The ball and the feet of the sportsman are the main fast moving objects, with displacements of up to 31 pixels.

From the visual perspective, no major disturbing artifacts are perceivable in the interpolated frames. By taking a closer look, some minor differences are notable along the border of the feet, where occlusions and disocclusions occur. These deviations are also indicated by the interpolation error in Figure 4.14, where the interpolated image is subtracted from the ground truth image. Nevertheless, these errors are only marginally noticeable in the intermediate frames, due to the highly textured background. With excellent FSIM values ranging from 0.988 to 0.997, the subjective good result is confirmed by the feature similarity measure, as shown in Figure 4.13. It is noticeable, that the similarity index slightly decreases in the last third of the sequence. This is caused by interpolation errors, which arise due to the increased camera motion. Since these errors mainly occur in the highly textured grass, they have not a negative impact on the visual quality of the final slow motion video. The slowed down movements appear very smooth during video playback.

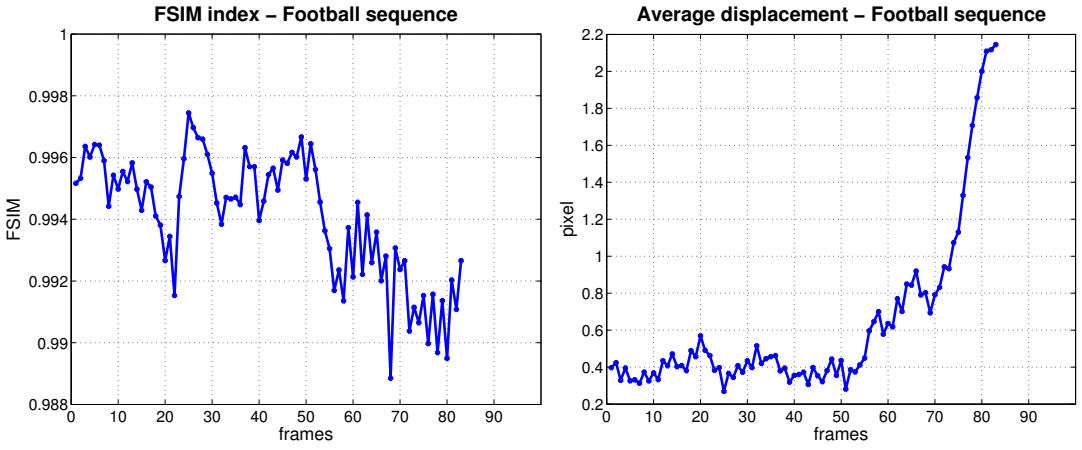
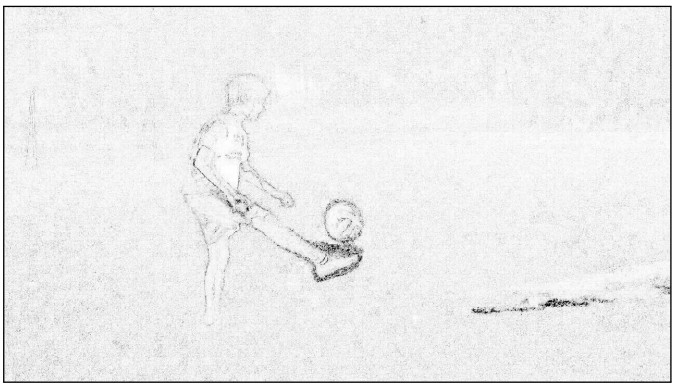


FIGURE 4.13: The interpolated frames and the corresponding ground truth frames have a large feature similarity, with an average value of 0.994. The slight decrease at the end of the sequence is explained with the increasing average displacement, which is caused by camera panning.



(a) Interpolation error

(b) Ground truth frame

(c) Interpolated frame $I_{1/2}$

FIGURE 4.14: The interpolation error is calculated by computing the difference of the interpolated frame and the ground truth frame. For a better visibility, the result of the subtraction is inverted. Due to the rich texture, only a few minor interpolation errors are perceivable in the interpolated frame.

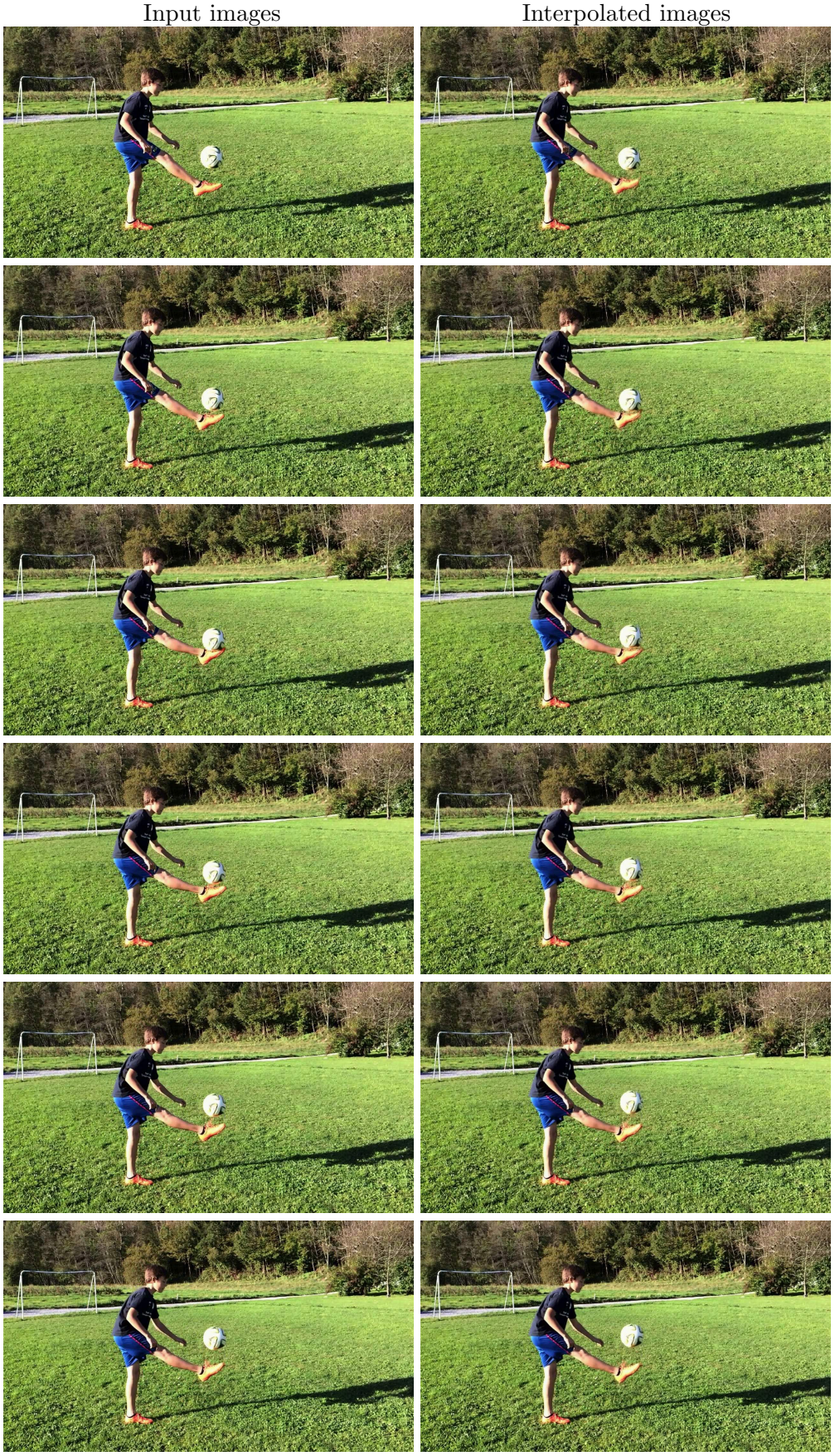


FIGURE 4.15: Shortened *Football* sequence

4.3.2 Test-Sequence: *Jump*

For this test video a person was recorded, while jumping down a hill. The background is a blue sky with slowly moving scattered clouds. A shortened sequence of the whole video with 98 frames is shown in Figure 4.19. The person reaches displacements of about 66 pixels, whereas the movement of the camera is negligible, since it was mounted on a tripod.

In the first half of the image sequence, the FSIM index is ranging from 0.988 to 0.997. These intermediate frames are very well interpolated, with only a few minor artifacts, at the border of the jumping person. However, the second half has slightly worse feature similarity values down to 0.977. The reason for this are the large displacements at the end of the sequence, as shown in Figure 4.16.

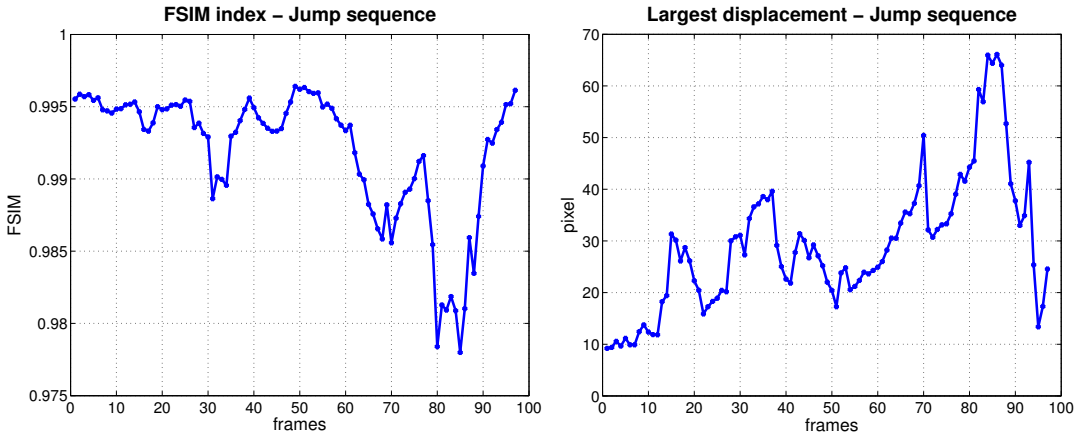


FIGURE 4.16: The interpolated frames of the *Jump* video have very good FSIM values, except of 20 frames at the end of the sequence, where the feature similarity decreases. This behavior inversely correlates with the *Largest displacement* curve, where the largest occurring displacement is plotted for every frame. We can conclude that the similarity index decreases, due to the interpolation errors caused by very fast motion.

Especially the right hand is moving very fast, with displacements of about 66 pixels. In combination with its relative thin shape, this poses a challenging problem for the optical flow algorithm. As shown in Figure 4.17, the bidirectional interpolation fails to interpolate the moving hand properly, due to the wrong optical flow vectors. This leads to strongly disturbing artifacts, which are visible in the interpolated image. With our proposed algorithm, we are able to compute the large displacement of the hand and inpaint it at the proper position.

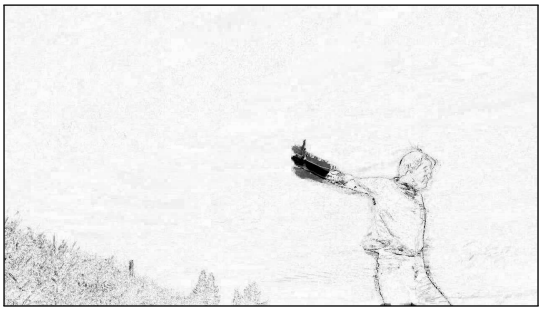
The transformation of the hand from frame I_0 to I_1 mainly consists of a translation and a rotation. Our sliding window-based algorithm only supports translations, to be more robust against similar repetitive objects. Therefore, the patch of the hand is inpainted with a slightly wrong rotation. However, this approximation is a huge improvement over

the standard bidirectional approach, and is sufficient for moderate slow motion videos. A comparison of the ground truth frame and the interpolated intermediate frame is shown in Figure 4.18. The final slow motion video features smooth movements, due to the corrected optical flow. In moderate slow motions, the wrong rotation of the inpainted hand is not noticeable during video playback. By increasing the number of interpolated images between each frame-pair, to achieve higher slowdowns, this error becomes increasingly perceivable.

(a) I_0

(b) I_1

(c) Bidirectional interpolation



(d) Interpolation error

(e) Forward flow

(f) Corrected forward flow

FIGURE 4.17: With the bidirectional interpolation, the fast moving hand is not interpolated properly, due to the wrong optical flow. This leads to interpolation errors, which become clearly visible when subtracting the interpolated image with the ground truth image. We are able to compute the large displacement of the hand, by applying our proposed method as post-processing step. The result of the inpainting algorithm in comparison to the ground truth is shown in Figure 4.18.

(b) Ground truth frame

(c) Interpolated frame $I_{1/2}$

FIGURE 4.18: A comparison of the ground truth frame and the corresponding interpolated frame. It can be observed that the hand is not inpainted exactly, due to its wrong rotation. However, for slow motion videos with moderate slowdowns, this is a good enough approximation.

4.3.3 Test-Sequence: *Skateboard-Trick*

This test sequence shows a skateboard-trick, where the skateboarder tries to land on a rotating board. Due to the direct sunlight, hard shadows of the feet and the skateboard are visible on the asphalt, as can be observed in Figure 4.22. These shadows pose a huge problem for the the motion estimation, because the bright textured asphalt is nearly static, whereas the dark shadows are moving. At the beginning and end of the video, the measured feature similarity index has excellent values in the range of 0.988 to 0.997. In the middle of the sequence the index clearly decreases to a minimum value of 0.926, as shown in Figure 4.20. This is on the one hand caused by very large displacements of about 77 pixels, and on the other hand by the moving shadows of the skateboard. As shown in Figure 4.21, wrong optical flow vectors are estimated for the moving shadows, due to optical flow constraint violations. This results in interpolation errors in this large region, and leads to a lower similarity index. By looking at a single interpolated image, the introduced errors are not clearly noticeable, due to the dark texture. However, if the slow motion video is played back, these artifacts become disturbing, since the

FIGURE 4.19: Shortened *Jump* sequence

normally static asphalt texture is moving. Despite these errors, the intermediate frames are interpolated properly, and the rotating skateboard and the jumping feet are moving smoothly in the slow motion video.

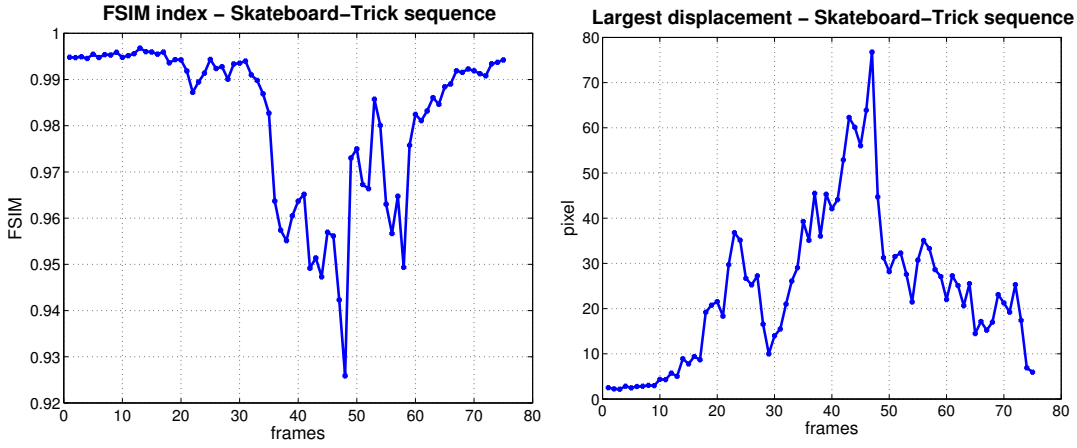
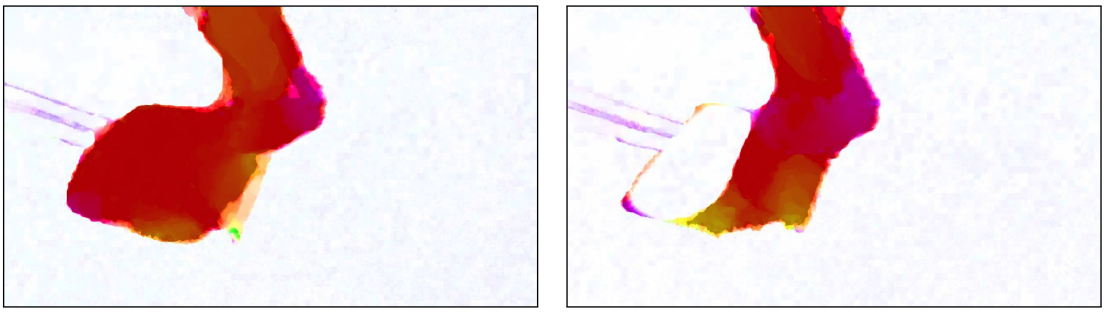


FIGURE 4.20: The FSIM index decreases from very good similarity values to a minimum of 0.926 in the middle of the sequence. This is on the one hand caused by fast moving objects, with displacements up to 77 pixels. On the other hand, also the moving shadows have a negative impact on the interpolated image. As shown in Figure 4.21, the optical flow is not estimated properly in that region, which leads to interpolation errors.

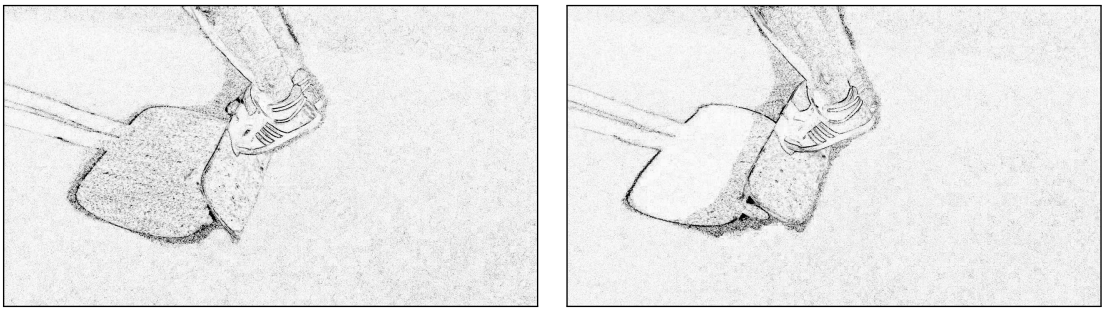
4.4 Summary

The evaluation of our video frame interpolation algorithm was done on multiple test sequences, with different types of scenes and movements. Depending on their properties, the test videos were divided in two separate datasets. The *Skiline* dataset has a frame rate of 24 fps, wherefore no ground truth data is available. The videos were evaluated by visually inspecting the interpolated frames and the slow motion video. Furthermore, also the flow consistency and the warp error were considered for the evaluation. In all these test sequences the moving objects are relatively small. Therefore, the frames were cropped and enlarged for the error analysis and interpolation result comparison. We have shown that our proposed algorithm is capable to enhance the variational optical flow. The improvements are measurable for large movements of thin objects (*Ski-Soelden*), and also for occluded or disoccluded objects (*Ski-Planai*). Both scenarios are in general a challenging problem for variational optical flow algorithms. With the corrected optical flow, also the interpolation results are improved, as shown with side-by-side comparisons. We also demonstrate the limits of our algorithm (*Mountain-bike*, *Snowboard-Jump*), where the patch with wrong flow vectors is too thin or not clearly distinguishable from the background, wherefore the result is not improved.



(a) Forward flow

(b) Interpolated images



(c) Interpolation error

FIGURE 4.21: A comparison of the estimated optical flow and the interpolation results, for the case where the skateboard shadow is moving (left column) and nearly static (right column). If the shadow boarders are moving, the whole shadow is assumed to move in that same direction too, although the underlying texture of the asphalt is static. This leads to interpolation errors, which have a negative influence on the similarity index, due to the large skateboard shadow. In the case where the shadow is nearly static, the optical flow in this region is estimated correctly.

To be able to measure the interpolation quality, we decided to develop a *Ground Truth* dataset. In addition to the subjective visual evaluation, also the feature similarity index and the interpolation error was measured for these videos. We have shown that in scenes with highly textured backgrounds, the interpolation errors appear less distracting (*Football*, *Skateboard-Trick*), in comparison to homogeneous regions like snow. Furthermore, we can say that image pairs with large object displacements, have a weaker feature similarity, due to the higher amount of occluded and disoccluded pixels. Our algorithm is able to enhance the optical flow of very fast moving objects (*Jump*), and improve the interpolation result over the standard bidirectional interpolation. In the same test case we

FIGURE 4.22: Shortened *Skateboard-Trick* sequence

have shown, that our sliding window-based algorithm supports only patch translations and no rotations. Therefore, the inpainting algorithm is on the one hand less precise, but on the other hand still a good approximation for videos. This restriction is needed, to lower the amount of false matches in scenes with similar repetitive objects.

Overall we can say that the performance of the bidirectional interpolation in combination with the variational optical flow in general yields good results. However, for scenes with large displacements or occlusions and disocclusions interpolation errors occur, which are very distracting during video playback. In such cases our proposed inpainting algorithm is able to enhance the optical flow and remove the artifacts in the interpolated frames.

Chapter 5

Conclusion and Outlook

In this work we presented an optical flow-based frame interpolation algorithm, for the computation of smooth slow motion sequences. The primary goal of our approach is to deliver artifact-free videos with authentic object movements.

5.1 Conclusion

The motion vector field, which describes the apparent object motion within two input images, is the foundation of the presented algorithm. We use the variational Huber-L1 optical flow model, with a coarse-to-fine warping scheme for large displacements, to obtain the required motion data. With its fill-in effect, this algorithm yields dense results also for sequences with large homogeneous regions.

Based on the optical flow, the intermediate frames are interpolated by linearly propagating pixel intensities along their corresponding motion vectors. We have shown that the results of the forward flow-based frame interpolation are not sufficient, due to unreliable disocclusion handling. Better interpolation results are obtained by using the bidirectional approach, which takes the motion field of the forward and backward flow into account. The huge benefit of this approach is a more accurate filling strategy for disocclusion holes, with vectors of the opposite flow direction. However, this method doubles the computational cost, due to the additional backward flow estimation.

The bidirectional frame interpolation yields pleasant results, on condition of a correct optical flow. Errors in the motion estimation in general lead to artifacts in the interpolated frame. We identify these errors, with the image-based warp error measure. To remove the introduced artifacts, we developed a two-step strategy. In the first step we perform a patch-based optical flow enhancement, to calculate the correct displacements

for the wrongly interpolated patches. In the second step these patches are inpainted at the corrected locations, while maintaining the proper depth order of the image. This is crucial, since also displacements of occluded objects are estimated in the optical flow enhancement step.

The proposed algorithm was evaluated with several different test videos. As the results suggest, the frame interpolation algorithm is applicable for a wide range of scenes, including homogeneous surroundings like snow or highly textured environments like grass. We have shown, that the flow enhancement algorithm improves the variational optical flow result, especially for fast moving objects, occlusions or disocclusions. Furthermore, our inpainting approach successfully removes the occurring image artifacts, and inpaints the patches at the proper position by considering the correct depth order. With a feature similarity index of about 0.99, the overall performance of the frame interpolation algorithm is very good. However, not all noticeable image artifacts were removed. Wrong interpolated objects, which are just a few pixels thin or very hard to distinguish from the background, pose a problem for our algorithm. Either they are not recognized as artifact from the warp error, or the patch area is too small, to perform reliable patch matching.

5.2 Outlook

The patch-based optical flow enhancement approach still has potential for improvement. Our current sliding window implementation only supports pure translations for patch matching. This restriction was introduced, to be more robust against repetitive structures, like for example fence poles. However, as concluded in the *Jump* test sequence, rotation invariant matching is required for certain scenes. A favorable solution is, to keep the false match ratio low, while supporting translations and a few degrees of rotation.

At very high slowdowns, object movements appear jaggy, because the intensities slowly jump from pixel to pixel. This problem may be reduced by rendering the interpolated image with subpixel precision. Furthermore, this method also adds a moderate amount of smoothness to the final interpolated image, for smooth object border transitions.

The Huber-L1 optical flow provides a good basis for our frame interpolation algorithm. However, further research in this field is necessary, to overcome the limitations regarding large displacements, thin moving objects, occlusions, disocclusions, illumination changes, shadows and layered motion. Since the frame interpolation algorithm directly depends on motion vectors, improvements in the optical flow estimation will positively affect the interpolation results.

Bibliography

[1] G. Aubert and P. Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer, 2006.

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[3] S. T. Barnard and W. B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):333–340, 1980.

[4] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision*, pages 29–43. Springer Berlin Heidelberg, 2010.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[6] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[8] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003.

[9] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2009.

[10] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[11] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004.

[12] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1): 120–145, 2011.

[13] T. Chan, G. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. In *International Conference on Analysis and Optimization of Systems Images*, pages 241–252. Springer Berlin Heidelberg, 1996.

[14] T. F. Chan, G. H. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *Society for Industrial and Applied Mathematics Journal on Scientific Computing*, 20(6):1964–1977, 1999.

[15] B.-T. Choi, S.-H. Lee, and S.-J. Ko. New frame rate up-conversion using bi-directional motion estimation. *IEEE Transactions on Consumer Electronics*, 46 (3):603–609, 2000.

[16] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 721–728. IEEE, 2003.

[17] R. Dudek, C. Cuenca, and F. Quintana. An application of optical flow: slow motion effect on streaming image sequences. In *Proceedings of the International Conference on Computer Aided Systems Theory*, pages 654–659. Springer, 2007.

[18] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1033–1038. IEEE, 1999.

[19] D. Fleet and Y. Weiss. Optical flow estimation. In *Mathematical Models in Computer Vision*, pages 237–257. Springer US, 2006.

[20] M. Granados, K. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background inpainting for videos with dynamic objects and a free-moving camera. In *European Conference on Computer Vision*, pages 682–695. Springer Berlin Heidelberg, 2012.

[21] M. Hadamard. Les problèmes aux limites dans la théorie des équations aux dérivées partielles. *Journal of Theoretical and Applied Physics*, 6(1):202–241, 1907.

[22] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17 (1–3):185 – 203, 1981.

[23] P. J. Huber. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5):799–821, 1973.

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

[25] H. Lau and D. Lyon. Motion compensated processing for enhanced slow-motion and standards conversion. In *Proceedings of the International Broadcasting Convention*, pages 62–66. IET, 1992.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[27] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 674–679, 1981.

[28] P. O'Donovan. Optical flow: Techniques and applications. *The University of Saskatchewan, TR*, T.R. 502425, April 2005.

[29] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *Proceedings of the IEEE International Conference on Image Processing*, pages 69–72. IEEE, 2005.

[30] K. A. Patwardhan, G. Sapiro, and M. Bertalmío. Video inpainting under constrained camera motion. *IEEE Transactions on Image Processing*, 16(2):545–553, 2007.

[31] L. Rakêt, L. Roholm, A. Bruhn, and J. Weickert. Motion compensated frame interpolation with a symmetric optical flow constraint. In *Advances in Visual Computing*, pages 447–457. Springer Berlin Heidelberg, 2012.

[32] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[33] T. K. Shih, N. C. Tang, and J.-N. Hwang. Exemplar-based video inpainting without ghost shadow artifacts by maintaining temporal continuity. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(3):347–360, 2009.

[34] Z. Wang and Q. Li. Information content weighting for perceptual image quality assessment. *IEEE Transactions on Image Processing*, 20(5):1185–1198, 2011.

[35] Z. Wang, A. C. Bovik, and L. Lu. Why is image quality assessment so difficult? In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3313–3316. IEEE, 2002.

[36] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, pages 1398–1402. IEEE, 2003.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[38] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for TV-L1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45. Springer Berlin Heidelberg, 2009.

[39] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg. A survey on variational optic flow methods for small displacements. In *Mathematical Models for Registration and Applications to Medical Imaging*, pages 103–136. Springer Berlin Heidelberg, 2006.

[40] M. Werlberger. *Convex Approaches for High Performance Video Processing*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria, June 2012.

[41] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference*, page 3, 2009.

[42] M. Werlberger, T. Pock, M. Unger, and H. Bischof. Optical flow guided TV-L1 video interpolation and restoration. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 273–286. Springer Berlin Heidelberg, 2011.

[43] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Proceedings of the German Association for Pattern Recognition Symposium*, pages 214–223. Springer Berlin Heidelberg, 2007.

[44] L. Zhang, D. Zhang, and X. Mou. FSIM: a feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386, 2011.

[45] L. Zhang, L. Zhang, X. Mou, and D. Zhang. A comprehensive evaluation of full reference image quality assessment algorithms. In *IEEE International Conference on Image Processing*, pages 1477–1480. IEEE, 2012.