universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Analysis and Improvements on the Horn and Schunck Optical Flow Algorithm“

verfasst von / submitted by

Florian Schwarz, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Otmar Scherzer
Mitbetreut von / Co-Supervisor:	Dipl.-Ing. Dr. Clemens Kirisits, BA

Contents

1. Introduction

Given a video consisting of many individual frames, which appears fluent to the human eye, moving objects which are depicted may be interpreted to *flow* across the screen. This flow can be mathematically calculated and described as a vector field, which is typically called optical flow field, velocity field or brightness displacement field. Since a video, or more generally any kind of sequence of images is inherently a transformation of the three-dimensional world to a two-dimensional domain, any motion we observe in such an image sequence may only ever be called apparent. This means that certain effects, e.g. change of illumination in the scene of interest, may lead an observer to misinterpret actual motion occurring in the scene, which is an example for why the problem of optical flow has shown to be a particularly hard one to solve.

Therefore, calculating apparent motion of objects in a sequence of images is still an important open question in computer vision. Applications include traffic analysis, autonomous driving, segmentation of regions in an image and surveillance systems. Generally, the determination and calculation of optical flow will often times be a basic algorithm in higher level applications and algorithms.

A key distinction is that of sparse vs. dense optical flow: while the former describes optical flow only for the most distinct features of the images, making it computationally significantly less demanding and prone to error caused by noise, the latter calculates the flow for every pixel in the image, which delivers more exact results in theory. The two most basic and well-known algorithms in each regard are the Lucas-Kanade-Algorithm [25] (sparse) and the Horn-Schunck-Algorithm [19] (dense).

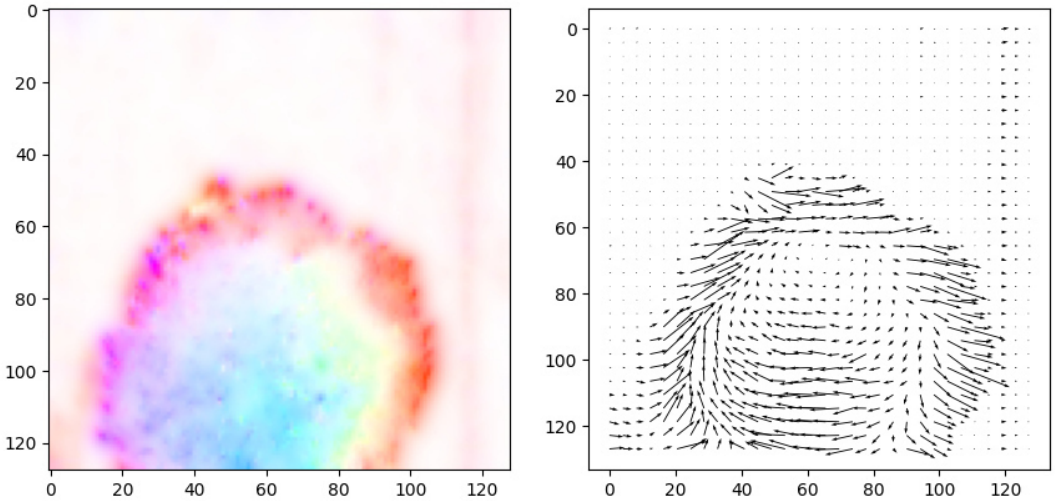

Figure 1: Optical flow of the image sequence in figure 2, shown in the two most common display methods: color wheel-[1] and arrow-representation

[1] see Appendix (Fig. 34)

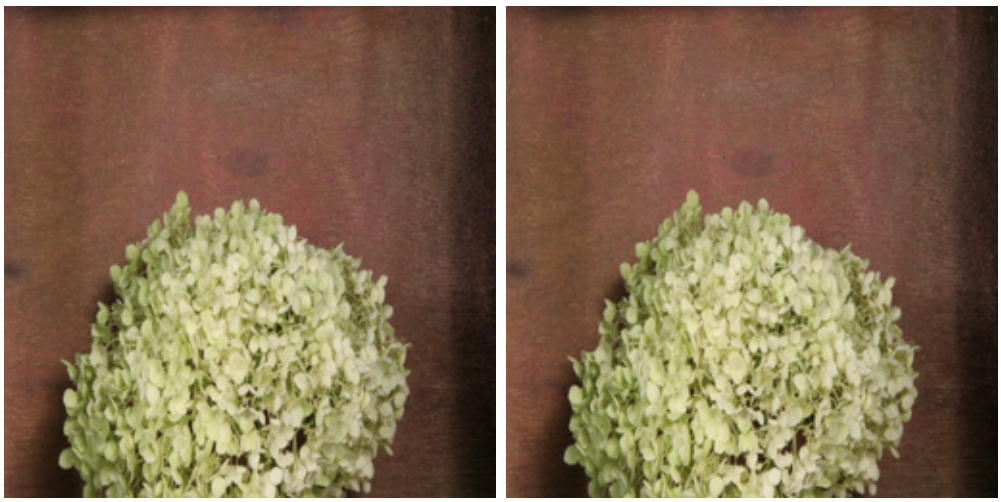

Figure 2: An image sequence[2] taken from [3] as it is usually fed into optical flow algorithms

1.1. Forms of Representation

Optical flow can be visualized in different manners, the two ones relevant for this thesis are depicted in Figure 1 and described below.

- **Color Wheel Representation:** In order to capture both the absolute value as well as the direction of the flow vectors, while maintaining an uncluttered look, this representation is used. It requires the definition of a color wheel, where varying saturation corresponds to displacement distance and color corresponds to displacement direction.

- **Arrow Representation:** This representation offers a very intuitive understanding of the flow present in the image sequence. Drawbacks includes cluttering especially for complex flow fields. In this case it is advised not to represent the displacement vector of every pixel, but only represent vectors at certain intervals. Secondly, visualisation of image sequences with particularly little motion need to be artificially enhanced after the fact, in order to be able to make out flow vectors to begin with. This naturally manipulates actual length of displacement and should be used with caution.

Accurate visualisation is critical, as it is vital for humans mainly in the context of verification and debugging during algorithmic development but also provides the most intuitive understanding of the optical flow at hand. Of course, in many applications when computationally evaluating an optical flow field by e.g. comparing it with the assumed correct flow of the sequence (see Chapter 1.2), visualisation is simply omitted in the process.

[2] resolution and aspect ratio changed

1.2. Evaluation

In order to determine whether the optical flow computed by some technique or algorithm reflects the actual underlying flow of an image sequence accurately enough, it typically does not suffice to judge the result by eye. Instead, the optical flow field will be compared numerically either with flow fields computed by different algorithms or, if available, the so-called *ground-truth flow*. The ground truth is understood to model the actual underlying flow as well as possible using various techniques. As validation of algorithms is important especially during developement of new techniques and for sequences with complex optical flow[3], computation of accurate ground-truth has proven to be an important task in the field.

Techniques for computing ground-truth include:

- **Structured Light:** In this approach, every pixel is *labelled* by a unique code via usage of structured light. By matching codes in different images, displacement vectors of each pixels can be obtained [34].
- **Hidden markers:** Fluorescent paint and additional imaging in UV light or minute visible markings on the scene's contents make it possible to compute the exact displacement of objects [33].
- **Synthetic Generation:** As an alternative for natural sequences, at this point, scenes may synthetically be generated quick and highly realistic using 3D-modelling software. In [26], the authors developed a plugin of such a software, namely *Maya*, which makes precise optical flow calculation possible. Some of these scenes are used in the theoretical part of this thesis in Chapter 4.

As described by Baker et al. in [3], the two most common ways of numerically comparing two flow fields with one another are the *Angular Error* (AE)

$$AE = \arccos\left[\frac{\vec{h}_{F1} \cdot \vec{h}_{F2}}{\|\vec{h}_{F1}\| \cdot \|\vec{h}_{F2}\|}\right], \tag{1.2.1}$$

and the *Endpoint error* (EE),

$$EE = \|\vec{h}_{F1} - \vec{h}_{F2}\|, \tag{1.2.2}$$

where $\vec{h}_{F1}$ and $\vec{h}_{F2}$ describe the flow vectors of the two flow fields to be compared, at specific points. These are the evaluation methods which will also be used in this thesis, with the extension of averaging over all pixels of the flow field ($\bar{AE}, \bar{EE}$), as well as computing standard deviations (σ_{AE}, σ_{EE}), in order to get a grasp on outliers.

[3] e.g. non-rigid and independent motion

1.3. Optical Flow and Brightness Constancy

In order to compute optical flow from a sequence of images, the most popular and basic approach is to consider the constancy of brightness I of a point (x, y) in an image sequence. This means that an object should not change its brightness *too much* when moving across the screen to be properly tracked via this method, a pre-condition which is not met in a scene with drastic changes in illumination.
Mathematically, brightness constancy is expressed as

$$I(x, y, t) = I(x + \partial x, y + \partial y, t + \partial t). \tag{1.3.1}$$

Here, dt describes the time interval between two images, which may be arbitrarily big - however in practise is usually very small, e.g. 1 frame. dx and dy refer to the displacement of the brightness at point (x, y) to a new position $(x + \partial x, y + \partial y)$ in the 2D image space and therefore describe the optical flow field.

Approximation via Taylor Series (to the 1^{st} order) is now applied on the right side

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}\partial x + \frac{\partial I}{\partial y}\partial y + \frac{\partial I}{\partial t}\partial t, \tag{1.3.2}$$

which leads to

$$0 = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t}\frac{\partial t}{\partial t}, \tag{1.3.3}$$

after cancelling and dividing by ∂t. Dependencies are omitted to avoid cluttering.

On an important note, this truncation of the Taylor Series approximation after the 1^{st} order implicates, that schemes born from this approach only work accurately if changes in $\partial x, \partial y$ and ∂t are *small enough*. Finally, introducing velocities in the limit $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$,

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v,$$

or equivalently and most commonly in vector notation,

$$-I_t = \vec{\nabla} I \cdot \vec{h}, \tag{1.3.4}$$

is derived. Here, $\vec{h}(x, y) = \begin{pmatrix} u(x, y) \\ v(x, y) \end{pmatrix}$, the optical flow at point (x, y). Further, unless indicated otherwise throughout this work $a_b = \frac{\partial a}{\partial b}$ holds.

The complication of (1.3.4) arises from the fact that two unknowns u, v appear in only one equation. This lack of information is often referred to as the *aperture problem* in the field of motion detection: When observing the motion of an object through an aperture, one may only comprehend the motion in the direction of the brightness gradient of that object. This leads to perceived motion potentially being created by a multitude of different actual motions (Fig. 3).

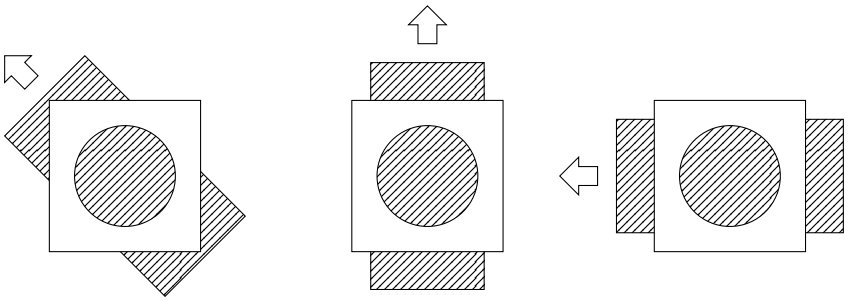

Figure 3: Various actual motions appear identical through an aperture [40]

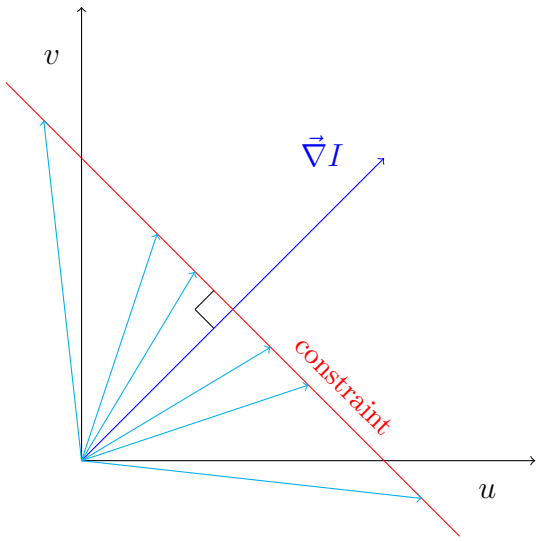

Figure 4: An infinite number of vectors $\vec{h}$ (cyan) solve (1.3.4)

(1.3.4) is also depicted geometrically in Figure 4. Naturally, due to the definition of the scalar product $\vec{a} \cdot \vec{b} = \|\vec{a}\| \cdot \vec{b_a}$, there is not one solution that can be attributed to the problem, but rather infinitely many. The constraint line merely constrains the length of the projection of $\vec{h}$ onto $\vec{\nabla} I$ in the following way [19],

$$\vec{\nabla} I \cdot \vec{h} = \|\vec{\nabla} I\| \cdot \vec{h}_{\vec{\nabla} I} = -I_t$$
$$\|\vec{h}_{\vec{\nabla} I}\| = \frac{I_t}{\|\vec{\nabla} I\|}.$$

In order to solve the problem of optical flow with the main assumption of brightness constancy it is therefore necessary to employ additional constraints, most popularly introduced in the variational approach by Horn and Schunck which is discussed in Chapter 3.

Many algorithms do not consider brightness constancy in the first place, utilising entirely different approaches in order to tackle optical flow, and are discussed in the Chapter 2. They include gradient-, region-, feature- and machine learning-based approaches.

1.4. Complications in Optical Flow Computation

- **Large Displacements**: One of the main challenges in optical flow computation is that of properly dealing with so-called Large Displacements. A Large Displacement is usually defined as an object's or scene's motion larger than merely a few pixels, which leads to schemes derived from linearisation like (1.3.2) to struggle. Employing pyramidal approaches, where large movement is resolved at lower resolutions aids in tackling this issue. The effects of Large Displacements on an algorithm not fit for dealing with them are more closely demonstrated in Chapter 3.7.

  Aside from pyramidal or multi-scale approaches [28], usage of non-linear approaches [1, 32] or diverting from gradient-based approaches all-together is typically employed. Specifically feature- or region-based approaches generally offer satisfying results as well [7].

  In Fact, appropriately dealing with Large Displacements will be the focus of the improvements made over the basic Horn and Schunck algorithm (Chapter 4).

- **Occlusions**: Parts of a scene or objects may become more or less occluded throughout a sequence of images. This is problematic, since an object which becomes visible only in the latter frame can not be assigned a flow field, since its origin is unknown to the algorithm. Occluded areas are thus often times understood to be un-solvable, i.e. any flow could be assigned to them. However, if e.g. parts of uniformly moving background become occluded, *Occlusion filling* via e.g. *inpainting methods* [5], which are usually for instance used to restore damaged paintings, may be viable.

  Although occlusions are encountered during measurements and in part discussed in Chapter 4.5, solving them is not a particular focus of this thesis.

- **Illumination Changes**: Naturally, algorithms based on brightness constancy are not suited to deal with significant Illumination Changes throughout an image sequence. Possible improvement of the situation may be offered by schemes employing brightness gradient constancy alongside brightness constancy, as seen e.g. in the variational approach of Papenberg et al.[32]. On top of that, since illumination change occurs in the structural rather than the textural part of an image, basing optical flow computation on the latter is the basis of the work of Wedel et al. in [39].

  Of the image sequences analysed in this thesis, none exhibit particular illumination changes, i.e. pixel intensities of the same objects in two images of a sequence remain largely unchanged.

- **Motion Discontinuities**: Assuming smooth flow and low amounts of motion discontinuities was popularised by Horn and Schunck in their work originally and significantly simplifies calculation [19]. The assumption of large areas in an image sequence moving in unison is not unwarranted, however flow computation may as a result appear unsatisfying around structural edges and corners.

  Generally this issue may be dealt with for instance by usage of specific regularisation potentials, which support smooth motion in areas of uniform movement, while still detecting significant discontinuities [15]. More precisely, combining so-called Total Variation,

$$R_{TV}(\vec{h}) = \|\nabla\vec{h}\|_1,$$

  which deals well with sharp edges, and the smoothness assumption posed by Horn and Schunck,

$$R_{HS}(\vec{h}) = \|\nabla\vec{h}\|_2^2,$$

  is an approach most favourable for real-life sequences, as summed up in [9]. The nature and use of such a regularisation term is more closely discussed in Chapter 3.

1.5. Relevant Mathematical Principles

- **Least Squares Solution**:

  An over-determined system of equations $A\vec{x} = \vec{b}$ is unsolvable, since $\vec{b} \notin col(A)$, i.e. $\vec{b}$ is not in the column space of A. In order to approximate a solution $\vec{x}^*$, the least squares approach is employed,

$$\vec{x}^* = \underset{\vec{x}}{argmin}\|\vec{b} - A\vec{x}\|^2, \tag{1.5.1}$$

  which aims to minimize the error of that solution. Geometrically, the goal is to find the orthogonal projection $\vec{b}_{\|}$ onto $col(A)$, i.e. the closest point to $\vec{b}$ still in $col(A)$. In order to eliminate the orthogonal component $\vec{b_{\perp}}$ and be left only with $\vec{b}_{\|}$, $A^\top$ is applied onto $\vec{b}$, since $A^\top\vec{b}_{\perp} = \vec{0}$. This then leads to the normal equations,

$$A^\top A\,\vec{x} = A^\top\vec{b}, \tag{1.5.2}$$

  which may be solved if $A^\top A$ is regular, i.e. has full rank, is invertible.

- **Tikhonov Regularisation**:

  An under-determined system of equations $A\vec{x} = \vec{b}$ is ill-conditioned, since it may have infinitely many solutions. In order to fix this problem one may use a prior in form of an additional regularisation term in (1.5.1),

  $$\vec{x}^* = \underset{\vec{x}}{argmin}(\|A\vec{x} - \vec{b}\|^2 + \|\Gamma\vec{x}\|^2). \tag{1.5.3}$$

  This term should contain additional a-priori information of the system in order to select for a favourable solution, which is then defnined as,

  $$\vec{x}^* = (A^\top A + \Gamma^\top \Gamma)^{-1} A^\top b,$$

  i.e. the (almost) singular matrix $A^\top A$ is manipulated such that its condition number decreases and the system becomes well-conditioned.

- **Euler-Lagrange Equations**:

  Minimizing a functional of form[4],

  $$J[y] = \int_\Omega F(x, y, y')dx,$$

  is known to be the *simplest problem* of variational calculus [12]. It is conventionally addressed by solving the related Euler-Lagrange-Equations, which are a necessary but not sufficient condition a function y must satisfy in order to minimize functional (1.5). With n functions and m dimensions the Euler-Lagrange-Equations in first order are derived to be,

  $$\frac{\partial F}{\partial y_i} - \sum_{j=1}^{m} \frac{\partial}{\partial x_j} \frac{\partial F}{\partial y_{i,j}} = 0, \tag{1.5.4}$$

  $$i = 1, ..., n \qquad y_{i,j} = \frac{\partial y_i}{\partial x_j} \qquad \frac{\partial y}{\partial n} = 0 \quad on\ \partial\Omega$$

  with the natural homogeneous Von Neumann boundary conditions.

- **Frobenius Norm**:

  The Frobenius Norm as it is used in the scheme of Horn and Schunck as well as for evaluation purposes in the practical part of this thesis, is defined as,

  $$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \|a_{i,j}\|^2}, \tag{1.5.5}$$

  i.e. the Euclidian Norm extended to a matrix.

[4] first order, one function, one dimension

- **Lemma of Lax-Milgram** [35, 42]:

  (A vector $\vec{x}$ is denoted as $\boldsymbol{x}$.)

  Given a bilinear form $a(\cdot,\cdot) : V \times V \to \mathbb{R}$, with a Hilbert space V. Assume the following holds,

$$\begin{aligned} \|a(\boldsymbol{h},\boldsymbol{g})\| &\leq c_1\|\boldsymbol{h}\|_V\|\boldsymbol{g}\|_V && \text{(continuity)} \\ \|a(\boldsymbol{h},\boldsymbol{h})\| &\geq c_2\|\boldsymbol{h}\|_V^2 && \text{(coercivity, V-ellipticity)} \end{aligned}$$

  with constants $c_1 > 0$ and $c_2 > 0$. Then, assuming a continuous, linear mapping $F : V \to \mathbb{R}$, i.e. $F \in V'$, where V' is the dual space of V, there exists exactly one solution $\boldsymbol{h}^*$ for every $\boldsymbol{g}$,

$$a(\boldsymbol{h}^*,\boldsymbol{g}) = F(\boldsymbol{g}) = \langle F,\boldsymbol{g}\rangle. \tag{1.5.6}$$

  Identifying $a(\cdot,\cdot)$ with linear bounded operator $A : V \to V'$ and specific to coercivity, A is invertible with,

$$\|A^{-1}\| \leq \frac{1}{c_2}. \tag{1.5.7}$$

2. Approaches for solving Optical Flow: an Overview

As with many problems in computational science, there is not one single approach which obtains the best results, but it is rather about weighing advantages and drawbacks against one another. Over the last decades, a multitude of different schemes with the aim of producing the most accurate flow field from a sequence of images, have been developed. The following shall give a short overview of the most relevant concepts.

2.1. Horn and Schunck

One of the earliest works on the topic of optical flow was done by Horn and Schunck in 1981 [19]. To this day it is often viewed as the prototypical optical flow algorithm and, even though it is not used in high performance environments today anymore, certain key aspects are still employed in various modern algorithms. As it plays a vital role in the practical section of this thesis, an extensive overview is provided in Chapter 3.

2.2. Lucas and Kanade

Also in 1981, Lucas and Kanade published an approach for solving the problem of *Image Registration* (IR) [25], which is related to optical flow, using a translational warp model. It is especially relevant in fields like medical imaging and deals with the alignment of two images of the same scene[5], which may be arbitrarily shifted, scaled and skewed. Optical flow can be seen as a sub-problem of IR and for both, similar solving techniques are employed. Mathematically, IR may be solved by finding the disparity vector $\vec{h}^*$ at some position $\vec{x}$, e.g. in an L_1 norm,

$$\vec{h}^* = \underset{\vec{h}\in\Omega}{argmin} \|I_1(\vec{x}+\vec{h}) - I_2(\vec{x})\|, \tag{2.2.1}$$

where I_1, I_2 denote the two images respectively.

As a special case of this approach, Lucas-Kanade-Optical-Flow was developed, which much like the scheme of Horn and Schunck, is initially based on brightness constancy (1.3.1) and referred to as a gradient-based scheme. In order to solve the under-determined system of equations (1.3.4), the constraint of locally constant flow is employed. This means that given a small enough window in the image, it can be assumed that all contained pixels share the same[6] optical flow vectors.

Using e.g. a 3×3 window, (1.3.4) becomes,

[5] at least partially

[6] in practise: very similar

$$\begin{pmatrix} \vec{\nabla} I_{0,0}^{\top} \\ \vec{\nabla} I_{0,1}^{\top} \\ .. \\ \vec{\nabla} I_{2,2}^{\top} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_{t;0,0} \\ I_{t;0,1} \\ .. \\ I_{t;2,2} \end{pmatrix},$$

which is simply a system of form,

$$A\,\vec{x} = \vec{b},$$

most likely overdetermined and thus solvable by a least squares approach (1.5.2).

As was already mentioned in the introduction, the approach of Lucas and Kanade produces sparse optical flow, because it assumes patches of unified flow. The method struggles with two situations specifically [31]:

- Smooth areas, i.e. those with small and uniform brightness changes in all directions, as there is not much texture for the algorithm to "cling" onto. If gradients in a chosen patch are equal for all pixels, the resulting matrix is not regular and the patch becomes unsolvable.

- Edges, i.e. areas where the derivative in one direction is much greater in magnitude than the derivative in the other direction. This is due to the eigenvalues of $A^{\top}A$ corresponding to the largest gradient of the window in each direction x and y. The condition number of a self-adjoint matrix A, which measures the stability of the solution x given some perturbation to the matrix or the right hand side, is given by,

  $$cond(A) = \frac{\lambda_{max}}{\lambda_{min}},$$

  where λ_{max} and λ_{min} describe the biggest and smallest eigenvalue. For big disparity between greatest and smallest eigenvalues, the condition number will be large and thus the system will be ill-conditioned and hard to solve.

2.3. Region-based

The key idea of region-based optical flow is to have an algorithm recognize similar-looking regions in a sequence of images and thus compute the displacement this region has undergone. Usually, as e.g. described in [16] such an approach consists of three steps: Region-Extraction, Region-Matching, Flow-Calculation.

- **Region-Extraction:** The process of region-extraction or region- respectively object-detection aims at identifying regions or areas in an image which meaningfully describe a coherent structure. It is itself an extensive and important field in computer vision and may be solved using a multitude of different algorithms,

as e.g. described in [27, 17]. Most of these methods usually employ some kind of region hierarchy and different applications, e.g. computation of optical flow, operate on high or low hierarchy respectively. Depending on the algorithm at hand, a certain hierarchy level is chosen in order to have a region represent a uniformly moving part of the image as well as possible.

- **Region-Matching**: Two images, for which region extraction has previously been performed can be processed by a region-matching algorithm. Its goal is to find pairs of regions, which are most likely to correspond to each other by usage of affinity measures. Measures such as average intensity, area distance and distance between region centroids may be employed [16]. Techniques like nearest-neighbour-matching may be used in order to evaluate the differences in these parameters of different regions [7].

- **Flow-Calculation**: By merely comparing the region centroids of two matched regions, a first approximation of the displacement field in that region may be obtained. Further smoothing via median filtering may be applied [16], or the results of region-matching may be introduced into further computation via a different technique, like a variational approach [7], among others.

2.4. Feature-based

Similar to region-based flow, the steps of extraction and matching also need to be undertaken in the approach of feature-based optical flow. As the name suggests, striking image features are extracted by usage of algorithms and are subsequently matched in separate frames of an image sequence. Depending on what features to extract, there exists a wide variety of edge detectors (e.g. [10]), corner detectors (e.g [18]) or blob detectors (e.g. [23]). The latter detection type is especially useful if some large area in the image with low brightness gradient does not have distinct features to track by other methods, and is close in essence to region-extraction. Still, these methods may struggle when dealing with images which feature large untextured patches and produce rather sparse flow.

Optical flow is calculated on a feature-based approach for instance in [41], which also tackles the problems of feature-wise over- and under-representation of certain objects in the scene. After matching the features, the resulting motion fields are then influencing a final minimization procedure to obtain the displacement for individual pixels.

2.5. Machine-learning-based

As in most areas of computational science, many state-of-the-art algorithms are based on machine learning whose importance is ever increasing in the field of science. Convolutional Neural Networks (CNNs) have been especially successful (popularly e.g. *flowNet*

[13] and *RAFT*[36]). Here, end-to-end processes are employed, meaning given the most basic input, i.e. two images, optical flow is computed for every pixel. Naturally, machine-learning, specifically CNNs, may be trained to perform a sub-task of other approaches, e.g. extracting features from an image.

The performance of machine-learning based approach *RAFT* is shown and compared to that of the improved Horn and Schunck approach implemented in this thesis, in Chapter 5.

3. Horn-Schunck-Optical-Flow in detail

In 1981, Berthold Horn and Brian Schunck published their first work focused on optical flow, with the goal of solving the issue of inherent lack of information in the aperture problem by introducing an additional constraint. Even though effort was put into determining velocities of objects on a screen up to that point (e.g. [14, 22]), the work of Horn and Schunck kick-started the field of optical flow estimation and led to its increased attention throughout the scientific community.

3.1. Setting up the Horn-Schunck Energy functional

Tikhonov Regularisation (1.5.3) is introduced into the problem of Optical Flow by first constructing an energy functional, which is further solved using variational calculus, more specifically the Euler-Lagrange Equations (1.5.4).

Setting up a functional such as this based on the aperture problem (1.3.4) and expanding it with a regularization term, leads to the famous and rather generic energy-functional proposed by Horn and Schunck,

$$J[\vec{h}] = \int_\Omega [(\vec{\nabla} I \cdot \vec{h} + I_t)^2 + R(\vec{h})] d\vec{x}. \tag{3.1.1}$$

Commonly, $(\vec{\nabla} I \cdot \vec{h} + I_t)^2$ is referred to as the *data term*, which is depicted in its most basic form, but may be altered in various ways in order to improve performance of consequential algorithms.
In the case of Horn and Schunck, $R(\vec{h})$ can be interpreted as some kind of Frobenius Norm (1.5.5) of the outer product of $\vec{\nabla}$ and the optical flow, where the former serves as the analog to matrix Γ in (1.5.3),

$$R(\vec{h}) = \int_\Omega \alpha^2 \|\vec{\nabla}\vec{h}\|_F^2 d\vec{x}. \tag{3.1.2}$$

Here, and in the following, $\vec{h} = \begin{pmatrix} u \\ v \end{pmatrix}$ again describes the optical flow field at a certain point. α is a commmonly introduced scaling factor which is usually chosen and adapted empirically but famously difficult to optimize for. In the original work, the regularization term is reasoned as being a *smoothness constraint*, since the added term is equivalent to a sum of squared derivatives in all directions,

$$\|\vec{\nabla}\vec{h}\|_F^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

Minimizing the squared derivatives in all directions assumes smooth flow generally, an assumption which holds in many image sequences, where large parts of the image move in uniform and abrupt changes in flow are rare.

3.2. Solving the Euler-Lagrange-Equations

Starting from (3.1.1), the general case (1.5.4) reduces to two equations for the components of $\vec{h}$, u and v respectively,

$$\begin{aligned} \frac{\partial F}{\partial u} - \frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y} = 0 \\ \frac{\partial F}{\partial v} - \frac{\partial}{\partial x}\frac{\partial F}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial v_y} = 0, \end{aligned} \tag{3.2.1}$$

with $F = \left[\vec{\nabla} I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t\right]^2 + \alpha^2 \left\| \vec{\nabla} \begin{pmatrix} u \\ v \end{pmatrix} \right\|_F^2$.

Afterwards, (3.2.1) simplifies to

$$\begin{aligned} 2\left[\vec{\nabla} I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t\right] I_x - \frac{\partial}{\partial x}\left(2\alpha^2 \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(2\alpha^2 \frac{\partial u}{\partial y}\right) = 0 \\ 2\left[\vec{\nabla} I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t\right] I_y - \frac{\partial}{\partial x}\left(2\alpha^2 \frac{\partial v}{\partial x}\right) - \frac{\partial}{\partial y}\left(2\alpha^2 \frac{\partial v}{\partial y}\right) = 0 \end{aligned}$$

$$\begin{aligned} 2\left[\vec{\nabla} I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t\right] I_x - 2\alpha^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \\ 2\left[\vec{\nabla} I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + I_t\right] I_y - 2\alpha^2 \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0 \end{aligned}$$

and finally identifying the Laplacian,

$$\begin{aligned} (I_x u + I_y v + I_t) I_x - \alpha^2 \triangle u = 0 \\ (I_x u + I_y v + I_t) I_y - \alpha^2 \triangle v = 0 \end{aligned} \tag{3.2.2}$$

yielding a system of two partial differential equations of second order. In order to numerically solve (3.2.2), Horn and Schunck make use of an iterative approach, by first approximating

$$\triangle u \approx \kappa(\bar{u} - u),$$

where $\bar{u}$ uses the stencil

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\ \frac{1}{6} & 0 & \frac{1}{6} \\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \end{pmatrix}, \tag{3.2.3}$$

similarly for $\triangle v$. $\kappa = 3$ for this approximation scheme, although it is further omitted as it serves merely as a scaling factor for the weighting term α. Numerical approximations of derivatives I_x, I_y, I_t are further shown in Chapter 3.4 alongside implementation of the scheme. Then, for every pixel, a system of equations $A\vec{x} = \vec{b}$ is recovered,

$$\begin{pmatrix} I_x^2 + \alpha^2 & I_x I_y \\ I_x I_y & I_y^2 + \alpha^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t I_x + \alpha^2 \bar{u} \\ -I_t I_y + \alpha^2 \bar{v} \end{pmatrix},$$

which is solved via usage of iteration scheme (3.2.5) in the original work. It may be derived using *Cramer's Rule*, which states the i-th component of the solution vector,

$$x_i = \frac{det(A_i)}{det(A)}, \tag{3.2.4}$$

where A_i describes the original matrix A with the i-th column being substituted by right hand side b. For the case of u, with

$$A_u = \begin{pmatrix} -I_t I_x + \alpha^2 \bar{u} & I_x I_y \\ -I_t I_y + \alpha^2 \bar{v} & I_y^2 + \alpha^2 \end{pmatrix},$$

the determinants compute to be,

$$det(A) = \alpha^2(I_x^2 + I_y^2 + \alpha^2), \qquad det(A_u) = \alpha^2(-I_x I_t + I_y^2 \bar{u} + \alpha^2 \bar{u} - \bar{v} I_x I_y),$$

using (3.2.4) and rearranging,

$$u = \frac{-I_x I_t + (I_y^2 + \alpha^2)\bar{u} - \bar{v} I_x I_y}{I_x^2 + I_y^2 + \alpha^2}$$

$$u(I_x^2 + I_y^2 + \alpha^2) = -I_x I_t + (I_y^2 + \alpha^2)\bar{u} - \bar{v} I_x I_y + (I_x^2 \bar{u} - I_x^2 \bar{u})$$
$$u(I_x^2 + I_y^2 + \alpha^2) = -I_x(\bar{u} I_x + \bar{v} I_y + I_t) + \bar{u}(I_x^2 + I_y^2 + \alpha^2)$$

finally leading to the update scheme,

$$\begin{aligned} u^{k+1} &= \bar{u}^k - \frac{I_x(\bar{u}^k I_x + \bar{v}^k I_y + I_t)}{I_x^2 + I_y^2 + \alpha^2} \\ v^{k+1} &= \bar{v}^k - \frac{I_y(\bar{u}^k I_x + \bar{v}^k I_y + I_t)}{I_x^2 + I_y^2 + \alpha^2}, \end{aligned} \tag{3.2.5}$$

where v^{k+1} is derived analogously.

3.3. Well-posedness and Convergence

In his work in 1991, Christoph Schnörr [35] answered the question of well-posedness, i.e. existence, uniqueness and continuous dependence on input data, for the problem posed by Horn and Schunck[7] (3.1.1, 3.1.2).
(A vector $\vec{x}$ is denoted as $\boldsymbol{x}$.)
Schnörr sets up a basic functional of form,

$$J(\boldsymbol{g}) = \frac{1}{2} a(\boldsymbol{g}, \boldsymbol{g}) - f(\boldsymbol{g}) + c \qquad J : G \to \mathbb{R}, \tag{3.3.1}$$

[7] and also that of Nagel [29]

and subsequently uses the Lax-Milgram-Lemma (Chapter 1.5) based on assuming $a(\boldsymbol{g}, \boldsymbol{g})$ to be a continuous bilinear and $f(\boldsymbol{g})$ a continuous linear form. Then, $J(\boldsymbol{g})$ is uniquely minimized by the solution $\boldsymbol{h}$ of,

$$a(\boldsymbol{h}, \boldsymbol{g}) = f(\boldsymbol{g}). \tag{3.3.2}$$

With (1.5.6), (3.3.2) may be written as,

$$\langle A\boldsymbol{h} - f, \boldsymbol{g} \rangle = 0,$$

implying *existence*,

$$A\boldsymbol{h} = f,$$

and further *uniqueness*,

$$\boldsymbol{h} = A^{-1} f,$$

with inverse A^{-1}. *Continuous data dependence* is then guaranteed by,

$$\|\boldsymbol{h}\| \leq \frac{1}{c_2} \|f\|,$$

using (1.5.7), again with $c_2 > 0$.
In order to check whether the Horn-Schunck functional fits into this framework, $a(\boldsymbol{h}, \boldsymbol{g})$ and $f(\boldsymbol{g})$ need to be set up first.

$$\begin{aligned} a(\boldsymbol{h}, \boldsymbol{g}) = \int_\Omega & I_x^2 h_1 g_1 + I_x I_y (h_1 g_2 + h_2 g_1) + I_y^2 h_2 g_2 \\ & + \alpha^2 (h_{1,x} g_{1,x} + h_{1,y} g_{1,y} + h_{2,x} g_{2,x} + h_{2,y} g_{2,y}) \, d\boldsymbol{x}, \\ f(\boldsymbol{g}) = -2 \int_\Omega & I_t I_x g_1 + I_t I_y g_2 \, d\boldsymbol{x}, \end{aligned} \tag{3.3.3}$$

with $\boldsymbol{h} = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$, $\boldsymbol{g} = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$ and $h_{i,j} = \frac{\partial h_i}{\partial j}$ where the constant factor c in (3.3.1) may be omitted. The key results are then those for continuity of $a(\boldsymbol{h}, \boldsymbol{g})$ and $f(\boldsymbol{g})$,

$$\begin{aligned} \|a(\boldsymbol{h}, \boldsymbol{g})\| \leq C \|\boldsymbol{h}\|_1 \|\boldsymbol{g}\|_1, \qquad & C = 2 \ max\{2\|I_x^2\|_\infty, 2\|I_y^2\|_\infty, \alpha^2\} \\ \|f(\boldsymbol{g})\| \leq 2 \|I_t \vec{\nabla} I\|_0 \|\boldsymbol{g}\|_1, & \end{aligned} \tag{3.3.4}$$

where the norm subscripts other than ∞ denote the order of derivative influencing the calculation and are based on the scalar product in the corresponding *Hilbert Space* V, which is a product of *Sobolev Space* $H^1(\Omega)$,

$$V := \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \in H^1(\Omega) \times H^1(\Omega),$$

and $\|\cdot\|_\infty$ the maximum norm. Further, the bilinear form $a(\boldsymbol{h}, \boldsymbol{g})$ is shown to be elliptic with reference to G under the critical assumption of intensity gradients I_x, I_y being linearly independent, i.e. strictly fulfilling the Cauchy-Schwarz-Inequality,

$$\|\langle I_x, I_y \rangle_0\| < \|I_x\|_0 \cdot \|I_y, I_y\|_0.$$

Another important result is that of convergence of the approximate to the exact solution in the case of the Horn-Schunck algorithm. After previous attempts the problem seems to have finally been solved in 2013 for arbitrary dimension in [21].

Precisely, the authors first set up three hypotheses about the Laplace-Approximation $M(\boldsymbol{h})_i = \sum_{j\in\Lambda} \lambda_{i,j}\boldsymbol{h}_j$ used by Horn and Schunck, based on stencil (3.2.3) in $2D$ in the given lattice of pixels Λ:

- $\lambda_{ji} = \lambda_{ij}, \forall i, j \in \Lambda$
- $\sum_{j\in\Lambda} \lambda_{i,j} = 1, \forall i \in \Lambda$
- Graph G with vertices $V(G) = \Lambda$ and edges $E(G) = \{(i,j) \in \Lambda^2 : \lambda_{i,j} \neq 0\}$ is connected.

With $\boldsymbol{\nabla} I_i$ being the gradient of intensity field I at point i, it is subsequently proven that,

- if the rank of $\boldsymbol{\nabla} I_i$ is 2, (3.2.2) has a unique solution and iteration scheme (3.2.5) converges to it.
- if the rank of $\boldsymbol{\nabla} I_i$ is not 2, i.e. all intensity gradients are contained in the same hyperplane, the problem is ill-posed and the solution of (3.2.2) is not unique.

This result is further extended to an approximation of the Laplacian in n dimensions.

3.4. Implementation

Naturally, the Horn-Schnunck Optical Flow Algorithm has been implemented in various programming languages over the years, e.g. *OpenCV* [6], Open Source Computer Vision Library, among others. Nevertheless, all code in this Thesis is written from scratch, starting from the original Work by Horn and Schunck [19]. This excludes packages for tasks like plotting data, etc. Using *Python 3.10* [37] as a language of choice is justified by personal experience as well as high flexibility and great community/developer support online. Of course, this comes at the price of speed and efficiency in comparison to languages like C or Fortran, which are regarded highly in the field of Computational Science, however these properties should not be in focus here. All code, including comments, may be found at `https://github.com/schwarzflo/masters_thesis`, where key passages are explained in comment where necessary.

Relevant File: *basic_hs.py*
Starting from the update scheme (3.2.5), calculation of the image intensity gradients I_x, I_y, I_t as well as $\bar{u}, \bar{v}$ is necessitated beforehand. The latter is computed via usage of a built in function of the package *scipy* [38], as the application of stencil (3.2.3) is understood to be a convolution of it with the initial or previous velocity field.
Further, for calculation of the image gradients in space and time at a given pixel the following finite difference scheme is proposed by Horn and Schunck.

$$\begin{aligned} I_x(j,k) &= \frac{1}{4} \cdot (I_{i,j+1,k} - I_{i,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} \\ &\quad + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i+1,j+1,k+1} + I_{i+1,j,k+1}) \\ I_y(j,k) &= \frac{1}{4} \cdot (I_{i,j,k+1} - I_{i,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} \\ &\quad + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i+1,j+1,k+1} + I_{i+1,j+1,k}) \\ I_t(j,k) &= \frac{1}{4} \cdot (I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i,j,k+1} \\ &\quad + I_{i+1,j+1,k} - I_{i,j+1,k} + I_{i+1,j+1,k+1} + I_{i,j+1,k+1}) \end{aligned} \tag{3.4.1}$$

Here, the three indices in $I_{i,j,k}$ refer to time, x-axis and y-axis in the image in that order. That is, neighbouring pixels in space and time are taken into account. The resulting gradients lose the time component since for an image sequence of two images, only one set of gradients is computed. The scheme can geometrically be represented as a cube where each of the eight corners refers to a term in the gradient computations in (3.4.1). At the image edges $\partial\Omega$, homogeneous Neumann boundary conditions are employed,

$$\frac{\partial I}{\partial n} = 0 \quad on\ \partial\Omega, \tag{3.4.2}$$

meaning the intensity values are assumed constant from the edge onwards, in all directions. Unless a previous guess for the flow field is available, the flow field is initially

considered to be 0 everywhere,

$$\vec{h_0}(x,y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \forall\ (x,y) \in \Omega. \tag{3.4.3}$$

A quick summary of the algorithm looks as follows:

Data: images, α, iteration count/error threshold
for *width of image* **do**
  for *height of image* **do**
    Gradient Calculation (3.4.1);
  end
end
while *iteration count/error threshold not met* **do**
  scipy.ndimage.convolve[8];
  for *width of image* **do**
    for *height of image* **do**
      Update Scheme (3.2.5);
    end
  end
end

Algorithm 1: Horn-Schunck Optical Flow

The runtime estimation of this algorithm is composed of $24 \cdot m \cdot n$ for the gradient calculation and $it \cdot (2 \cdot 9 \cdot m \cdot n + 24 \cdot m \cdot n)$ for the update scheme. Here, m and n denote the width and height of the images, respectively. it refers to the maximum iteration count, either previously chosen or based on the error threshold used. The factor of 24 refers to the floating point operations (FLOPS) needed for executing (3.4.1) and (3.2.5) respectively. The additional $2 \cdot 9 \cdot m \cdot n$ in brackets refers to the *spn.convolve* routine, which naturally needs to work through every pixel of the image twice for each direction u and v and apply the stencil (3.2.3).

Overall, runtime performance R_{HS} is,

$$\begin{aligned} R_{HS} &= 24 \cdot m \cdot n + it \cdot (2 \cdot 9 \cdot m \cdot n + 24 \cdot m \cdot n) \\ &= 24 \cdot m \cdot n + 42 \cdot it \cdot m \cdot n \\ &= (24\frac{1}{it} + 42) \cdot it \cdot m \cdot n \\ &\in O(it \cdot m \cdot n). \end{aligned}$$

[8] Scipy Package: apply stencil (3.2.3) via convolution to compute $\bar{u}, \bar{v}$

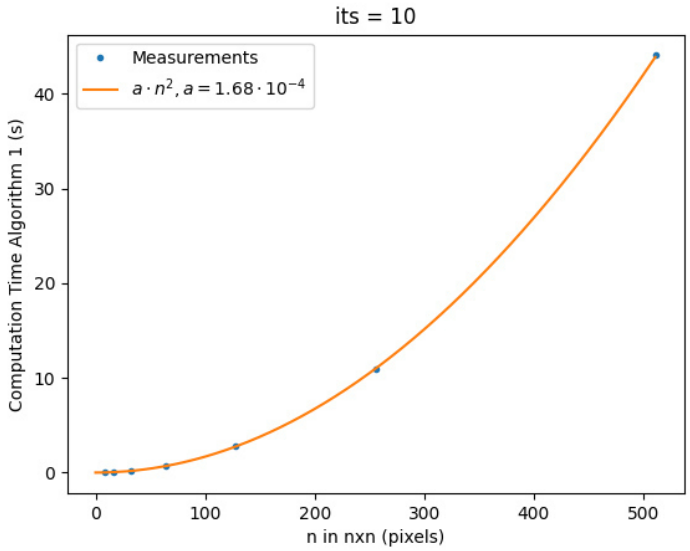

Figure 5: The runtime is expectedly fitted by $a \cdot n^2$, in a squared image $(n \times n)$ and negligibly small iteration count $it = 10$

3.5. Influence of α and the Iteration Count

As mentioned previously, optimizing for the regularisation parameter α is generally considered a serious challenge and is highly dependent on the image sequence at hand. In the following, the *Whale* Sequence (Fig. 42) is examined and the influence of changing α, as well as the iteration count, is highlighted. To verify the validity of a value, comparison with the ground-truth, i.e. the best possible optical flow estimate available of a sequence, is made. The measures described in Chapter 1.2, namely Angular Error (1.2.1) as well as Endpoint Error (1.2.2) are employed. Afterwards, the average and the standard deviation of these errors over the whole image plain are computed.

Importantly, as mentioned in Chapter 1.1, arrow representation needs to be altered for small displacement sequences,in order to improve visualisation, which is also done in the following. Lengths of arrows therefore do *not* accurately reflect displacement distance.

Differences in the computed flow fields are already easily noticeable visually for various values of α (Fig. 6), although even when knowing the original sequence, it may be hard to determine the *best* result. Therefore, scalar fields of the Angular Error and the Endpoint Error are displayed for each case (Fig. 7), to provide a direct visual comparison with ground truth. Specifically, increasing α seems to lead to the computed flow being more prominent around significant edges, while the opposite holds for decreasing the parameter.

This may be reasoned in the following way: choosing α large leads the denominator in (3.2.5) to be similar for areas of big and small intensity change. Therefore, the values in the numerator control the final value $\vec{h}$ in magnitude and enhance it at structural edges. Small α in contrast do not *wash out* insignificant gradient changes and place less emphasis on edges, leaving a more uniform flow field. However, the extreme case of very small α results in a noisy flow field, where small intensity changes are not suppressed and in-appropriately favoured.

In an attempt to truly optimize for α, Fig. 8 (*top*) depicts averages and standard deviations of these errors for different values. As can be observed, very low values of α tend to lead to high standard deviation in the errors of both angle and endpoint. This is to mean there will be areas in an image sequence which may be described particularly poorly, while the general optical flow approximation seems fine. This is also visible in Fig. 7 for $\alpha = 1$, where the highest Endpoint Error is > 25 pixels. The metrics generally hint at an improvement and a subsequent decline in performance for increasing values of α. In any case, the optimal choice of α will be a trade-off in some way, and lies somewhere between $\alpha = 1$ and $\alpha = 10$ for the present sequence of images.
Since the ground truth files themselves usually contain many *holes*, i.e. unknown flow values, mainly due to occlusions, the validity of the computed flow field can not be evaluated in these positions. In Fig. 7, these positions are coloured white - they are further simply omitted in calculations of averages and standard deviations.

In order to empirically prove convergence of the algorithm, the change of the optical flow field from one iteration to the next is monitored. More precisely, the relative change is computed as follows,

$$\Delta u_k = \frac{\|\mathbf{u_{k-1}} - \mathbf{u_k}\|_F}{\|\mathbf{u_{k-1}}\|_F}, \qquad \Delta v_k = \frac{\|\mathbf{v_{k-1}} - \mathbf{v_k}\|_F}{\|\mathbf{v_{k-1}}\|_F}. \tag{3.5.1}$$

Here, $\mathbf{u_k}, \mathbf{v_k}$ refer to scalar fields for the flow in u and v direction at every pixel in the image at the k-th iteration. Interpreting these as matrices, the difference and subsequence use of the Frobenius Norm (1.5.5) give an insight on the (relative) change at each iteration.
Figure 8 (*middle*) highlights the convergence of the algorithm for different α towards no change between iterations at all. Specifically, lower values of α tend to converge faster. This makes sense, since large values will reduce the effect of the quotient and thus the change per iteration in (3.2.5).
Lastly, Figure 8 (*bottom*) depicts how the result of the computation on average improves for more iterations compared to ground truth flow, with relatively steady behaviour of the standard deviation. This is expected, since problem areas will not be drastically improving via applying more iterations. Therefore, variation in the result should stay roughly the same.

[9] parameter *div* controls in which period flow vectors are shown in the arrow representation

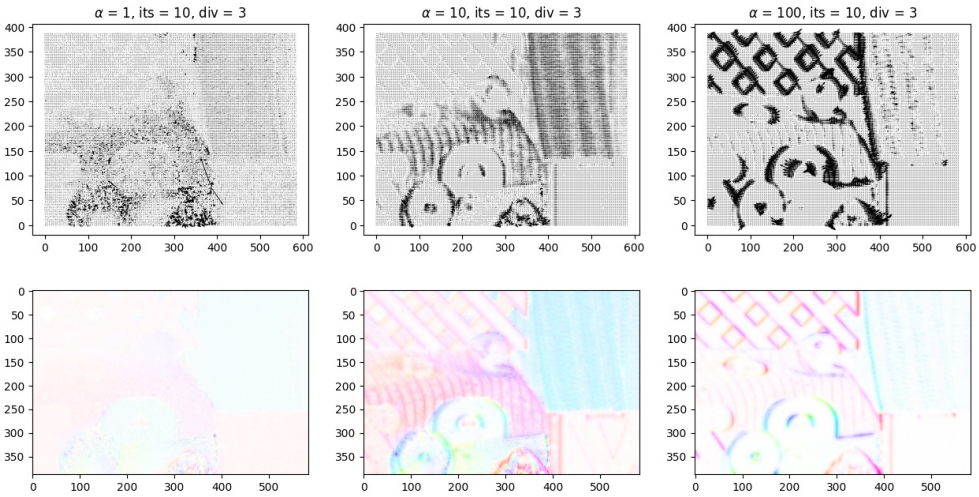

Figure 6: Computed optical flow[9] from sequence *Whale* 42

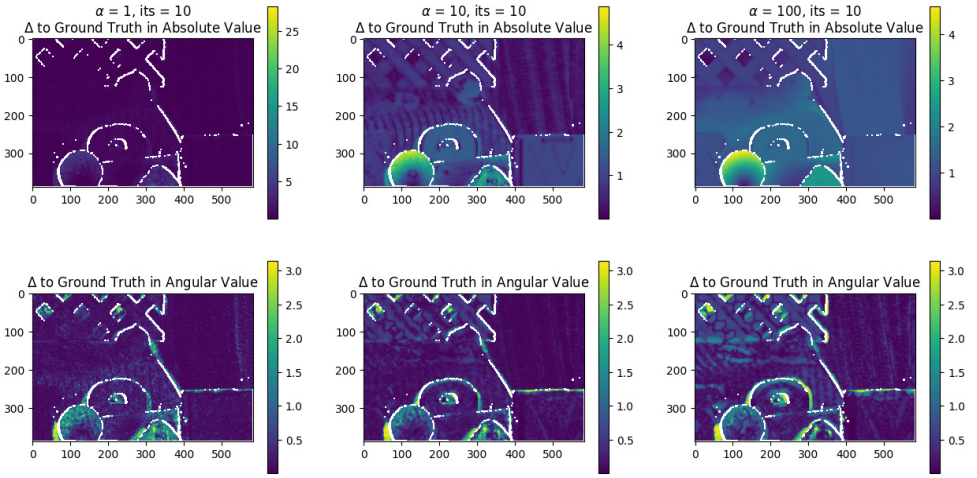

Figure 7: Angular and Endpoint Error for different α

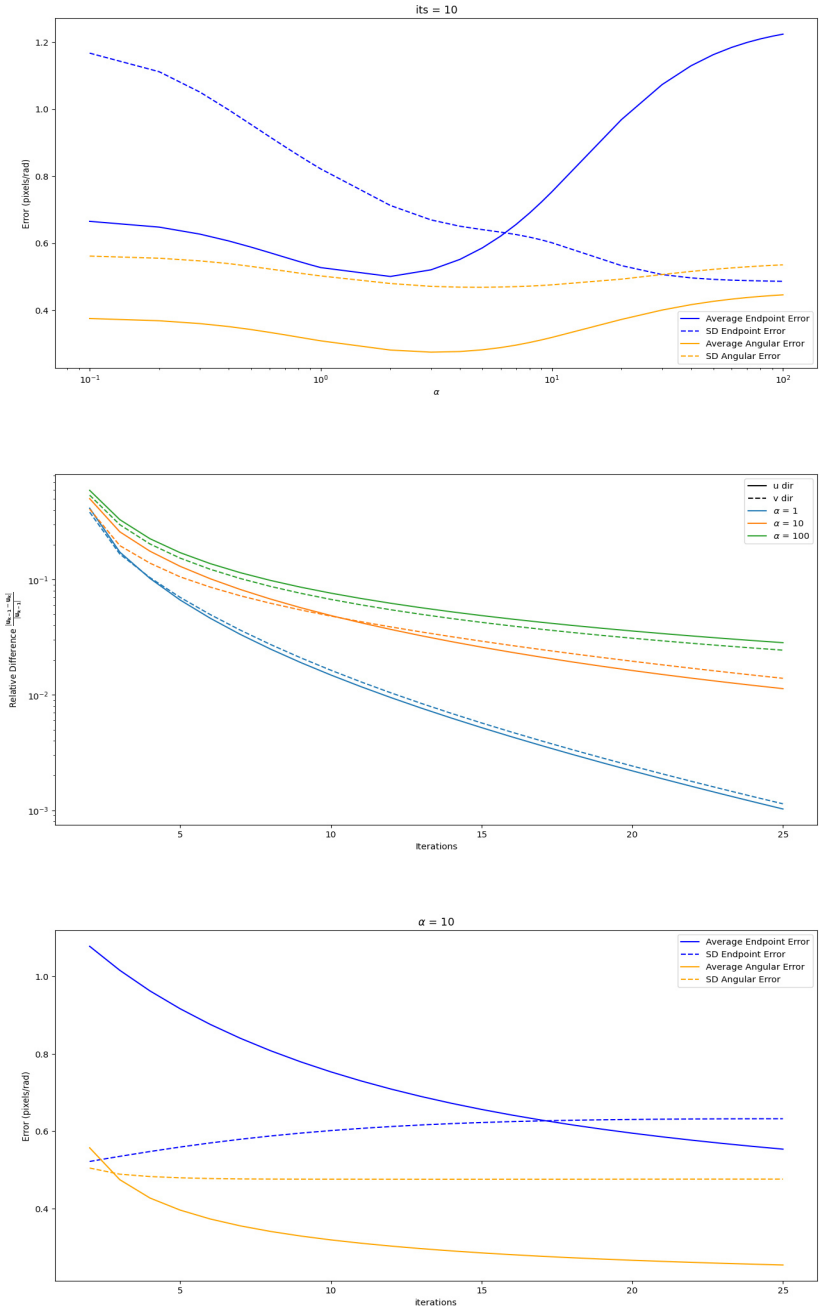

Figure 8: *top:* Error Statistics for different α and fixed iteration; *middle:* Relative Difference to previous iteration for different α; *bottom:* Error Statistics for different iterations and fixed α

3.6. Influence of Gaussian Blur

Application of a Gaussian Filter on image sequences before computing optical flow is a common attempt to improve results [32]. Due to the nature of Horn-Schunck Optical Flow and its smoothness constraint in form of the regularisation term, it generally does not perform as well around sharp flow edges and corners, but decent for gradual flow changes as found for instance in rotations of objects (Fig. 35). The use of Gaussian Blur aims so diminish exactly these rapid flow changes, by smoothing the image intensities themselves.

However, the expected effect can not be observed in every case. Fig. 9 depicts error statistics of two image sequences different in nature. While results of the *Hydrangea* Sequence (Fig. 41) admit of slight improvement in Average Errors when blurring the images, the same can not be said for the *Whale* Sequence. This seems to be due to the former featuring a distinct edge separating the background from the foreground at which the flow is rapidly changing. Additionally, the plant shows many intensity irregularities which get lessened by blurring. Opposed to this, the latter sequence features multiple smaller regions, where adjacent ones rarely do not move in unison.

Of course, since a Gaussian Filter alters the image intensities, valuable information may be lost at some stage. For many sequences, this is still outweighed by the advantages of smoothing up to some *radii* - for others, there may never be an advantage and subsequent improvement to begin with.

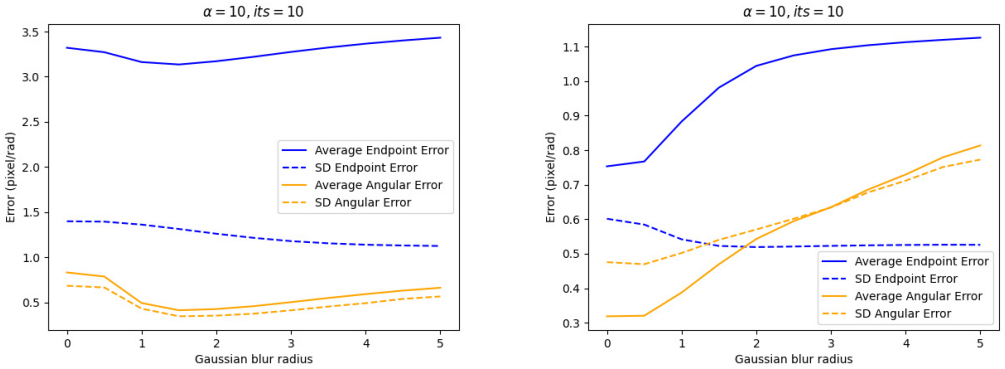

Figure 9: Error Statistics for increasing Gaussian blur[10]: *Hydrangea* Sequence (*lhs*); *Whale* Sequence (*rhs*)

[10] *radius* refers to the standard deviation of the gaussian kernel in function *GaussianBlur()* from package *Pillow* [11], $radius = 0$ being the unblurred image sequence

3.7. The Problem of Large Displacements

As previously discussed, the problem of Large Displacements remains an important challenge in Optical Flow Calculation. Sequence 50 displays two images where large object motion occurs. When applying the Horn-Schunck Optical-Flow Algorithm 1 to it, the results appear highly insufficient in comparison to the ground truth (Fig. 10) with the usual error statistics applied in Fig. 11[11]. The computed flow field appears particularly noisy, tolerable only in large areas of continuous flow. Further, the phenomenon of certain objects appearing twice is noticeable - consider the triangle structure in the lower left corner for instance.

Fig. 12 displays a simpler, albeit extreme[12] case of the problem - when an object moves too far from its initial position, and the algorithm has no access to constant brightness in the immediate area, it assumes the object to vanish at the original and reappear at the new position. In the flow field, this will be portrayed as a sink (i.e. background brightness *flowing* into the position) at the initial position and a source at the new position.

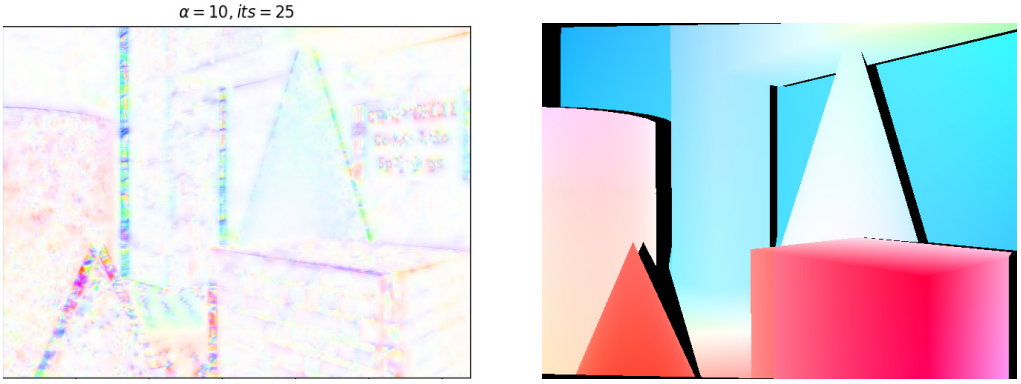

Figure 10: The Horn-Schunck Optical Flow algorithm struggles with sequences where large displacements occur: Computed Flow (*lhs*), Ground-Truth Flow (*rhs*)

Naively integrating the original Horn-Schunck Scheme into a multi-scale system serves only little to improve the situation. Here, the algorithm is repeatedly applied on the same image sequence with increasing resolution, always using the result of the last step as initial flow field of the next one[13]. Since a large motion may be visible in the computed flow of a low resolution image sequence, this hopes to *guide* the algorithm in the right direction.

However, as Fig. 36 and 37 are showing, even though the multi-scale version outperforms the default version of the algorithm in some areas and the result seems less noisy,

[11]Neither changes of α nor of the iteration count serve to improve the result
[12]i.e. very Large Displacement
[13]The details of such a pyramidal scheme will be discussed in the following chapter

the same issues still persist. Ultimately, the input given to the Horn-Schunck algorithm may only change convergence speed, not actual result. Even when given a good initial guess to start on, the Horn-Schunck Algorithm remains limited by approximation (1.3.3) and will thus always struggle when dealing with large enough motion.

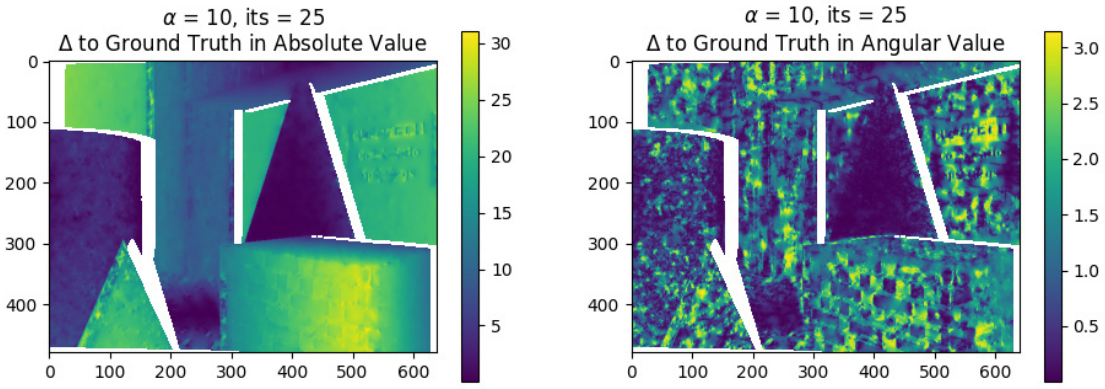

Figure 11: Angular and Endpoint Error for $\alpha = 5$, 25 iterations

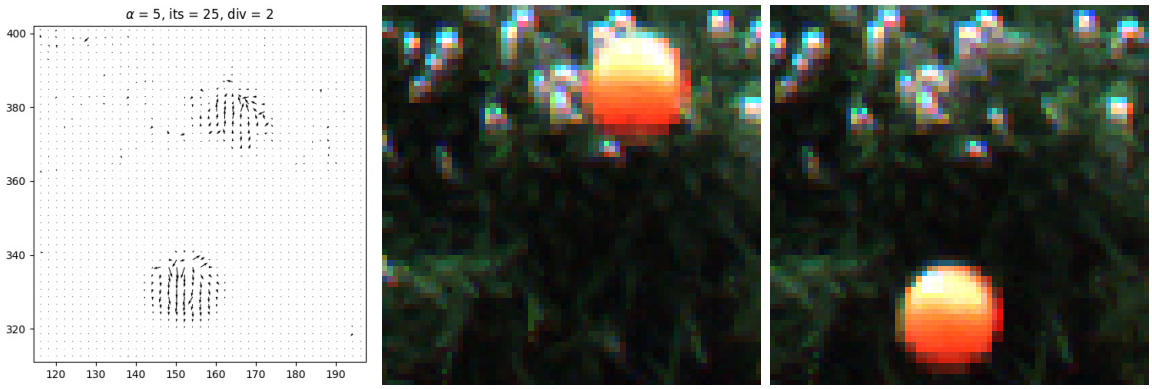

Figure 12: An object moving over multiple times its own size creating a source and a sink in the flow field (from *Backyard* Sequence (Fig. 45))

4. Improving on Horn and Schunck via a Warping Technique

In this chapter, the Horn and Schunck approach is improved by modifying the original brightness constancy assumption (1.3.1) in such a way, that the parameter of interest is a change in the flow field $\vec{dh}$, rather than the flow field $\vec{h}$ itself. This results in the usage of a version of a *warping technique*, as it has already been investigated in by Anandan et al. [2] and Bruhn et al. [8], for instance. Besides showing the extent of the improvement achieved through such a scheme, its justification as well as relation to non-linear approaches is pointed out.

4.1. Deriving the Scheme

Again, starting from (1.3.1),

$$I(\vec{x}, t) = I(\vec{x} + \vec{h}(\vec{x}), t + dt),$$

and substituting

$$\vec{h}(\vec{x}) = \vec{h_0}(\vec{x}) + \vec{dh}(\vec{x}),$$

$$I(\vec{x}, t) = I(\vec{x} + \vec{h_0}(\vec{x}) + \vec{dh}(\vec{x}), t + dt),$$

represents the brightness constancy assumption for an incremental scheme, where change $\vec{dh}$ should be optimized for and $\vec{h_0}$ refers to an already existing initial guess.

Notably, $\vec{h} = \begin{pmatrix} dx \\ dy \end{pmatrix}$ represents a displacement instead of a velocity field here. The schemes remain comparable still, due to the ability to freely choose $dt = 1$ in the basic Horn-Schunck approach, which makes differentiating between velocity and displacement field unnecessary. In the following due to simplicity, notation is again $\vec{h} = \begin{pmatrix} u \\ v \end{pmatrix}$.

After applying Taylor-Approximation to 1^{st} order to the right side,

$$I(\vec{x}, t) = I(\vec{x} + \vec{h_0}(\vec{x}), t + dt) + \vec{\nabla} I(\vec{x} + \vec{h_0}(\vec{x}), t + dt) \cdot \vec{dh}(\vec{x}), \tag{4.1.1}$$

is derived. Letting $I_1 = I(\vec{x}, t)$ and $I_2 = I(\vec{x}, t + dt)$ be the intensities of the two consecutive images, (4.1.1) may be written as,

$$\tilde{I_2} + \vec{\nabla} \tilde{I_2} \cdot \vec{dh} - I_1 = 0, \tag{4.1.2}$$

where $\tilde{I_2}$ practically refers to the application of the initial flow field $\vec{h_0}(\vec{x})$ onto the second image I_2, i.e. it is *warped* by the flow.

Naturally, (4.1.2) is still considered a gradient-based scheme, due to Taylor-

Approximation. Therefore, the guess of $\vec{h_0}(\vec{x})$ for the initial flow field need be *good enough*, so to avoid the change $\vec{dh}$ to be optimized for, needing to be too large. Generally, $\|\vec{dh}\| \ll \|\vec{h}\|$ holds. This prerequisite may be fulfilled by usage of a pyramidal scheme, under the same reasoning as given in Chapter 4.5: Optical flow estimation at low resolution sufficiently captures large motion.
(4.1.2) is now substituted into (3.1.1) as a new data term, giving,

$$J_W[\vec{dh}] = \int_\Omega (\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \vec{dh} - I_1)^2 + \alpha^2 \|\nabla(\vec{h_0} + \vec{dh})\|_F^2, \tag{4.1.3}$$

as the energy functional, where the regularisation-*smoothness*-term is still that of the original Horn-Schunck approach, now also employing the increment $\vec{dh}$.

Setting up the Euler-Lagrange-Equations works similar to Chapter 3.2, with analogous notational convention:

$$\begin{aligned} \frac{\partial F}{\partial du} - \frac{\partial}{\partial x}\frac{\partial F}{\partial du_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial du_y} = 0 \\ \frac{\partial F}{\partial dv} - \frac{\partial}{\partial x}\frac{\partial F}{\partial dv_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial dv_y} = 0, \end{aligned} \tag{4.1.4}$$

with $F = \left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \begin{pmatrix} du \\ dv \end{pmatrix} - I_1\right]^2 + \alpha^2 \left\| \vec{\nabla} \begin{pmatrix} u_0 + du \\ v_0 + dv \end{pmatrix} \right\|_F^2.$

Then (4.1.4) becomes,

$$\begin{aligned} 2\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \begin{pmatrix} du \\ dv \end{pmatrix} - I_1\right] \tilde{I}_{2_x} - \frac{\partial}{\partial x}\left(2\alpha^2 \frac{\partial(u_0 + du)}{\partial x}\right) - \frac{\partial}{\partial y}\left(2\alpha^2 \frac{\partial(u_0 + du)}{\partial y}\right) = 0 \\ 2\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \begin{pmatrix} du \\ dv \end{pmatrix} - I_1\right] \tilde{I}_{2_y} - \frac{\partial}{\partial x}\left(2\alpha^2 \frac{\partial(v_0 + dv)}{\partial x}\right) - \frac{\partial}{\partial y}\left(2\alpha^2 \frac{\partial(v_0 + dv)}{\partial y}\right) = 0 \\ \left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \vec{dh} - I_1\right] \tilde{I}_{2_x} - \alpha^2 \triangle (u_0 + du) = 0 \\ \left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \vec{dh} - I_1\right] \tilde{I}_{2_y} - \alpha^2 \triangle (v_0 + dv) = 0 \end{aligned} \tag{4.1.5}$$

These two Euler-Lagrange Equations are subsequently rearranged into a linear system of equations in the following.

$$\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \vec{dh} - I_1\right] \vec{\nabla}\tilde{I_2} - \alpha^2 \triangle (\vec{h_0} + \vec{dh}) = 0$$

Rewriting $\vec{dh} = \vec{h^{k+1}} - \vec{h^k}$, with $\vec{h_0} = \vec{h^k}$ simply referring to the previous iteration, then leads to,

$$\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot (\vec{h}^{k+1} - \vec{h^k}) - I_1\right] \vec{\nabla}\tilde{I_2} - \alpha^2 \triangle \vec{h}^{k+1} = 0, \tag{4.1.6}$$

and,

$$\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot (\vec{h}^{k+1} - \vec{h^k}) - I_1\right] \vec{\nabla}\tilde{I_2} - \alpha^2 (\vec{\vec{h}}^k - \vec{h}^{k+1}) = 0,$$

with usage of stencil (3.2.3) in order to approximate[14] $\triangle \vec{h}^{k+1} \approx \kappa(\vec{\vec{h}}^k - \vec{h}^{k+1})$, with $\kappa = 3$.

[14] Using $\vec{\vec{h}}^k$ instead of $\vec{\vec{h}}^{k+1}$ simplifies computation drastically

Here again, κ is omitted as in Chapter 3.2. Then,

$$\left[\tilde{I_2} + \vec{\nabla}\tilde{I_2}\vec{h}^{k+1} - \vec{\nabla}\tilde{I_2}\vec{h^k} - I_1\right]\vec{\nabla}\tilde{I_2} - \alpha^2\vec{\bar{h}}^k = -\alpha^2\vec{h}^{k+1}$$

$$\left[\tilde{I_2} - \vec{\nabla}\tilde{I_2}\vec{h^k} - I_1\right]\vec{\nabla}\tilde{I_2} - \alpha^2\vec{\bar{h}}^k = -\alpha^2\vec{h}^{k+1} - \vec{\nabla}\tilde{I_2}\vec{\nabla}\tilde{I_2}^T\vec{h}^{k+1}.$$

Therefore, a linear system of equations is recovered,

$$\underbrace{\left(\alpha^2 I + \vec{\nabla}\tilde{I_2}\vec{\nabla}\tilde{I_2}^T\right)}_{\mathrm{A}}\underbrace{\vec{h}^{k+1}}_{\vec{x}} = \underbrace{\left[\vec{\nabla}\tilde{I_2}\vec{h^k} + I_1 - \tilde{I_2}\right]\vec{\nabla}\tilde{I_2} + \alpha^2\vec{\bar{h}}^k}_{\vec{b}}. \tag{4.1.7}$$

Here, I denotes the 2×2 identity matrix. At every iteration, a previous *good enough* guess $\vec{h^k}$ and its Laplace-Approximation $\vec{\bar{h}}^k$ need to be available. Additionally, image intensities of the warped second image $\tilde{I_2}$ and the corresponding derivatives $\vec{\nabla}\tilde{I_2}$ are needed.

(4.1.7) may be expressed similarly to (3.2.5) by again using *Cramer's Rule* starting from (4.1.6) as $A\vec{x} = \vec{b}$,

$$\begin{pmatrix} \tilde{I_{2x}}^2 + \alpha^2 & \tilde{I_{2x}}\tilde{I_{2y}} \\ \tilde{I_{2x}}\tilde{I_{2y}} & \tilde{I_{2y}}^2 + \alpha^2 \end{pmatrix}\begin{pmatrix} u^{k+1} \\ v^{k+1} \end{pmatrix} = \begin{pmatrix} -\tilde{I_2}\tilde{I_{2x}} + \tilde{I_{2x}}^2 u^k + \tilde{I_{2x}}\tilde{I_{2y}}v^k + I_1\tilde{I_{2x}} + \alpha^2\bar{u}^k \\ -\tilde{I_2}\tilde{I_{2y}} + \tilde{I_{2x}}\tilde{I_{2y}}u^k + \tilde{I_{2y}}^2 v^k + I_1\tilde{I_{2y}} + \alpha^2\bar{v}^k \end{pmatrix},$$

with determinants,

$$det(A) = \alpha^2(\tilde{I_{2x}}^2 + \tilde{I_{2y}}^2 + \alpha^2)$$

$$det(A_u) = \alpha^2[-\tilde{I_2}(\tilde{I_{2x}} + I_1\tilde{I_{2x}} + u^k\tilde{I_{2x}}^2 + v^k\tilde{I_{2x}}\tilde{I_{2y}} + \bar{u}^k(\tilde{I_{2y}}^2 + \alpha^2) - \bar{v}^k\tilde{I_{2x}}\tilde{I_{2y}}]$$

for the case of u^{k+1}. Finally arriving at update schemes,

$$\begin{aligned} u^{k+1} &= \bar{u}^k - \frac{\tilde{I_{2x}}(\tilde{I_2} - I_1 + \tilde{I_{2x}}(\bar{u}^k - u^k) + \tilde{I_{2y}}(\bar{v}^k - v^k))}{\tilde{I_{2x}}^2 + \tilde{I_{2y}}^2 + \alpha^2}, \\ v^{k+1} &= \bar{v}^k - \frac{\tilde{I_{2y}}(\tilde{I_2} - I_1 + \tilde{I_{2x}}(\bar{u}^k - u^k) + \tilde{I_{2y}}(\bar{v}^k - v^k))}{\tilde{I_{2x}}^2 + \tilde{I_{2y}}^2 + \alpha^2}, \end{aligned} \tag{4.1.8}$$

where v^{k+1} is derived analogously.

4.2. Pyramidal Approach

As described in Chapter 1.4, employing multi-scale, alias pyramidal schemes in optical flow computation is no novelty. The obvious appeal is to resolve large motion by either applying a Gaussian Filter or decreasing the image resolution to such an extent so that a basic non-large-displacement algorithm computes sufficiently good results. It is further explicitly necessary to employ such an algorithm at the initial stage, since no *good enough*

guess $\vec{h_0}$ can be assumed to be available a priori for any given image sequence. For all following stages, the scheme derived in Chapter 4.1 is then used and the result of the previous stage is chosen as initial condition.

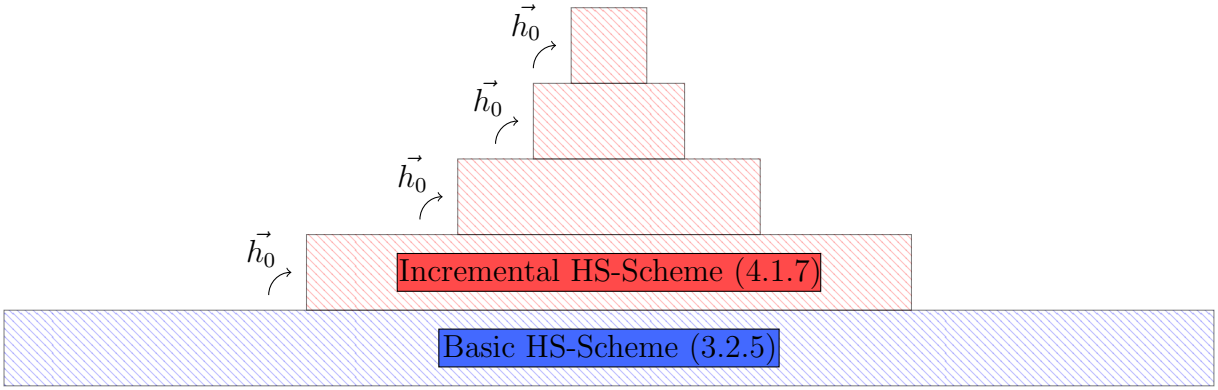

Figure 13: Pyramidal scheme in the present case

More specifically, the pyramidal scheme used in the following is based on using quadratic images: the pixel count from one image to the next is increased by a factor of 4, the width and height of the images in each pyramid step are powers of 2. This necessitates not only resizing the initial image sequence but also changing its aspect ratio, which may introduce errors. However, using a pyramid such as this greatly simplifies the implementation.

Figure 13 essentially equates the width of one pixel at each stage with the width of the rectangles shown. From bottom to top, the width of the images in pixels could for instance increase like $\{32, 64, 128, 256, 512\}$. The initial image resolution should be chosen in a way that the majority of the motion is best resolved, but is usually chosen as 16×16 or 32×32. The last step should optimally be closest to the original image resolution, since conversion of the flow field from the quadratic representation to that size is favourable[15] and conversion errors are minimized that way. As indicated, the initial flow field $\vec{h_0}$ at each stage is taken from the previous stage, except for the basic Horn-Schunck algorithm, where the initial flow field will again be chosen as the zero flow (3.4.3).

4.3. Implementation

Again, all code is provided at `https://github.com/schwarzflo/masters_thesis`.
Relevant File: *incr_warping_ms.py*
Concerning the individual terms in (4.1.7), α and I_1 are trivially available and $\vec{\bar{h}}^k$ is computed like in Chapter 3.4, via *scipy.ndimage.convolve()*.

[15] e.g. in order to compare with ground-truth flow

In order to warp the second image $I_2 \rightarrow \tilde{I}_2$, the current flow field estimation $\vec{h}^k$, is applied onto a matrix storing the intensity values of the second image,

$$I_2(i + u^k, j + v^k) = \tilde{I}_2(i, j), \tag{4.3.1}$$

i.e. with ground truth flow as input and no occlusions[16] present, $\tilde{I}_2 = I_1$ holds. It is necessary to introduce boundary conditions (3.4.2), since a pixel (i, j) could be warped to the outside of the image. Due to the discrete nature of the images, any values for u, v will be rounded down, e.g $\vec{h}(i, j) = \begin{pmatrix} 1.2 \\ 0 \end{pmatrix}$ will warp the intensity value from $(i + 1, j)$ to (i, j), which has empirically proven to be the best choice.

The warped derivatives $\vec{\nabla}\tilde{I}_2$ are further computed via a simple central difference approach, in x and y direction respectively,

$$\begin{aligned} \tilde{I}_{2x} &= \frac{\tilde{I}_2(i + 1, j) - \tilde{I}_2(i - 1, j)}{2}, \\ \tilde{I}_{2y} &= \frac{\tilde{I}_2(i, j + 1) - \tilde{I}_2(i, j - 1)}{2}. \end{aligned} \tag{4.3.2}$$

In all definitions above, indexes i and j denote the position in the intensity matrices. Choosing the definition of $\vec{\nabla}\tilde{I}_2$ such as this and not applying derivatives first and warping subsequently, i.e. $(\vec{\nabla}I_2)(i + u, j + v)$, is reasoned with better performance and warping for almost all reviewed sequences. In general, equality between these two approaches does not hold,

$$(\vec{\nabla}I_2)(i + u, j + v) \neq \vec{\nabla}I_2(i + u, j + v). \tag{4.3.3}$$

Algorithm 2 outlines the key points of the implementation:
Besides the original images of the sequence at hand and the side lengths of the resized images in the pyramid, two separate sets of parameters have to be chosen as input: a regularisation weight $(\alpha_{hs}, \alpha_{ihs})$ for each of the two present schemes, as well as an iteration count (it_{hs}, it_{ihs}) or alternatively, an error threshold. As it is later discussed, the best choice of these parameters appears non-trivial and counter-intuitive at times.

Resizing the images is done via the function *Image.resize*, of *Python* package *Pillow* [11]. Although different resampling filters may be chosen, none substantially change the end result, therefore default cubic interpolation is used.

Between pyramid stages, a conversion takes place $\vec{h}^k_{prev} \rightarrow \vec{h}^0_{next}$. Not only is the initial field for the next stage set, but the resolution is increased along with the actual values being doubled. The reason for this is that the computed numerical values naturally depend on the resolution of the images used[17]. An increase in image-side-length by 2 results in a needed change of the values by that factor.

[16] practically impossible in real life scenes

[17] A motion in pixels in a 32×32 image will be half as large as in a 64×64 image

Lastly, *numpy.linalg.solve* employs LU-Factorisation to solve the linear system of equations.

The algorithm runtime R_{IHS} will naturally be dominated by the last pyramid stage $(m_{max} \cdot n_{max})$, and should increase by a factor of 4 each time. Additionally, the following FLOPS are assigned to the various operations:

- $\tilde{\boldsymbol{I}}_{\boldsymbol{2}}$: 1 FLOP for transitioning in the storage matrix (4.3.1), for both x and y direction
- $\vec{\boldsymbol{\nabla}}\tilde{\boldsymbol{I}}_{\boldsymbol{2}}$: 2 FLOPS using (4.3.2), for both x and y direction
- $\vec{\boldsymbol{h}}^{\boldsymbol{k}}$: 9 FLOPS using (3.2.3), for both u and v
- ***numpy.linalg.solve***: 13.3 FLOPS for solving linear system of equations (4.1.2) with LU-Factorisation of 2×2 matrix

Therefore, in any given stage a factor of $1 \cdot 2 + 2 \cdot 2 + 2 \cdot 9 + 13.3 = 37.3$ must be employed. Then, the following holds,

$$\begin{aligned} R_{IHS} &= \underbrace{(24\frac{1}{it_{hs}} + 42) \cdot it_{hs} \cdot m_{init} \cdot n_{init}}_{\text{Basic HS-Scheme}} \\ &+ 37.3 \cdot it_{ihs} \cdot \underbrace{[.. + \frac{1}{16}(m_{max} \cdot n_{max}) + \cdot\frac{1}{4}(m_{max} \cdot n_{max}) + m_{max} \cdot n_{max}]}_{\text{Incremental HS-Scheme}}, \end{aligned} \tag{4.3.4}$$

where it_{ihs} is similarly pulled out. The Basic HS-Scheme is executed only once at initial resolution $(m_{init} \cdot n_{init})$, while the Incremental HS-Scheme is executed one time less than k_{res}, the number of pyramid stages, excluding exactly that initial execution. Additionally, the sum $\sum_0^{k_{res}-1} \left(\frac{1}{4}\right)^2 = 1 + \frac{1}{4} + \frac{1}{16} + ..$ may be identified, which approximates to $\frac{4}{3}$, for pyramids over 3 stages tall, i.e. $k_{res} > 3$. Finally, the runtime R_{IHS} comes out to be,

$$\begin{aligned} R_{IHS} &= (24\frac{1}{it_{hs}} + 42) \cdot it_{hs} \cdot m_{init} \cdot n_{init} + 49.7 \cdot it_{ihs} \cdot m_{max} \cdot n_{max}, \\ &\in O(m_{max}^2), \end{aligned} \tag{4.3.5}$$

for squared images. This circumstance is verified in Fig. 14, where each measurement is taken after completion of the indicated stage, including the first stage of basic Horn-Schunck algorithm at 16×16 pixels.

Data: images, resolutions
Data: α_{hs}, iteration count it_{hs}/error threshold for hs
Data: α_{ihs}, iteration count it_{ihs}/error threshold for incr. hs
for *resolutions* **do**
  Resize images;
  if *lowest resolution* **then**
    Basic HS-Algorithm (1);
  end
  else
    Convert $\vec{h}^k_{prev}$ and set to $\vec{h}^0_{next}$
    while *iteration count/error threshold not met* **do**
      Compute Warped Image $\tilde{I}_2$;
      Compute Gradients of Warped Image $\vec{\nabla}\tilde{I}_2$ (4.3.2);
      Compute $\vec{\bar{h}}^k$ via scipy.ndimage.convolve[18];
      for *width of image* **do**
        for *height of image* **do**
          Solve (4.1.7) with numpy.linalg.solve;
          Update $\vec{h}^k$;
        end
      end
    end
  end
end

Algorithm 2: Incremental Iterative Optical Flow Algorithm with Warping

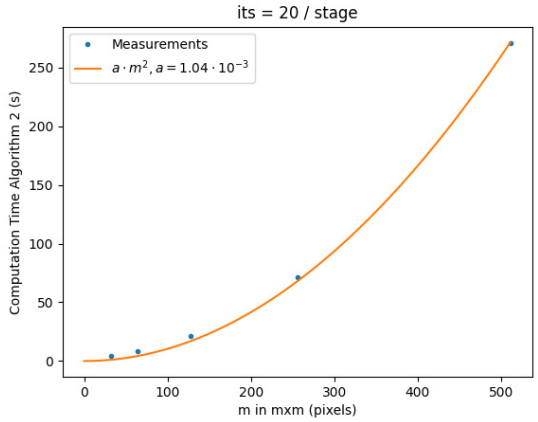

Figure 14: The runtime is expectedly fitted by $a \cdot m^2$, with iteration count $it = 20$ per stage

[18]Scipy Package [38]: apply stencil (3.2.3) via convolution to compute $\vec{\bar{h}}^k$

4.4. Justification and Relation to Non-Linear Schemes

The effectiveness of the algorithm in dealing with Large Displacements is not arbitrary - it rather stems from close relationship with non-linear schemes, which are usually introduced to combat the problem, i.e.,

$$J_{NL} = \frac{1}{2}\int_{\Omega} \Psi\left(I_2(\vec{x}+\vec{h}) - I_1(\vec{x})\right) d\vec{h} + R(\vec{h}), \tag{4.4.1}$$

as energy functional based on brightness constancy. In [28] for instance, a variant of (4.4.1) is used with Horn-Schunck regularisation term (3.1.2) and subsequent introduction into a multi-scale pyramidal scheme is investigated.

A connection of incremental schemes with warping in general and non-linear approaches was already made in [32], the following shows the relation. Starting from the semi-implicit fixed-point scheme based on the Euler-Lagrange Equations for J_{NL} set up by the authors in [32][19],

$$\Psi'\left(\|I_2(\vec{x}+\vec{h}^{k+1}) - I_1(\vec{x})\|^2\right)\left(I_2(\vec{x}+\vec{h}^{k+1}) - I_1(\vec{x})\right)\nabla I_2(\vec{x}+\vec{h}^{k}) + R'(\vec{h}^{k+1}) = 0.$$

Now, linearising $I_2(\vec{x}+\vec{h}^{k+1}) \approx I_2(\vec{x}+\vec{h}^{k}) + \nabla I_2(\vec{x}+\vec{h}^{k})\cdot d\vec{h}^{k}$ and writing $\vec{h}^{k+1} = \vec{h}^{k} + d\vec{h}^{k}$,

$$\Psi'\left(\|I_2(\vec{x}+\vec{h}^{k}) + \nabla I_2(\vec{x}+\vec{h}^{k})\cdot d\vec{h}^{k}\|^2\right)$$
$$\left(I_2(\vec{x}+\vec{h}^{k}) + \nabla I_2(\vec{x}+\vec{h}^{k})\cdot d\vec{h}^{k} - I_1(\vec{x})\right)\nabla I_2(\vec{x}+\vec{h}^{k}) + R'(\vec{h}^{k}+d\vec{h}^{k}) = 0.$$

Again, the warped notation $\tilde{I}_2 = I_2(\vec{x}+\vec{h}^{k})$ and $\nabla\tilde{I}_2 = \nabla I_2(\vec{x}+\vec{h}^{k})$ is introduced, leaving us with,

$$\Psi'\left(\|\tilde{I}_2 + \nabla\tilde{I}_2\cdot d\vec{h}^{k} - I_1\|^2\right)\left(\tilde{I}_2 + \nabla\tilde{I}_2\cdot d\vec{h}^{k} - I_1\right)\nabla\tilde{I}_2 + R'(\vec{h}^{k}+d\vec{h}^{k}) = 0.$$

This however, just describes the Euler-Lagrange Equations for a more generalised version of Energy Functional J_W (4.1.3),

$$J = \frac{1}{2}\int_{\Omega} \Psi\left(\|\tilde{I}_2 + \nabla\tilde{I}_2\cdot d\vec{h} - I_1\|^2\right) d\vec{h} + R(\vec{h_0}+d\vec{h}),$$

using initial guess $\vec{h_0}$ instead of previous guess $\vec{h}^{k}$ and $d\vec{h}$ instead of $d\vec{h}^{k}$ therefore highlights the connection between non-linear schemes and iterative incremental models with warping, as well as explains the success of the latter.

[19] Note that R' denotes the derivative of a functional, as opposed to Ψ'

4.5. Solving Large Displacements

As described in Chapter 3.7, the basic Horn and Schunck Algorithm 1 HS_b as well as a naive extension to a pyramid approach prove insufficient in dealing with the problem of large displacements. In the following, the Iterative Incremental approach with Warping 2 HS_{inc} is applied to such problematic image sequences, demonstrating significant improvement in performance.

Starting from the *Brickbox* sequence (Fig. 50), which was examined (Figs. 10, 37) by the mentioned algorithms, improvements become visible already without numerical analysis (Fig 15). Obvious errors are only present in areas where occlusion occurs, i.e. where warping will be at least partly un-successful. These are however usually also those areas, where optical flow may not be determined, since information is missing in the first place. Other error prone regions include image borders, where possible disappearance of objects or patterns leads to flow discrepancies (e.g. left edge Fig. 15, lhs).

As can be further observed in Fig. 16, rhs, some areas around the occluded parts are affected by erroneous warping in those regions, i.e. along the right edge of the cylindrical structure in the foreground. In order to improve the readability of Fig. 16, lhs, the extreme points on the left edge of the scalar field were removed, since they negatively affect visual representation of other problematic regions. However, these extreme points will still be used for numerical evaluation, since their existence is caused by insufficient behaviour of HS_{inc} around image edges, rather than unsolvable occlusions.

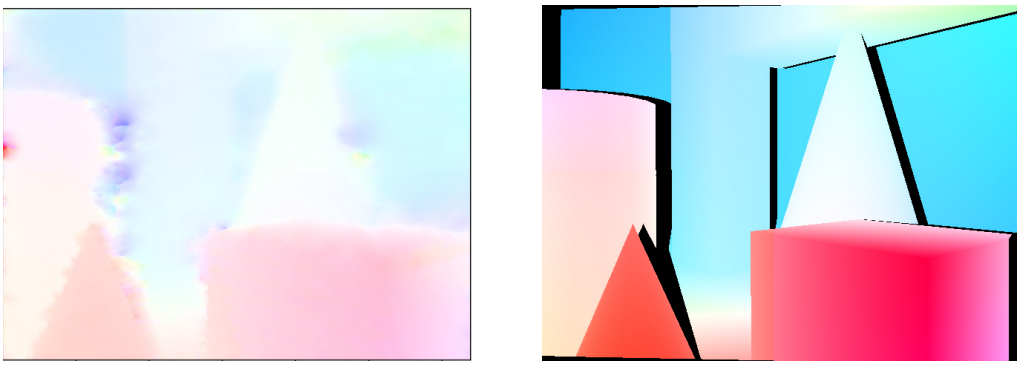

Figure 15: The incremental Horn-Schunck approach with warping produces visually pleasing results[20]: Computed Flow (*lhs*), Ground-Truth Flow (*rhs*)

[20] due to some erroneous extreme values the colors of the whole image plain are damped

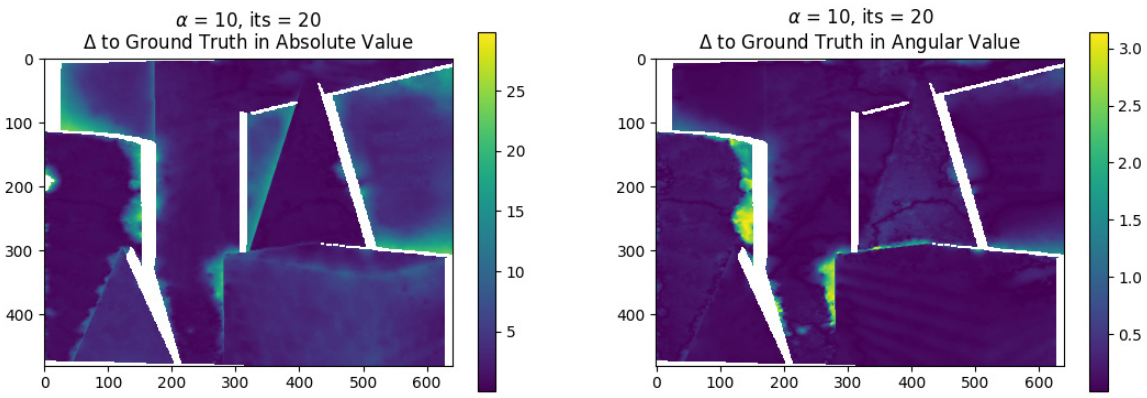

Figure 16: Angular and Endpoint Error for $\alpha_{hs} = 1$, $it_{hs} = 25$, $\alpha_{ihs} = 10$, $it_{ihs} = 20$

In the following, the computed flow fields by HS_b and HS_{inc} are shown for additional sequences taken from [24], originally produced in [26] where movements contained are always declared as *large*. All observed sequences may be found in the Appendix, a more precise numerical comparison of performances in Endpoint- and Angular Errors in reference to ground truth for these sequences are depicted in Table 1.

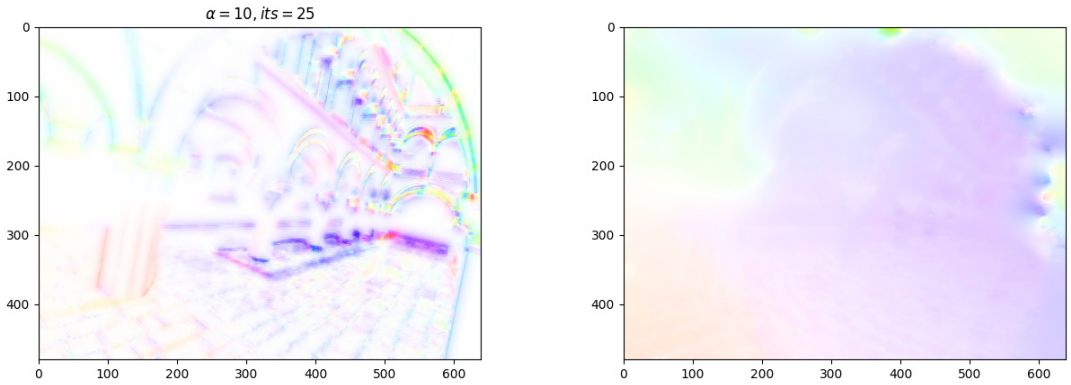

Figure 17: *Sponza* Sequence 49: Basic Horn-Schunck Algorithm (*lhs*), Incremental Horn-Schunck Algorithm with Warping (*rhs*)

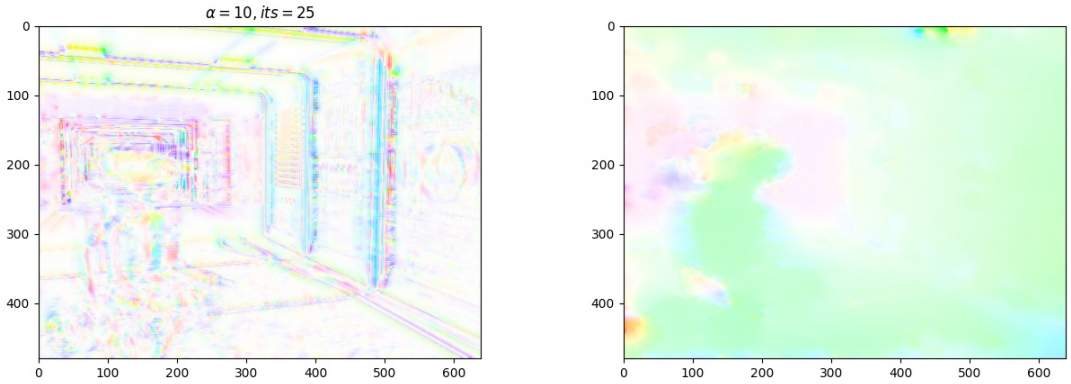

Figure 18: *Robot* Sequence 47: Basic Horn-Schunck Algorithm (lhs), Incremental Horn-Schunck Algorithm with Warping (rhs)

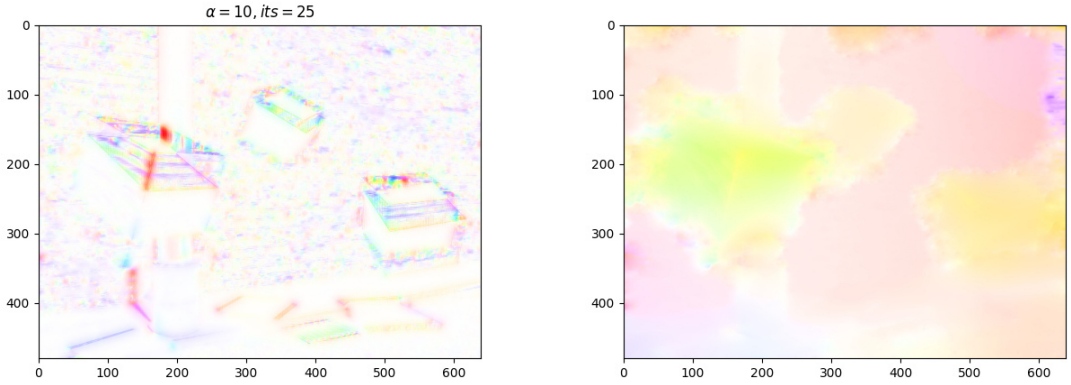

Figure 19: *Woodbox* Sequence 48: Basic Horn-Schunck Algorithm (lhs), Incremental Horn-Schunck Algorithm with Warping (rhs)

Notes regarding Figures 17 - 19:

- **Figure 17**: The *Sponza* sequence entails very smooth motion, which makes it reasonably solvable by HS_b. As the flow is ever over-pronounced along edges and corners, general flow orientation fits for large parts of the images. Still, an improvement is again provided by usage of HS_{inc} - with some irregularities on the right image-side, again due to occlusions. Nevertheless, satisfying results are obtained.
- **Figure 18**: HS_b only works in a very limited region in the background of the images - where motion is small enough, and fails completely in the foreground and around various structures. HS_{inc} on the other hand produces satisfying results for vast areas of the images, with significant errors only around the robot in the image center, where occlusions are present and along the image edges, which again, stem from disappearing structures of some kind.

- **Figure 19**: This sequence is the most demanding sequence movement-wise, to a point where even HS_{inc} produces insufficient results in multiple areas of the images, e.g. around the boxes. Further, even though background movement is minimal, the complex pattern of the bricks and subsequent intensity irregularities tend to pose problems around the image-edges. Still, significant improvement is made as HS_b can not deal with box-movement and the background-irregularities at all.

In all sequences, the parameters for HS_b ($\alpha = 10, it = 25$), and HS_{inc} ($\alpha_{hs} = 1, it_{hs} = 25, \alpha_{ihs} = 10, it_{ihs} = 20$) are chosen equally. Although this means they are not optimized for each sequence, they represent an educated guess working reasonably well for all sequences examined. Increased run times may lead to improvements by a few percentages on both sides in any case. A starting resolution of 32×32 is employed for all but the *Woodbox* Sequence, where 16×16 is chosen.

Overall, depending on the sequence at hand, the decrease in average Endpoint Error $\bar{EE}$ and average Angular Error $\bar{AE}$ ranges from around 50% down to under 25% of the corresponding errors for HS_b. Standard deviations (σ_{EE}, σ_{AE}) elicit similar behaviour, although problematic, unsolvable regions will always exist and create significant values in those measures even for HS_{inc}.

	Sponza		*Brickbox*		*Robot*		*Woodbox*	
	HS_b	HS_{inc}	HS_b	HS_{inc}	HS_b	HS_{inc}	HS_b	HS_{inc}
$\bar{EE}(px)$	6.70	3.07	12.91	3.76	15.75	4.45	17.59	7.60
$\bar{AE}(rad)$	0.84	0.30	1.01	0.21	1.19	0.27	1.13	0.35
$\sigma_{EE}(px)$	3.38	3.41	8.39	3.81	9.48	5.06	12.01	8.00
$\sigma_{AE}(rad)$	0.67	0.49	0.85	0.36	0.84	0.42	0.79	0.51

Table 1: Averages and Standard Deviations for Angular and Endpoint Errors

As before, regions where flow is un-computable due to lack of information[21] were excluded from numerical analysis. As previously described, the computed flow in the highest stage is converted to the original image's resolution. Because of this, conversion errors may occur but are hard to estimate. If not mentioned otherwise, 512×512 defines the last pyramid stage.

Moving away from Large Displacements, the algorithm also fares better in scenes like *Hydrangea* (Fig. 41). This proposes that in scenes with many intensity irregularities but little disappearing structures, HS_{inc} has the upper hand on HS_b, even without the main argument of Large Displacements in play. However, in other scenes with similar motion distances, which are already handled very well by HS_b, e.g. *Whale* (Fig. 42), application of an iterative incremental approach with warping should not be expected to improve the result significantly or even at all. Here, subtle errors in warping along edges and corners almost outweigh the gain that can be made in comparison to HS_b.

[21] white areas in ground truth flow

Processing another image sequence of similar kind, *Dimetrodon* (Fig. 43), it becomes obvious that it is of tremendous importance to properly choose the starting resolution in HS_{inc}, as will further be explained in Chapter 4.7. While HS_b does resolve the optical flow in the sequence reasonably well, it is easily outperformed by HS_{inc} choosing starting resolutions of 64×64 or higher. This makes sense, since in sequences of small displacement the basic Horn-Schunck algorithm in the first pyramid stage resolves the image well even at high resolution. A starting resolution at 256×256 gives the best initial flow field $\vec{h_0}$ for the subsequent warping scheme at 512×512 and produces the best results.

Table 2 depicts the performance of four non-large displacement sequences, including those already mentioned for the two algorithms in discussion.

	$Hydrangea$		$Whale$		$Dimetrodon$		$Venus$	
	HS_b	HS^*_{inc}	HS_b	HS^{**}_{inc}	HS_b	HS^{**}_{inc}	HS_b	HS^*_{inc}
$\bar{EE}(px)$	3.29	1.57	0.61	0.52	1.76	0.62	3.56	2.9
$\bar{AE}(rad)$	0.82	0.22	0.26	0.27	0.62	0.17	0.94	0.44
$\sigma_{EE}(px)$	1.48	1.36	0.64	0.46	0.86	0.45	2.00	3.36
$\sigma_{AE}(rad)$	0.71	0.18	0.46	0.41	0.65	0.14	0.78	0.61

Table 2: Averages and Standard Deviations for Angular and Endpoint Errors
Starting resolutions:
*32×32
**256×256

Even though parameters were not optimized particularly well in each algorithm, the general take-away is that HS_{inc} can certainly improve performance in comparison to HS_b for non-large displacement sequences. The extent of the improvement is however highly dependant on the image sequence at hand and the starting resolution chosen for HS_{inc}.

4.6. Contrast Invariance

As described in [20], contrast dependence in context of optical flow calculation may negatively affect results and wrongfully favour certain motion over other. However, contrast invariance may be recovered via dividing the data term in the energy functional such as (3.1.1) through a term ω,

$$J_\omega[\vec{h}] = \int_\Omega \left(\frac{\vec{\nabla} I \cdot \vec{h} + I_t}{\omega} \right)^2 + R(\vec{h}),$$
$$\omega = \sqrt{\|\bar{\nabla} I\|^2 + \epsilon^2}, \qquad \bar{\nabla} I = \begin{pmatrix} I_x \\ I_y \\ I_t \end{pmatrix}, \qquad \epsilon > 0. \tag{4.6.1}$$

In fact, the contrast dependence may be observed when considering famous Sequence *Hamburg Taxi* 39, the result of applying Algorithm 1 on it shown in Fig. 40, left. While the movement of the white car in the center is modelled well, the same can not be said about the black car on the bottom left, even though roughly the same displacement field is applied to it.

However, the introduction of ω in the described form into the iterative scheme does not amplify the under-represented motion of low contrast objects significantly, and rather increases the noise in the result as seen in Fig. 40, right.

Interestingly, contrast dependence is not as much of a factor when using Algorithm 2, even though the basic Horn-Schunck algorithm, which fails to capture the dark vehicle at the bottom in Fig. 40, is employed in the first pyramid stage at 32×32 pixels. In fact, since the movement of the dark car appears even larger than that of the white, it is rightfully displayed that way. Fig. 20 shows the satisfying representation of the actual flow[22], although the tree in the foreground on the right makes warping ineffective in that area which reduces performance.

Again, introducing weight $\frac{1}{\omega}$ into the energy functional (4.1.3) leads to,

$$J_{W\omega}[\vec{dh}] = \int_\Omega \left(\frac{\tilde{I_2} + \vec{\nabla}\tilde{I_2} \cdot \vec{dh} - I_1}{\omega} \right)^2 + \alpha^2 \|\nabla(\vec{h_0} + \vec{dh})\|_F^2, \tag{4.6.2}$$

and further into a system of equations,

$$\underbrace{\left(\omega^2\alpha^2 I - \vec{\nabla}\tilde{I_2}\vec{\nabla}\tilde{I_2}^T\right)}_{\mathrm{A}} \underbrace{\vec{h}^{k+1}}_{\vec{x}} = \underbrace{\left[\tilde{I_2} - \vec{\nabla}\tilde{I_2}\vec{h^k} - I_1\right]\vec{\nabla}\tilde{I_2} - \omega^2\alpha^2\vec{h}^k}_{\vec{b}}. \tag{4.6.3}$$

Here again, ω simply scales the regularisation term itself based on the position in the image, as defined in (4.6.1).

Figure 20 (*right*) displays the weighted version (4.6.2) incoorporated into Algorithm (2), $HS_{inc,\omega}$. Again, the overall impression is noisier, velocities seem to be amplified uniformly across the image plain and an improvement in performance may not be judged visually. Numerically, the case is not entirely clear: Table 3 shows how for some large-displacement sequences an improvement in multiple measures is noticeable, while for others, there is not. In any case, the change in results is restricted to less than about 5%.

[22] ground truth not available

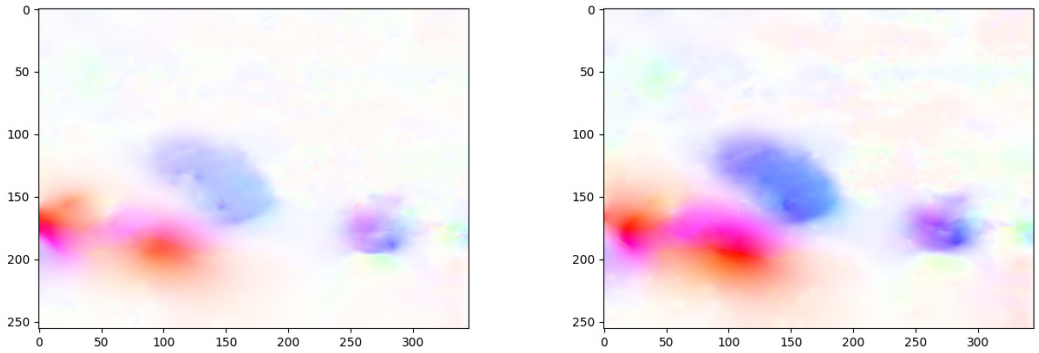

Figure 20: *Hamburg Taxi* Sequence: un-weighted (*left*), weighted (*right*)

	Sponza		*Brickbox*		*Robot*		*Woodbox*	
	HS_{inc}	$HS_{inc,\omega}$	HS_{inc}	$HS_{inc,\omega}$	HS_{inc}	$HS_{inc,\omega}$	HS_{inc}	$HS_{inc,\omega}$
$\bar{EE}(px)$	3.07	2.91	3.76	3.99	4.45	4.11	7.60	8.08
$\bar{AE}(rad)$	0.30	0.29	0.21	0.19	0.27	0.26	0.35	0.42
$\sigma_{EE}(px)$	3.41	2.82	3.81	3.71	5.06	4.51	8.00	7.97
$\sigma_{AE}(rad)$	0.49	0.48	0.36	0.29	0.42	0.44	0.51	0.58

Table 3: Averages and Standard Deviations for Angular and Endpoint Errors

4.7. Warping and its Limitations

The performance of the warping scheme (4.3.1) expectedly improves with higher resolution at which point the input flow field $\vec{h}^k$ is already a very good approximation of the ground truth flow and the resolution is close to that of the original image sequence. Figure 21 depicts warped images[23] at a resolution of 128×128, at the first and the 20^{th} iteration at that stage, as well as the same for resolution 512×512. As suggested before, the area of occlusion at the robot's head poses a major problem. However, most areas of the image resemble the original image (comp. Sequence 47) very well after 20 iterations on 512×512, even though large motion certainly occurs there.

[23] in greyscale

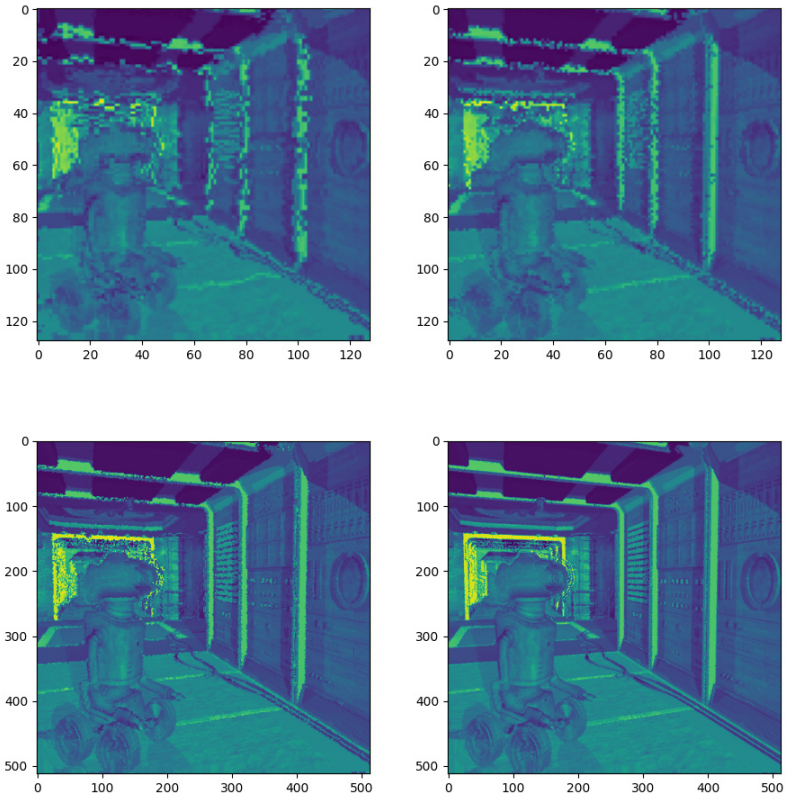

Figure 21: *Robot* Sequence 47: The warped image approximates the first frame of the Sequence increasingly well with more iterations and higher resolution

As previously mentioned, during computation of the optical flow for the *Woodbox* Sequence, a starting resolution of 16×16 is used. The reason for this is that the large motion of the boxes is not sufficiently captured by the basic Horn-Schunck algorithm in the first stage when using 32×32. In fact, Figure 22 shows how the final warping (20^{th} iteration) at resolution 512×512 changes for different initial resolutions.

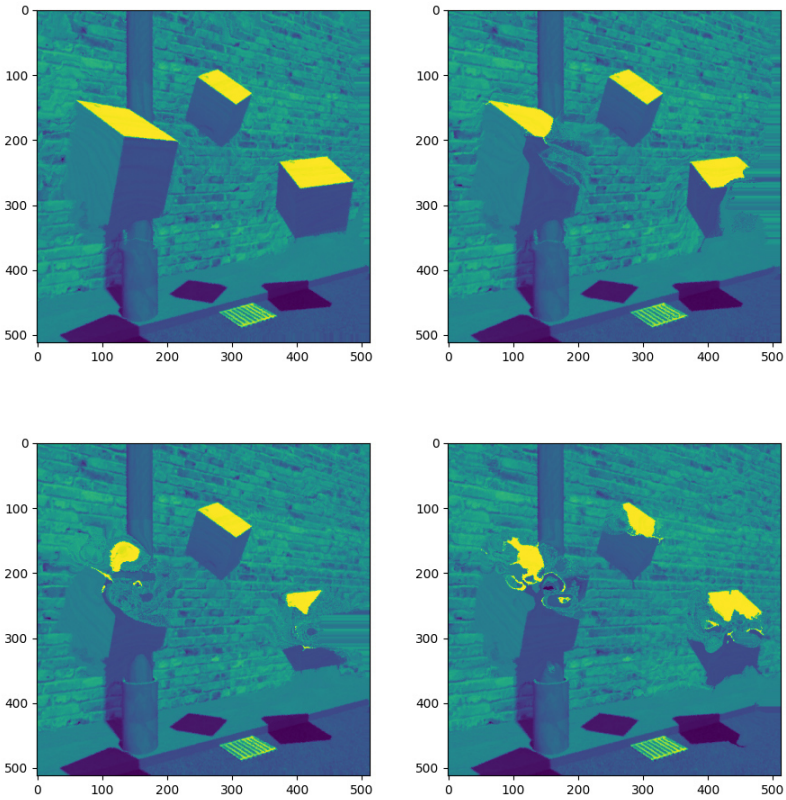

Figure 22: *Woodbox* Sequence 48, final warping: starting resolution from top left to bottom right: 16×16, 32×32, 64×64, 128×128

Throughout the pyramid stages, an insufficient initial flow field can never be corrected, and rather worsens the situation in the erroneous regions. Note that the areas where the warped image is still slightly flawed with starting resolution 16×16, e.g. top and right edge and around the boxes, are those areas where the computed flow appears faulty in Fig. 19, rhs.

Clearly, an object's translation of significantly more than one time its own width may generally not be tackled using HS_{inc}. Figure 23 shows how at a resolution, where all area moving in unison is represented by one pixel, movement distance is more than one pixel to the right. This subsequently leads to sub optimally performing HS_b at the first pyramid stage. At higher resolutions, there are even more pixels between the objects, at even lower resolutions, the whole motion might already be represented by a single pixel.

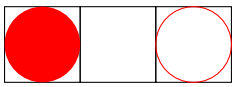

Figure 23: Optimal resolution leaves a pixel in between circle positions in first image (filled) and second image (outline)

Still, an interesting case of Large Displacements is that of a single object translating about once its own width, in front of a steady background, the same phenomenon as already discussed in Chapter 4.5.

A simple motion such as this (Seq. 38), uncovers a possible flaw in the warping technique at first sight: the ground-truth flow or a reasonably good guess for it would not properly warp the second image onto the first. Instead, the warped image appears like in Figure 24, essentially doubling the moving object. Note that the same would not be the case, if the input flow was uniform over the whole image plain, which given the input image sequence is also a possibility due to the lack of information about the background.

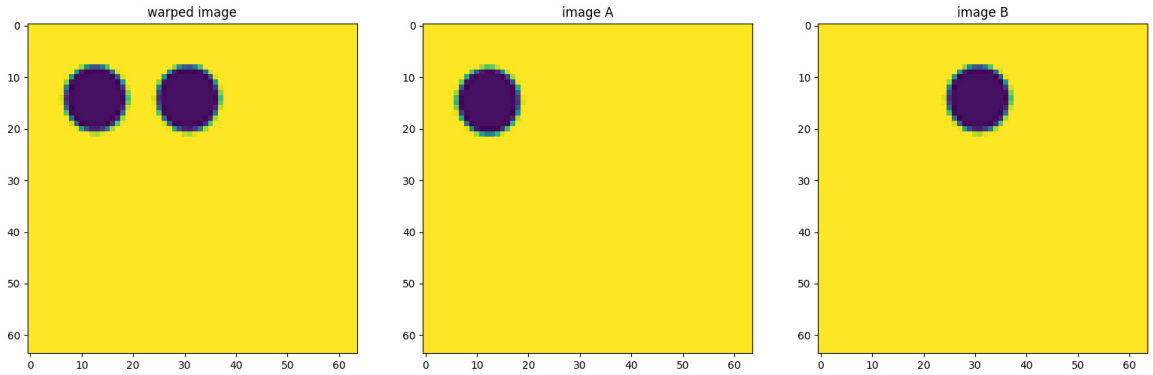

Figure 24: Warping initially fails to work for large motion over steady background for input flow already close to the ground-truth

Interestingly however, with increased iteration count the computed flow at the position of the circle in *image B* slowly drains the faulty circle in the warped image, until the warped image in fact closely resembles *image A*, as depicted in Figure 26. More precisely, the computed flow at each iteration improves the warped image and vice versa, until a satisfying end result is achieved. Therefore, this *doubling* of objects in images is not an issue, and should resolve itself[24].

Still, this kind of motion remains the biggest challenge in the context of large displacements alongside occlusions. As can be seen from the example of Seq. 46, both the basketball and the right person's arms undergo large displacement, they translate a little less than once their own width, in front of a steady background. The computed

[24]for more information, see Chapter 5

flow (Fig. 46, rhs), while a significant improvement over the basic Horn-Schunck approach HS_b, is ultimately not completely satisfactory. Especially the warping (Fig. 46, lhs) of the basketball in front of a background featuring intensity irregularities proves insufficient.

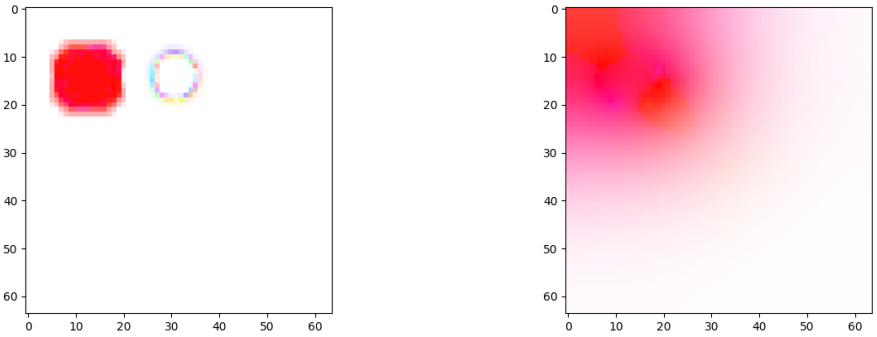

Figure 25: After 1 iteration (*lhs*), after 500 iterations (*rhs*)

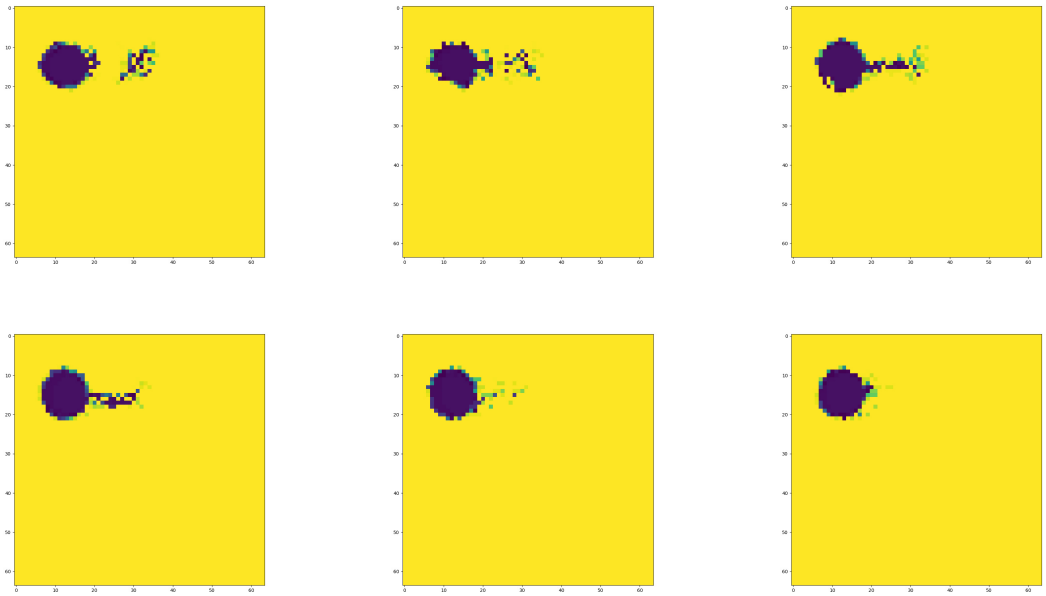

Figure 26: Warped Image at various iteration counts, top left to bottom right: 10, 25, 75, 150, 300, 500

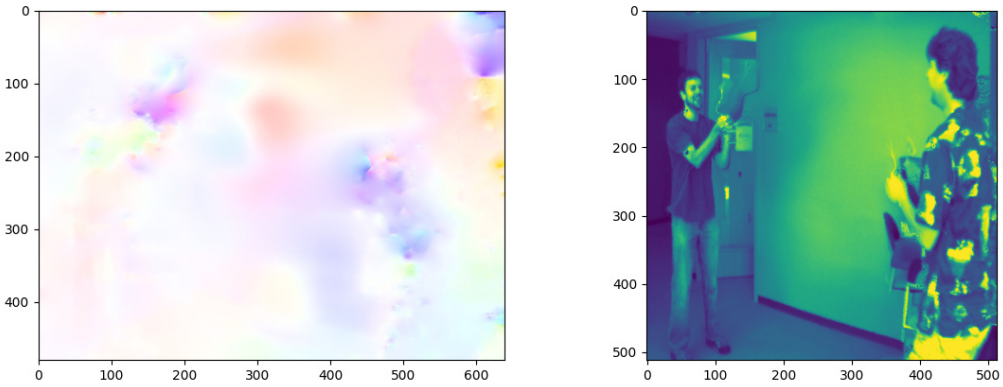

Figure 27: *Basketball* Sequence 46: Computed Flow Field[25](lhs), Warped Image at 20^{th} iteration and highest resolution (rhs)

[25]incorrect flow computation on the right upper edge due to occlusions

4.8. Parameter Influence and Convergence Behaviour

As previously discussed, two sets of parameters need to be chosen at the start of Algorithm 2: regularisation weight α_{hs} and iteration count it_{hs} for the first pyramid stage where the basic Horn-Schunck algorithm is employed, and α_{ihs} and it_{ihs}, for the rest of the pyramid stages. The former pair of parameters is chosen on the same basis as in Chapter 3.5, for most test runs specifically, $\alpha_{hs} = 1$ and $it_{hs} = 25$. In any case, the flow of the resized image sequence in the first stage should be estimated as well as possible and large motion should be captured.

At a fixed iteration count of $it_{ihs} = 20$ per stage, the error measures in dependence of α_{ihs} behave like depicted in Fig. 28 for the *Brickbox* and the *Robot* Sequence: similar convex behaviour as for the basic Horn-Schunck algorithm (Fig. 8, top) is observed.

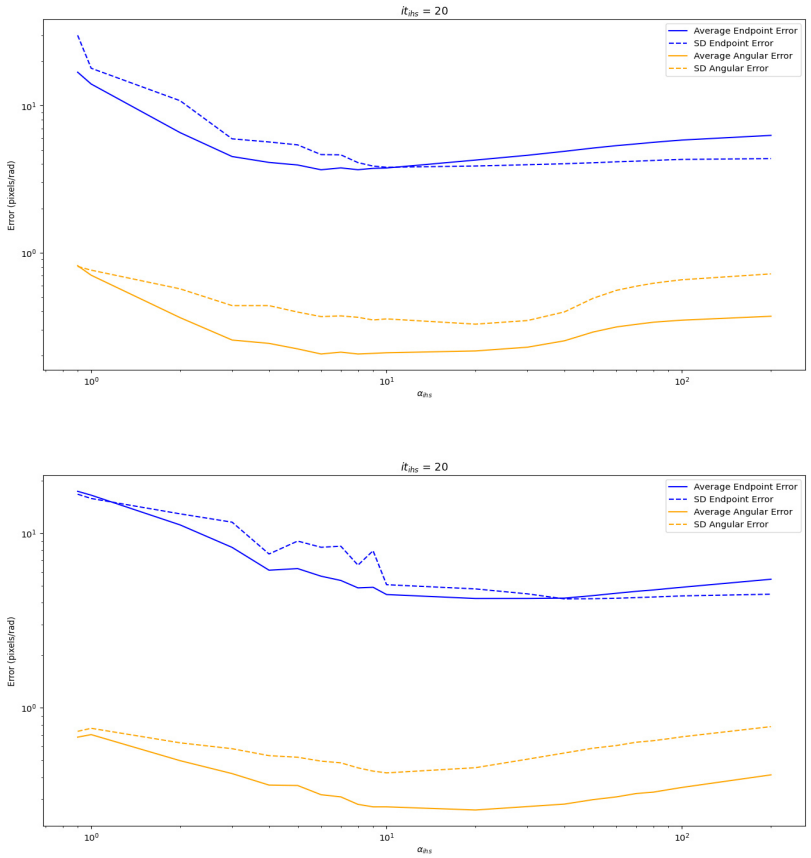

Figure 28: Error Statistics for different α_{ihs} at fixed iteration:
top: *Brickbox* Sequence; *bottom*: *Robot* Sequence

The choice of it_{ihs} in each pyramid stage proves to be more difficult, since while increasing iterations at a given stage should theoretically improve the result, it may also amplify the effects of erroneous warping in some areas of the images. For instance, Fig.

29, *top* depicts how the average endpoint error in fact increases, albeit only slightly, for more iterations due to poor warping at stage 128×128 for the *Robot Sequence*. At the same time, major improvements in the other error metrics occur within the first 10 iterations. Because of that, and in order to avoid negative effects to take over, iteration counts should be kept low at early pyramid stages and may be increased e.g. in the ultimate stage, where warping is already believed to have solved large parts of the image sequence. This is again underlined by Fig. 29, *top*, which shows that error metrics for stage 512×512 uniformly and smoothly improve for increased iteration count, with the exception of standard deviation of the endpoint error, due to faulty extreme values arising as already seen in Fig. 19, *rhs*.

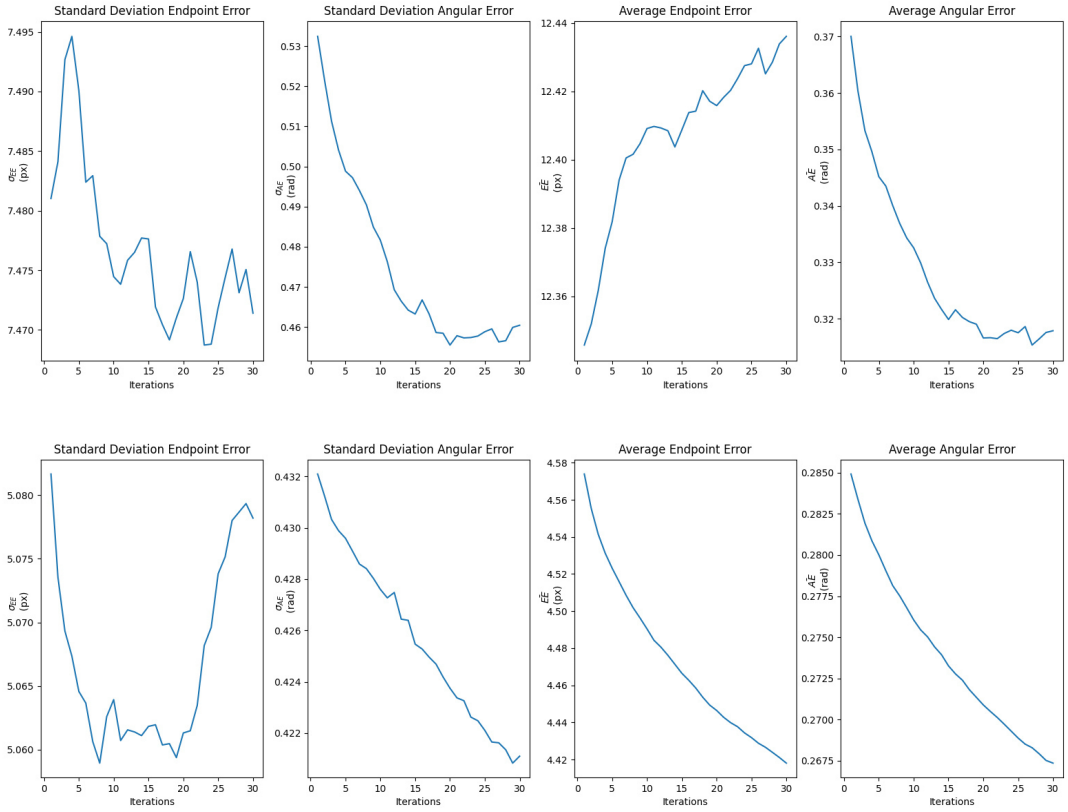

Figure 29: *Robot Sequence*, left-to-right: Standard Deviation EE (px), Standard Deviation AE (rad), Average EE (px), Average AE (rad); top: 128×128; bottom: 512×512

Further examining the change of the computed flow field in each iteration in the same manner as in Fig. 8 (*middle*), initially gives the impression of non-convergent and rather oscillatory behaviour (Fig. 30). However, this stagnation of the relative error (3.5.1) particularly for low resolutions comes about due to flow vectors in occluded regions seemingly arbitrarily changing direction and length each iteration because of

erroneous warping behaviour (Fig. 31). This fact should be expected and can be used to gauge when to halt iteration at a certain pyramid stage, without having to compute error measures, since it indicates the flow field outside the problematic regions remains largely unchanged.

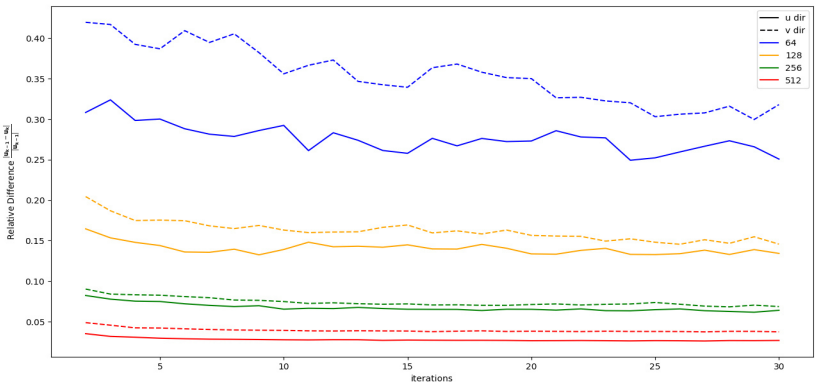

Figure 30: *Robot* Sequence: Convergence Behaviour for different pyramid stages

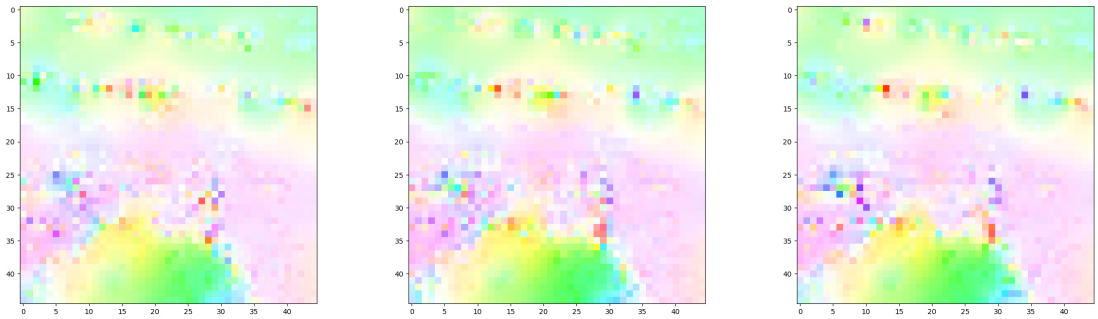

Figure 31: *Robot* Sequence: flow field unpredictably changes direction and magnitude in problematic regions

5. Conclusion

The basic Horn and Schunck algorithm as it was proposed in 1981 does give reasonable approximations for different sequences (e.g. Seq. 42), as long as no major complications are introduced. It is however particularly shown, that Large Displacements are absolutely unsolvable for this basic scheme and as expected even naive introduction into pyramid schemes does not improve the matter significantly. Certainly, usage of Gaussian filters may be employed to improve results, although that is highly dependant on the image sequence at hand and may also lead to adverse effects due to information loss. Otherwise, a fine-tuning of the regularisation weight α is always in-order, even though depending on the images, a trade-off might have to be struck in regards to different error measures. From a computational standpoint the run times of the algorithm are generally relatively short even for image sequences of high resolution, while the implementation remains rather simple.

Employing an iterative incremental scheme with warping in the context of Large Displacements is reasoned with its theoretical equivalence with non-linear approaches. The scheme derived works on the basis of an initial *good enough* guess in order for the warping to work reasonably well. Indeed, significant improvement may be observed for all considered scenes in which Large Displacements occur. Average errors only $\frac{1}{4}$ as large as those for the basic Horn and Schunck algorithm are achieved. The algorithm naturally still struggles when dealing with occlusions. Of course, sequences featuring Large Displacements are also more likely to feature significant areas which are being occluded in one of the two images processed, which leads to its own problems. In all observed sequences, performance is the poorest in these areas. As discussed, the immediate surrounding of an occluded area will also be affected by erroneous warping and therefore actively influences the result[26].

For sequences which do not feature large motion, the case is not entirely clear - and is very dependant on the individual image sequence. While performance for a sequence like Seq. 41 improves drastically, apparently due to improved handling of intensity irregularities in the center of the plant, the same can not be said for an unproblematic sequence like Seq. 42. Images which are dealt with well already are not guaranteed to further improve using this more complicated scheme, where warping may introduce more problems than it solves.

An issue which circumvents straight-forward use of this algorithm is that of parameter choice. Although α_{hs} and α_{ihs} may be chosen in the same vein as with the basic Horn-Schunck algorithm, i.e. there is a minimum where optimal performance is achieved, the iteration count is difficult to get right at times. In some cases, less iterations may approximate the optical flow better around tricky areas as errors may only be enhanced subsequently. Generally, iterating less in early pyramid stages and more in late stages is deemed favourable, although iteration stop is best estimated by observing the conver-

[26] occluded areas themselves are not evaluated

gence behaviour from one iteration to the next. At some point, areas in which warping fails will still continuously change while the rest of the flow field remains largely unchanged. Then, convergence halts and constant change in the relative error from one iteration to the next is observed. Further, choice of the initial resolution at the first pyramid stage is vital and dependent on the size of the motion present in the image sequence. The larger the motion, the less likely it is that it will be resolved by the basic Horn-Schunck algorithm at a given resolution, the antidote for which is reducing the resolution further.

Interestingly, contrast dependence does not seem to be a major problem for this algorithm. When observing a scene where dominant motion is caused by low and high contrast objects moving, motion appears to be captured very well. This may be reasoned with decreasing resolution acting in a similar way to Gaussian Blur, which may aid in decreasing the effect high contrast has in a sequence, since it washes out and creates a more uniform intensity field. Introducing the weight from [20] as done for the basic Horn-Schunck algorithm, again does not influence the result significantly and rather introduces noise. Numerical comparison between using the weight and not doing so, is also not making a clear cut case for either version.

Lastly, the interesting situation of Large Displacement for an object in front of a steady background gives rise to the question as to why warping even works in the first place. Indeed, in the first iteration the warped image does not at all resemble the initial image, but rather appears as a fusion of both images. Naturally, the produced flow field will not be correct and appear like in Fig. 25, *lhs*, reminiscent of how HS_b dealt with such a problem (Chapter 3.7). Repeatedly applying the scheme and updating the warping however slowly solves this issue and the produced flow (Fig. 25, *rhs*) certainly may be correct. Since no background information is present however, the correct flow may as well be a uniform movement to the right across all the image plain. What can be said nevertheless, is that a theoretical ground truth where only the circle itself is moving may never be obtained, rather some kind of washed-out version of it. Therefore, a limitation of warping schemes is observed.

A multitude of techniques lend themselves to being introduced into the proposed algorithm, altering data as well as regularisation term, to further improve its performance. This includes aforementioned TV-approaches [9] or additionally employing image gradient constancy [32] in order to deal with illumination changes, as well as spatio-temporal regularisation, where flow is also assumed to change smoothly over time. The mentioned Gaussian Filtering was not employed for Algorithm 2, but is expected to similarly enhance quality of the result in certain scenes. Avoiding to use only resolutions of powers of 2 and therefore having to change aspect ratio of the input image, should reduce errors at increased implementation complexity. Additionally, implementing (scene specific) a-priori knowledge about e.g. occluded areas in a sequence has its own benefits.

Finally, in order to test the performance of the suggested Algorithm 2 in the toughest environment, as well as compare[27] it to state-of-the art algorithms, the MPI Sintel Database [39] is consulted. As per the official website, the dataset includes problematic features like *Large Motion*, *Specular Reflections*, *Motion Blur*, *Defocus Blur* and *Atmospheric Effects*. Although the algorithm at hand is not designed to deal with most of these complications, an insight in what optical flow computation can currently already deal with is warranted. Figure 32, 33 depict the ground truth flow of sequence *Ambush_3* and *Market_3* from the database as well one of the best approximations, achieved by the previously mentioned *RAFT* [36] and the results obtained via algorithm (2).

As expected, performance of the incremental iterative scheme with warping can not come close to ground-truth for a complex sequence such as *Ambush_3*, with improvement for an easier sequence in *Market_3*. Specifically, the former sequence features plenty of large motion, occlusions and motion blur, which creates an insurmountable task for Algorithm 2. Again, this underlines the fact that machine-learning based methods far outscore a variational approach at this point, even for Large Displacements which the latter was specifically designed to deal with.

Figure 32: *Ambush_3* Sequence [39]: frame 1 (*left-top*), ground-truth (*right-top*), computed flow algorithm (2)(*left-bottom*), computed flow *RAFT*(*right-bottom*)

[27] only visually, as numerical ground-truth is not available for evaluation sets

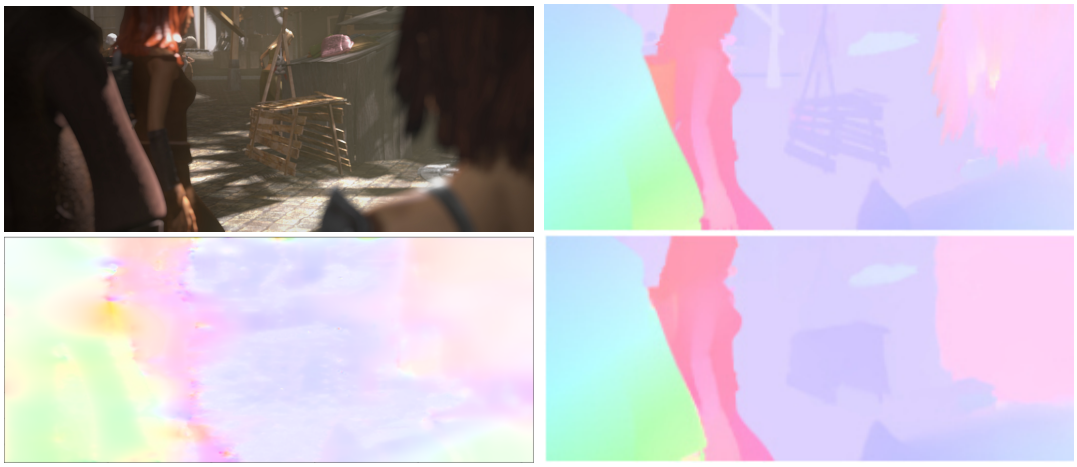

Figure 33: *Market_3* Sequence [39]: frame 1 (*left-top*), ground-truth (*right-top*), computed flow algorithm (2)(*left-bottom*), computed flow $RAFT$(*right-bottom*)

A. Additional Figures

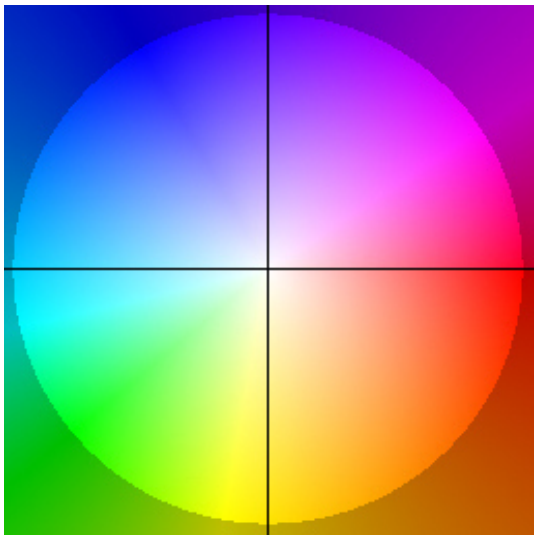

Figure 34: Colorwheel as it is also used in [3]

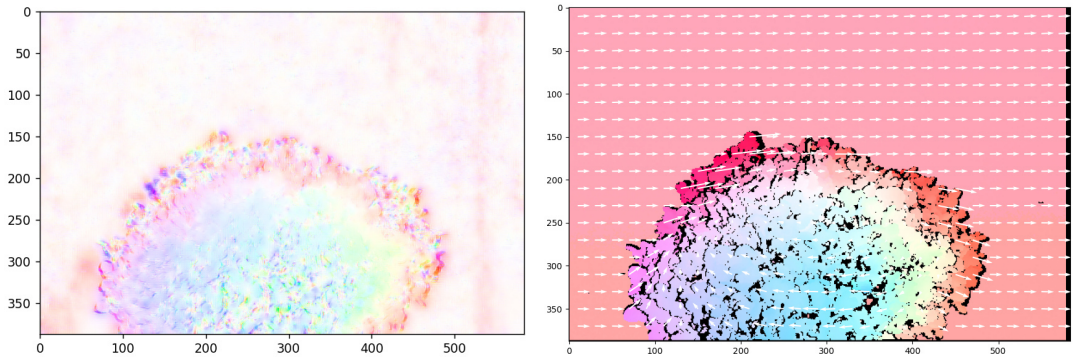

Figure 35: Visual Comparison of Computed Flow (*left*) vs. Ground Truth (*right*): while the rotational part (inside the plant) is modelled well, the algorithm struggles along the edges, where optical flow is changing rapidly

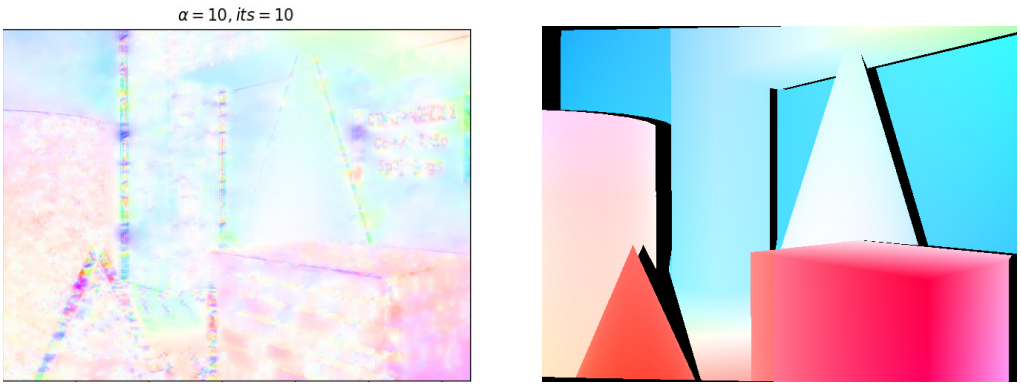

Figure 36: A multi-scale Horn Schunck Algorithm fares only slightly better for large Displacements: Computed Flow (*lhs*), Ground-Truth Flow (*rhs*)

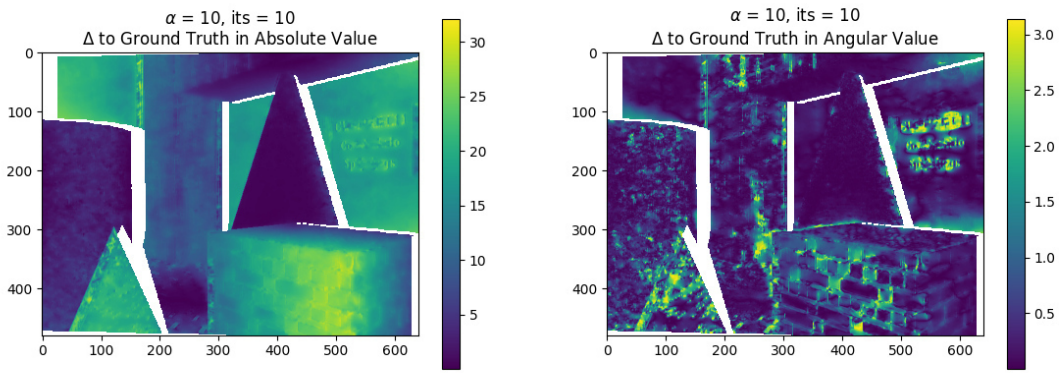

Figure 37: Angular and Endpoint Error for $\alpha = 10$; 10 iterations per stage; 5 stages

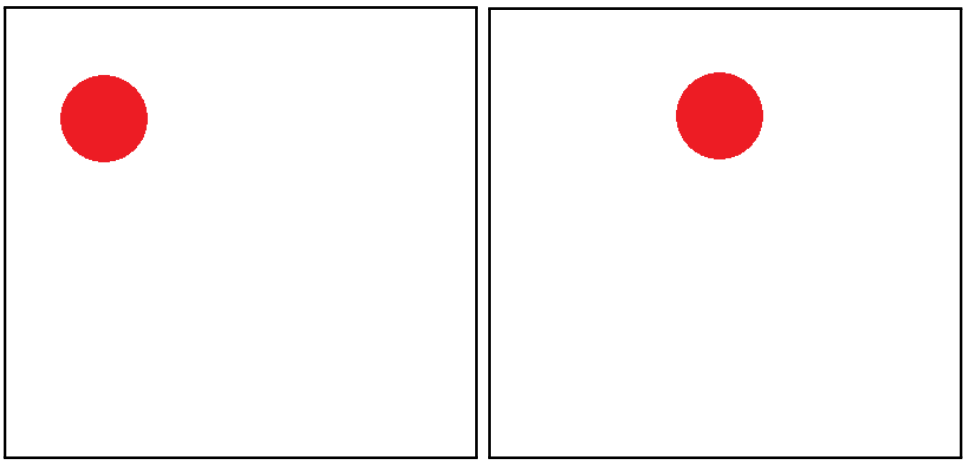

Figure 38: *Circle* Sequence

Figure 39: *Hamburg Taxi* Sequence [30]

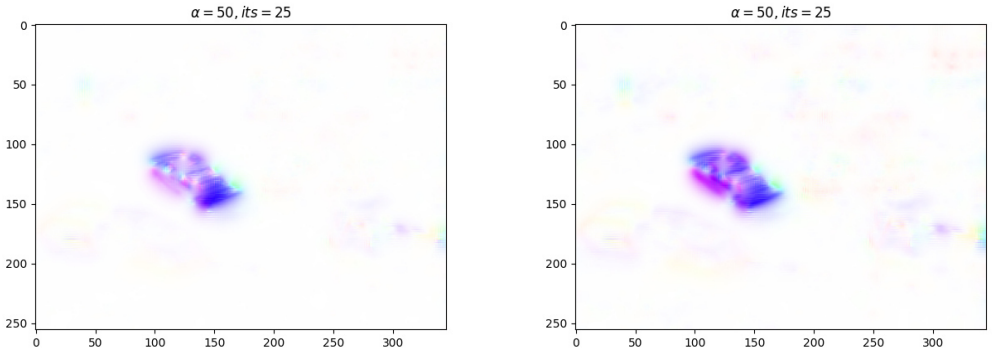

Figure 40: *Taxi* Sequence: unweighted (*left*), weighted (*right*) and $\epsilon = 0.01$

B. Middlebury Sequences [3, 4]

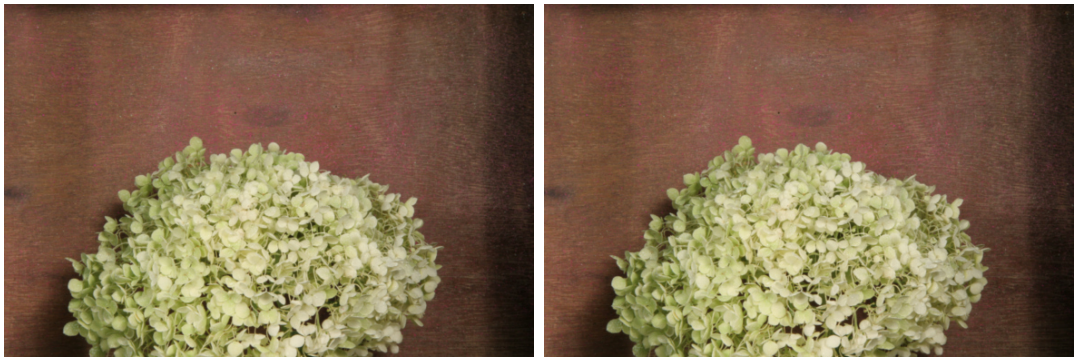

Figure 41: *Hydrangea* Sequence

Figure 42: *Whale* Sequence

Figure 43: *Dimetrodon* Sequence

Figure 44: *Venus* Sequence

Figure 45: *Backyard* Sequence

Figure 46: *Basketball* Sequence

C. UCL Sequences [24, 26]

Figure 47: *Robot* Sequence

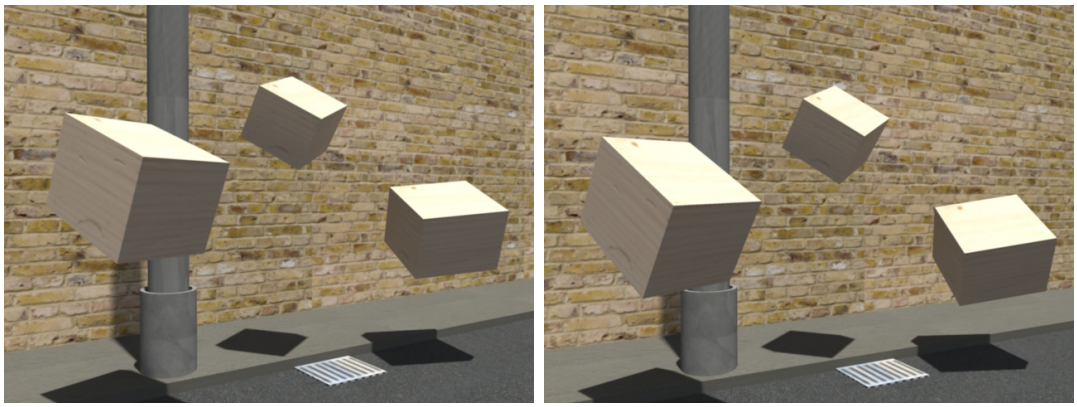

Figure 48: *Woodbox* Sequence

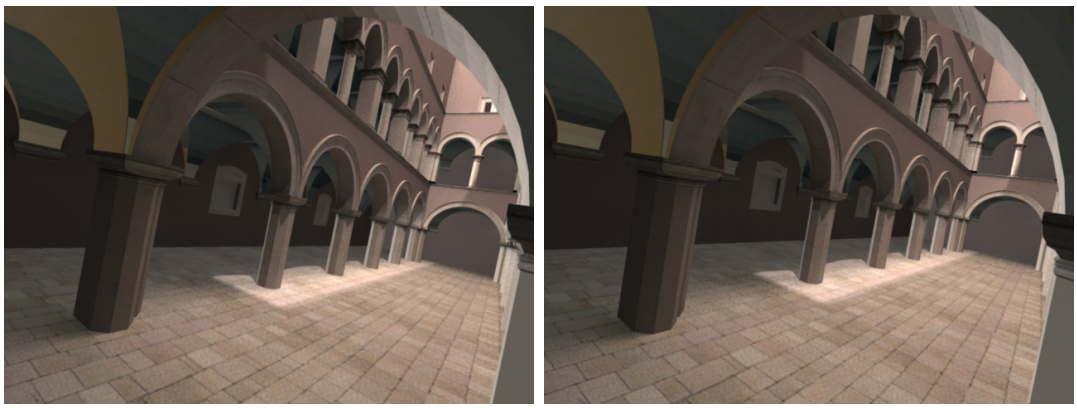

Figure 49: *Sponza* Sequence

Figure 50: *Brickbox* Sequence

D. Abstract

English Version

The topic of Optical Flow Determination is an important one in the field of Computer Vision and deals with the accurate numerical computation of apparent object motion in a sequence of images. In this Master's Thesis, challenges, various different approaches and recent advancements are discussed with the focus initially being laid on the prototypical algorithm of Horn and Schunck dating back to 1981 [19]. It famously makes use of a so-called variational approach, based on minimization of an energy functional including a regularization term, on which many algorithms developed thereafter have built on. The framework of said algorithm is thoroughly analysed and discussed, including its drawbacks compared to related schemes. An effort is made to firstly implement the basic algorithm in *Python* [37] as well as improvements to it, using finite-difference methods.

In the second part, a pyramidal or multi-scale approach (see e.g. [28]) is investigated and implemented, which also includes modification to the original Horn-Schunck energy functional, by searching for a change of the flow field $d\vec{h}$ instead of the whole flow field $\vec{h}$ in each iteration. The result is a so-called *warping* approach which aims to deal with the common problem of *Large Displacements* (LD) (see e.g. [7]) in Optical Flow Computation, where many basic approaches fail. An argument is made for such an approach to be equivalent in nature to LD-Algorithms that make use of non-linear data terms in energy functionals, which was already hinted at by Papenberg et al. in [32]. Results of the implemented algorithms are shown and compared to one another, advantages and short-comings of such a warping scheme are high-lighted.

German Version

Die Thematik der Bestimmung des Optischen Flusses ist von großer Wichtigkeit im Feld der maschinellen Bildverarbeitung und beschäftigt sich mit der akkuraten numerischen Berechnung von scheinbarer Objektbewegung in Bildsequenzen. In dieser Masterarbeit werden die Probleme, verschiedene Ansätze und kürzliche Weiterentwicklungen diskutiert, sowie der Fokus zunächst auf den prototypischen Algorithmus von Horn und Schunck aus dem Jahr 1981 gelegt [19]. Dieser verwendet bekanntermaßen einen sogenannten Variationsansatz, basierend auf Minimierung eines Energie-Funktionals mit zusätzlichem Regularisierungsausdruck, auf welchem verschiedenste darauffolgende Algorithmen aufgebaut sind. Das Horn-Schunck Konzept wird gründlich analysiert und diskutiert, sowie dessen Nachteile gegenüber ähnlichen Ansätzen hervorgestrichen. Hierzu wird der Grund-Algorithmus sowie Verbesserungen dieses in *Python* [37] unter Verwendung der Methode finiter Differenzen implementiert.

Im zweiten Teil der Arbeit wird ein Pyramidenansatz (siehe z.B. [28]) implementiert, welcher eine Modifikation des ursprünglichen Horn-Schunck Funktionals beinhaltet, in dem in jeder Iteration lediglich nach einer Änderung des Flussfeldes $d\vec{h}$ statt dem gesamten optischen Fluss $\vec{h}$ gesucht wird. Das Resultat hiervon ist ein sogenannter *Warping*-Ansatz, welcher in Theorie mit dem bekannten Problem der *Large Displacements* (LD) (siehe z.B. [7]) besser umgehen kann. Es wird argumentiert, dass ein solcher Ansatz mit anderen LD-Algorithmen gleichgesetzt werden kann, die nicht-lineare Datenterme im Energie-Funktional verwenden, wie bereits in [32] von Papenberg et al. angedeutet. Resultate der implementierten Algorithmen werden grafisch dargestellt und miteinander verglichen, Vor- und Nachteile eines solchen *Warping*-Ansatzes hervorgestrichen.

Bibliography

[1] Luis Alvarez, Joachim Weickert, and Javier Sánchez. "Reliable estimation of dense optical flow fields with large displacements". In: *International Journal of Computer Vision* 39.1 (2000), pp. 41–56.

[2] Padmanabhan Anandan. "A computational framework and an algorithm for the measurement of visual motion". In: *International Journal of Computer Vision* 2.3 (1989), pp. 283–310.

[3] Simon Baker et al. "A database and evaluation methodology for optical flow". In: *International journal of computer vision* 92.1 (2011), pp. 1–31.

[4] Simon Baker et al. *vision.middlebury.edu*. Middlebury College. 2011. URL: `https://vision.middlebury.edu/flow/`.

[5] Marcelo Bertalmio et al. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 417–424.

[6] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] Thomas Brox, Christoph Bregler, and Jitendra Malik. "Large displacement optical flow". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 41–48.

[8] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods". In: *International journal of computer vision* 61.3 (2005), pp. 211–231.

[9] Martin Burger, Hendrik Dirks, and Lena Frerking. "On optical flow models for variational motion estimation". In: *Variational Methods In Imaging and Geometric Control, M. Bergounioux, G. Peyré, C. Schnörr, J.-P. Caillau, and T. Haberkorn, eds* 18 (2017), pp. 225–251.

[10] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[11] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: `https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf`.

[12] R Courant and D Hilbert. "Methods of mathematical physics". In: Volume 1 (1954), p. 184.

[13] Alexey Dosovitskiy et al. "Flownet: Learning optical flow with convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[14] Claude L Fennema and William B Thompson. "Velocity determination in scenes containing several moving objects". In: *Computer graphics and image processing* 9.4 (1979), pp. 301–315.

[15] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. “Optical flow modeling and computation: A survey”. In: *Computer Vision and Image Understanding* 134 (2015), pp. 1–21.

[16] C-S Fuh and Petros Maragos. “Region-based optical flow estimation”. In: *1989 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society. 1989, pp. 130–131.

[17] Stephen Gould, Tianshi Gao, and Daphne Koller. “Region-based segmentation and object detection”. In: *Advances in neural information processing systems* 22 (2009).

[18] Chris Harris, Mike Stephens, et al. “A combined corner and edge detector”. In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[19] Berthold KP Horn and Brian G Schunck. “Determining optical flow”. In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[20] José A Iglesias and Clemens Kirisits. “Convective regularization for optical flow”. In: *Variational Methods in Imaging and Geometric Control* (2015), pp. 184–201.

[21] Louis Le Tarnec et al. “A Proof of Convergence of the Horn–Schunck Optical Flow Algorithm in Arbitrary Dimension”. In: *SIAM journal on imaging sciences* 7.1 (2014), pp. 277–293.

[22] JO Limb and JA Murphy. “Estimating the velocity of moving images in television signals”. In: *Computer graphics and image processing* 4.4 (1975), pp. 311–327.

[23] Tony Lindeberg. “Feature detection with automatic scale selection”. In: *International journal of computer vision* 30.2 (1998), pp. 79–116.

[24] University College London. “UCL Ground Truth Optical Flow Dataset v1.2”. In: (2022). http://visual.cs.ucl.ac.uk/pubs/flowConfidence/supp/.

[25] Bruce D Lucas, Takeo Kanade, et al. “An iterative image registration technique with an application to stereo vision”. In: Vancouver. 1981.

[26] Oisin Mac Aodha, Gabriel J Brostow, and Marc Pollefeys. “Segmenting video into classes of algorithm-suitability”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1054–1061.

[27] Michael Maire et al. “Using contours to detect and localize junctions in natural images”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[28] Enric Meinhardt-Llopis and Javier Sánchez. “Horn-schunck optical flow with a multi-scale strategy”. In: *Image Processing on line* (2012).

[29] Hans-Hellmut Nagel. “On the estimation of optical flow: Relations between different approaches and some new results”. In: *Artificial intelligence* 33.3 (1987), pp. 299–324.

[30] Hans-Hellmut Nagel. “Universität Karlsruhe, Fakultät für Informatik”. In: *Image Sequence Server: Taxi* (). http://i21www.ira.uka.de/.

[31] Shree Nayar. *Lucas-Kanade Method — Optical Flow*. First Principles of Computer Vision. 2021. URL: https://www.youtube.com/watch?v=6wMoHgpVUn8&t.

[32] Nils Papenberg et al. "Highly accurate optic flow computation with theoretically justified warping". In: *International Journal of Computer Vision* 67.2 (2006), pp. 141–158.

[33] Krishnan Ramnath et al. "Increasing the density of active appearance models". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[34] Daniel Scharstein and Richard Szeliski. "High-accuracy stereo depth maps using structured light". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 1. IEEE. 2003, pp. I–I.

[35] Christoph Schnörr. "Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class". In: *International Journal of Computer Vision* 6.1 (1991), pp. 25–38.

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

[37] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[38] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[39] Andreas Wedel et al. "An improved algorithm for tv-l 1 optical flow". In: *Statistical and geometrical approaches to visual motion analysis*. Springer, 2009, pp. 23–45.

[40] Mark Wibrow. *TikZ Aperture Problem*. StackExchange. 2014. URL: https://tex.stackexchange.com/questions/198491/tikz-aperture-problem.

[41] Josh Wills, Sameer Agarwal, and Serge Belongie. "A feature-based approach for dense segmentation and estimation of large disparity motion". In: *International Journal of Computer Vision* 68.2 (2006), pp. 125–143.

[42] Eberhard Zeidler. "Nonlinear Functional Analysis and Its Applications II/A: Linear Monotone Operators". In: Springer Science & Business Media, 2013, p. 69.