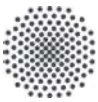


University of Stuttgart

Institute for Visualization and Interactive Systems,
Dept. Computer Vision

Universitätsstraße 38
70569 Stuttgart

Bachelor Thesis

Diffusion-Based Refinement of Optical Flow

Victor Thomas Oei

Study progam: Simulation Technology

Examiner: Prof. Dr.-Ing. Andrés Bruhn

Advisor: Lukas Mehl, M.Sc.

start date 15.01.2021

end date: 15.07.2021

UNIVERSITY OF STUTTGART

Abstract

Faculty 5 - Computer Science, Electrical Engineering and Information Technology
Institute for Visualization and Interactive Systems

Bachelor of Science

Diffusion-Based Refinement of Optical Flow

by Victor Thomas OEI

Estimating the optical flow of an image sequence is a major challenge in computer vision. Optical flow is a vector field that provides information about the apparent motion, for example of objects, edges or surfaces, within a visual scene. The recently introduced Recurrent All-Pairs Field Transforms (RAFT) by Teed and Deng [TD20] achieved significant improvements over prior methods on popular benchmarks. However, this method reduces each image dimension to an eighth at the beginning of the calculations and scales the resulting flow back to the original size afterwards. This upscaling leads to a visible structural error. We investigate this error and present various approaches to reduce it using adapted diffusion methods. The peculiarity of our approach is that we consider only the resulting optical flow and not the underlying image data. In the process, a framework was developed to examine and diffuse vector-valued images such as optical flow. In addition, we present advanced methods for blocking artifact detection. All methods are tested and evaluated on a common sample data set. The regular structure of the error of the RAFT method could be exploited to slightly reduce the average error of the method.

Contents

* Note: Chapters 1 and 2 correspond to the Propaedeuticum, which is part of the study program "Simulation Technology" and can be completed alongside the Bachelor Thesis.

To my brothers Yannick and Marius. You're the best.

Introduction

The determination of optical flow is a key method for extracting motion information in computer vision. It is one of the major problems in machine vision. In general, the optical flow represents the vector field of the displacement of image points between two successive frames of an image sequence. The first seminal work in this area goes back to Horn and Schunck [HS81].

Optical flow is used in many applications. These include object tracking [Shi+05; Mae+96], for example via motion estimation, object detection in moving environments such as environment detection in (autonomous) driving [GLU12; MG15], image stabilization in cameras [Cha+02], video compression in the field of motion compensation [SLZ98], and many more.

Current methods for determining optical flow are oftentimes learning-based methods [Dos+15; RB17; Sun+18; TD20]. These are based on the optimization of numerous parameters and often create very large correlation volumes, which map all possible correlations between two images to determine where a certain part of an image moved to in the next frame. In order to handle such large amount of data efficiently, most of these methods are simplified. One of the more recent approaches in this area is called Recurrent All-Pairs Field Transforms (RAFT) and was presented by Teed and Deng [TD20]. It currently delivers superior results on popular benchmarks such as MPI Sintel [But+12] and KITTI 2015 [Gei+13]. A main simplification made in RAFT is to reduce the amount of data to be processed by acting on only one eighth of the original image size. To obtain the optical flow in the original resolution at the end, the resulting optical flow is upsampled to the original size. This causes the emergence of blocking artifacts in the flow visualizations, which are accompanied by an increase in the average error.

The goal of this work is therefore to develop a diffusion-based refinement step that improves the RAFT method. For this purpose, a general framework for the application of diffusion processes is implemented. It is especially adapted to vector-valued images and supports a variety of diffusion methods including linear and nonlinear ones. Using this framework, a specific diffusion process is developed that reduces the blocking artifacts described above while simultaneously improving the overall accuracy of RAFT. Note that a premise of our approach is that we consider only the resulting optical flow and not the actual underlying image data itself. This is a novel approach to improving optical flow using diffusion.

The remainder of this thesis is divided into two main parts. The first part represents the Propaedeuticum and consists of Chapter 1 and Chapter 2, where related works are presented, providing a theoretical basis of diffusion and optical flow that is used in the methods presented afterwards. The second part represents the Bachelor Thesis and consists of the remaining chapters. The diffusion implementations are presented in Chapter 3. Subsequently, the methods we developed for refinement are explained in Chapter 4 and we present our main results and evaluate these methods in Chapter 5. Chapter 6 finally summarizes the most important findings and gives an outlook on possible improvements.

1 Diffusion

Diffusion finds application in many areas of image processing. Usually it is used for smoothing two-dimensional images. In this thesis, diffusion will be used to enhance optical flow, which is a vector-valued image. While this is an area in which diffusion algorithms have rarely been used before, it will prove to offer some advantages. In this chapter, we will discuss the history and some basic ideas of conventional diffusion in the field of image processing. We outline various crucial works in the development of diffusion-based algorithms including several different approaches.

In a general sense, diffusion describes the process of equalizing concentration differences. Since this concept originates in physics, we will use the general physical concept as starting point. Based on this, we will briefly outline linear diffusion filters. However, we will focus more on nonlinear diffusion methods, as they are more sophisticated and allow greater influence on the diffusion process. This includes anisotropic diffusion, which is basically a nonlinear, space-variant transformation of an image. It typically aims to reduce noise while preserving crucial parts of the image such as edges and will play a major role in this work. This chapter is structurally based on Weickert's "Theoretical Foundations of Anisotropic Diffusion in Image Processing" [Wei96], where the concepts presented here are discussed in great detail.

1.1 Physical Concept

As an entry point to diffusion, the physical description of a diffusion process is suitable since it is quite easy to conceive. In the following, it will be analogized to diffusion filters in image processing. For example, the basic physical idea of concentration equalization in liquids can be transferred to the smoothing of images. More precisely, first consider Fick's first law in two dimensions

$$J = -D\nabla u, \tag{1.1}$$

where u is a concentration, J is a diffusion flux and D a diffusion coefficient, a positive definite symmetric matrix. Also, the nabla operator ∇ represents the differential operator of the gradient throughout the thesis. More formally, it is a vector whose components are the partial derivative operators $\frac{\partial}{\partial x}$, so in our case we mainly get $\nabla u = \left(u_x, u_y\right)^\top$ with the shorthand notation $u_x = \frac{\partial u}{\partial x}$.

In image processing, the concentration u may be equated with the gray value of a pixel. Note that u depends on a location x and a time parameter t. Formally, we denote $\Omega \to \mathbb{R}^2$ as a subset of the plane, where $\Omega := (0, a_x) \times \left(0, a_y\right)$ is the rectangular image domain. So the image u is rather a family of rectangular gray scale images and can be described as $u(x,t) : \Omega \to \mathbb{R}$. Going back to Equation (1.1) and additionally considering mass conservation, which states that the overall physical mass does not change due to diffusion, we can apply this idea to imaging and obtain the continuity equation

$$\operatorname{div} J + \partial_t u = 0. \tag{1.2}$$

Using this, the general diffusion equation is obtained, which is given by

$$\partial_t u = \operatorname{div}(D \cdot \nabla u). \tag{1.3}$$

The whole system is a partial differential equation (PDE) for which we require an initial condition. In image processing, the input image f is used as the initial condition with

$$u(x,0) = f(x). \tag{1.4}$$

In Equation (1.3), the choice of the diffusion tensor D is decisive. If it is spatially constant in the image, we speak of homogeneous diffusion. If D is spatially invariant, it is called inhomogeneous diffusion. Furthermore, one can distinguish whether D depends on the structure of the evolving image. This is not the case in the following section, where we get linear diffusion. Only nonlinear methods allow a functional dependence $D(\nabla u)$.

1.2 Linear Diffusion

One of the simplest methods for smoothing images is linear diffusion. This method is similar to the convolution with a Gaussian. As already mentioned, the diffusion tensor does not depend on the image in linear methods and can be replaced for example by the unit matrix. Substituting this into Equation (1.3) yields the linear diffusion equation

$$\partial_t u = \Delta u. \tag{1.5}$$

The entire diffusion process can also be understood on the basis of the scale-space. This term refers to the concept of embedding the original image into a family of subsequently simpler, more global representations of it. The parameter t is also called scale parameter. It can be understood in such a way that image structures, which are smaller than a certain spatial size proportional to t, are largely smoothed in the scale-space plane of scale t.

One of the major drawbacks of linear diffusion is that important features of the image are smoothed, such as edges and corners. Since these are crucial for object recognition, they should not be lost in the method developed here. This can be seen clearly in Figure 1.2, which shows different diffusion methods applied to an example image, with Figure 1.2c corresponding to linear diffusion.

1.3 Nonlinear Diffusion

More complex methods for diffusion are nonlinear diffusion filters. These are based on the idea of making the diffusion process itself depending on the image. The process is thus a combination between the original image and a filter that depends locally on the image content. In this chapter, the first formulations of nonlinear filters are presented to provide a basic understanding. Subsequently, more sophisticated methods are considered, which offer a greater influence on properties such as direction and strength of diffusion, especially also depending on local structures.

Perona-Malik Model

This section would be incomplete without giving credit to Perona and Malik for their first PDE-based formulation of a nonlinear diffusion method [PM90; PSM94]. The basic idea of their method is to detect important features of an image and avoid smoothing those, unlike linear diffusion. To locate features such as edges, their model uses the image gradient. This

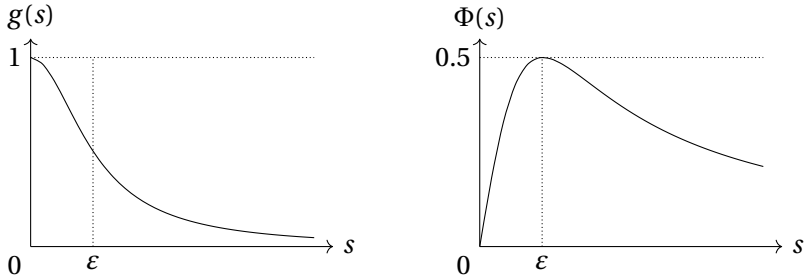


FIGURE 1.1: Comparison of the diffusivity $g\left(s^2\right) = \frac{1}{1+s^2/\varepsilon^2}$ (left) and its corresponding flux function $\Phi(s) = sg\left(s^2\right)$ (right). The parameter ε is located exactly where the flux is largest, thereby separating areas of low contrast from those of high contrast.

leads to the Perona-Malik equation

$$\partial_t u = \operatorname{div}\left(g\left(|\nabla u|^2\right)\nabla u\right), \tag{1.6}$$

where g is a scalar function called *diffusivity*. It is a decreasing function, that is large for small values of the squared magnitude of the image gradient $|\nabla u|^2$. Perona and Malik propose a diffusivity of the form

$$g\left(s^2\right) = \frac{1}{1+s^2/\varepsilon^2} \tag{1.7}$$

with a contrast parameter $\varepsilon > 0$. An example using this diffusivity is shown in Figure 1.2d. Other possible choices of the diffusivity leading to edge-preserving regularization are presented by Blanc-Feraud et al. [BF+95].

This method yields good results especially in terms of edge-enhancement. To understand this, consider the *flux function* $\Phi(s) := sg\left(s^2\right)$ in the one-dimensional case. This flux function reflects the flux in the image in the sense that it takes large values for edges and is small elsewhere. It can be shown that the diffusion equation in one dimension can be rewritten to

$$\partial_t u = \Phi'(u_x)\, u_{xx}. \tag{1.8}$$

The flux function is monotonically increasing for values smaller than ε and monotonically decreasing for larger values. This can be seen in Figure 1.1 using the Perona-Malik diffusivity. From Equation (1.8) we can deduce that the diffusion process is a forward PDE for monotonously increasing flux, so for gradient values with $|u_x| \leq \varepsilon$, which leads to smoothing of the image. For a negative derivative of the flux, on the other hand, we get backward diffusion, which increases the contrast of the image and we get edge-enhancement. Note that a problem with backward diffusion is that it can generate numerically unstable solutions and may even have multiple solutions, which is also known as *ill-posedness*. Overall, we can summarize that the parameter ε affects the contrast of the image. Thus, a correct choice not only smooths unwanted noise but also sharpens edges.

Unfortunately, this model has some shortcomings, which are listed in detail by Weickert [Wei96]. Among other things, this includes the problem that the model delivers very poor results when the image is noisy. The image gradient introduced by the model as a feature detector fails in this case because it oscillates very strongly and noisy edges remain. Another problem of the model results from an investigation of Höllig [Hö83]. He constructed a forward-backward diffusion process similar to the one in the Perona-Malik model, which can have infinitely many solutions. This suggested a certain ill-posedness of the model, which was later confirmed by the absence of uniqueness and stability. Kawohl and Katev [KK98] proved that the Perona-Malik process has no global (weak) C^1 solutions for initial data

involving backward diffusion.

There are numerous papers proposing improvements to Perona and Malik's model. For example, Catté et al. [Cat+92] introduce a variation based only on the scale parameter. In doing so, they change the diffusivity to $g(|\mathrm{d}G_\sigma * u|)$. This leads to the modified diffusion equation

$$\partial_t u = \operatorname{div}\left(g(|\mathrm{d}G_\sigma * u|)\nabla u\right), \tag{1.9}$$

where $*$ denotes a convolution and $\mathrm{d}G_\sigma$ is the derivative of a Gaussian-like

$$G_\sigma(x) = C\sigma^{-\frac{1}{2}} \exp\left(-\frac{|x|^2}{4\sigma}\right). \tag{1.10}$$

Note that this function G is the fundamental solution of the heat equation. This means that the diffusivity g in Equation (1.9) depends on a term that is simply the gradient of the solution at time σ of the heat equation with $u(x,t)$ as initial condition. Overall, this small change in the method should reduce susceptibility to problems caused by noise in the image without smoothing it beforehand. In addition, the instabilities of the Perona-Malik method that arise when the flux function Φ is not nondecreasing are eliminated, since the modified method has been proven by Catté et al. [Cat+92] to have a unique smooth solution.

Wei [Wei99a] proposed a generalization of the model by adding a real-valued bounded edge-enhancement functional to the equation of Perona and Malik. This edge-enhancement feature is useful when handling blurred edges, low contrast as well as low resolution images. A more recent improvement of this approach is presented by Guo et al. [Guo+12]. They adapt the method by combining it with the heat equation, which corresponds to Equation (1.5), and introducing an edge indicator to alternate between Perona-Malik diffusion and Gaussian smoothing.

It is worth mentioning that Perona-Malik diffusion is an isotropic model (although Perona and Malik originally presented it as an anisotropic model). Isotropic methods use the scalar value of the diffusivity g instead of the tensor D in Equation (1.3). Thus the flux $J = -g\nabla u$ of the filter is always parallel to the image gradient. Since we want to have greater influence on the flux and its direction, we consider anisotropic diffusion with a diffusion tensor D in the following section.

Anisotropic Diffusion

As already mentioned, anisotropic filters play a particularly important role in this work. One main advantage is that they allow influencing the direction of the diffusion process and therefore also for it to be directed towards the orientation of features of interest. Generally speaking, anisotropic diffusion is a space-variant transformation of an image and is similar to the process of creating a scale-space, with the difference that each resulting image is a combination of the original image and a filter that depends locally on the content of the original image.

Crucial to this method is the choice of the diffusion tensor D. Since it is positive definite and symmetric, it can be easily split up into eigenvectors and eigenvalues, which correspond to the local directions and strengths of diffusion. A useful choice is to make them dependent on the local image structure. With eigenvectors v_1 and v_2 and eigenvalues λ_1 and λ_2, the decomposition of the diffusion tensor is given as

$$D(u) := (v_1 \mid v_2)\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} v_1^\top \\ v_2^\top \end{pmatrix}. \tag{1.11}$$

One possible choice of D is a regularization of the Perona-Malik process and is presented by Weickert [Wei96] as *edge-enhancing anisotropic diffusion*. With this method, the

eigenvectors are chosen based on the image gradient smoothed by a Gaussian K_σ with

$$K_\sigma(x) := \frac{1}{\left(2\pi\sigma^2\right)} \cdot \exp\left(-\frac{|x|^2}{2\sigma^2}\right), \tag{1.12}$$

where σ is the kernel size. This leads to the smoothed image gradient

$$\nabla u_\sigma := \nabla K_\sigma * u, \tag{1.13}$$

where $*$ denotes a convolution. The eigenvectors are then chosen as

$$v_1 \| \nabla u_\sigma \quad \text{and} \quad v_2 \perp \nabla u_\sigma, \tag{1.14}$$

where $\|$ stands for parallelism and $\perp$ for orthogonality. The eigenvalues on the other hand are chosen depending on the diffusivity g. This way, smoothing along edges is preferred over smoothing across them. A possible choice is suggested by Weickert as

$$\lambda_1 := g\left(|\nabla u_\sigma|^2\right) \quad \text{and} \quad \lambda_2 := 1, \tag{1.15}$$

where λ_2 leads to diffusion along edges. Figure 1.2e shows an example of this method. Note that this is only one possible choice of eigenvalues. We will evaluate several other options later.

There are also more sophisticated methods to describe the local structure. For example, it can sometimes be of interest to maintain flow-like structures in the image. However, with the previous choice of the diffusion tensor, too much smoothing of the image gradient can cause neighboring gradients with the same direction but opposite orientation to cancel, and the flow field loses important information as a result. To prevent this, one can consider the so-called *structure tensor*. It is a symmetric, positive semidefinite matrix J_ρ and is calculated on the basis of the image gradient as

$$J_\rho(\nabla u_\sigma) := K_\rho * \left(\nabla u_\sigma \nabla u_\sigma^\top\right), \tag{1.16}$$

where K_ρ again denotes a Gaussian, only this time with kernel-size ρ instead of σ in order to distinguish the two. Using this structure tensor, Weickert now proposes *coherence-enhancing anisotropic diffusion* [Wei99b; Wei95], which smoothes mainly along coherent structures. See Figure 1.2f for an example of coherence-enhancement. For this purpose, the eigenvectors of the diffusion tensor are chosen equal to those of the structure tensor. For the eigenvalues, we can additionally introduce a parameter $\gamma \in (0,1)$ and obtain

$$\lambda_1 := \gamma, \tag{1.17}$$

$$\lambda_2 := \begin{cases} \gamma + (1-\gamma)\exp\left(\frac{-c}{(\mu_1-\mu_2)^2}\right) & \text{if } \mu_1 \neq \mu_2 \\ \gamma & \text{else.} \end{cases} \tag{1.18}$$

Here, $\mu_1 \geq \mu_2$ are the eigenvalues of the structure tensor J_ρ and c is a threshold parameter. The exponential function and γ guarantee the smoothness of the diffusion tensor and at the same time ensure that there is always some amount of linear diffusion. This is used to prove well-posedness.

Other Methods

In addition to the methods of nonlinear diffusion presented here, there is a variety of other methods [Wei96; Sze10; NFD97; Bla+98; YA02] and we are unable to present all of them here. Ultimately, however, most of them come down to choosing the eigenvectors and eigenvalues

of the diffusion tensor D depending on the specific application. For this reason, we are able to have a consistent implementation of the diffusion process, which we will present later. A large part of this work consists of cleverly choosing the diffusion tensor as well.

1.4 Relation of Diffusion and Variational Methods

We have previously mentioned the method presented by Horn and Schunck [HS81]. The Horn-Schunck method is a widely used technique for estimating the optical flow of an image sequence. It is generally classed as a differential method, where one can make a further classification [BWS05] into local methods such as that of Lukas and Kanade [LK81] and global methods such as that of Horn and Schunck.

We do not want to present a lengthy explanation of such variational methods. A more detailed description of variational methods for the calculation of optical flow is given by Weickert et al. [Wei+03]. Instead, we want to highlight an interesting connection between the Horn-Schunck method and diffusion.

Variational Methods Simply put, variational methods calculate minimizers of functionals. A functional in our case is a mapping of functions to a scalar value. If we then consider the given image sequence as a continuous function f and compute the optical flow u from it, we can compute this optical flow with the help of an energy functional $E(u)$. It provides a scalar value for a given optical flow, which is large for large deviations of the flow from given assumptions and vice versa. Horn and Schunck [HS81] make two basic assumptions, which we divide into a data term D and a smoothness term S. This results in the energy functional

$$E(u) = \int_\Omega D + \alpha S \, \mathrm{d}x \, \mathrm{d}y, \tag{1.19}$$

where Ω is the image domain and α is a regularization constant, with larger values of α leading to a smoother flow.

Data Term First, consider the data term D. It is based on constancy assumptions about the image sequence. This includes the brightness constancy assumption, which states that the value of a pixel in one image corresponds to the value of the pixel at its new shifted position in the next image. The gray value of objects in the image would therefore not change over time (or only slowly). So if the optical flow is $u = (u_1, u_2)^\top$ and $\big(x(t), y(t)\big)^\top$ denotes some movement in the image at time t, we obtain the optic flow constraint [Wei+03]

$$f_x u_1 + f_y u_2 + f_t = 0, \tag{1.20}$$

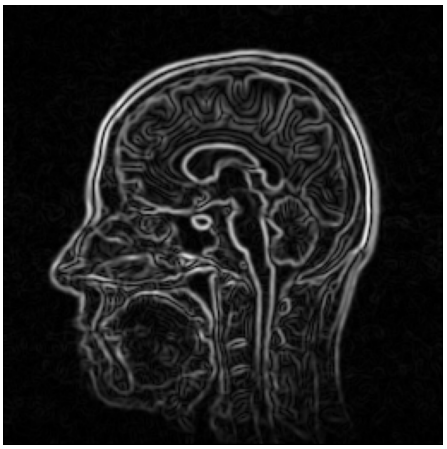
where the index in f_x denotes a partial derivative $f_x := \partial_x f$. This assumption usually holds only for small movements and small changes in image values.

Smoothness Term and Relation to Diffusion More important in the context of this comparison to diffusion is the smoothness term S. It acts as a regularizer of the image, with Horn and Schunck using a simple homogeneous regularizer of the form

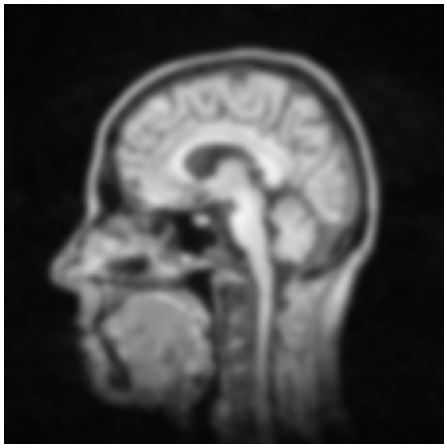
$$S(\nabla u) = |\nabla u_1|^2 + |\nabla u_2|^2 . \tag{1.21}$$

This term represents the assumption that the solution is smooth, meaning that spatial changes are small. A formulation as a minimization problem of an energy functional

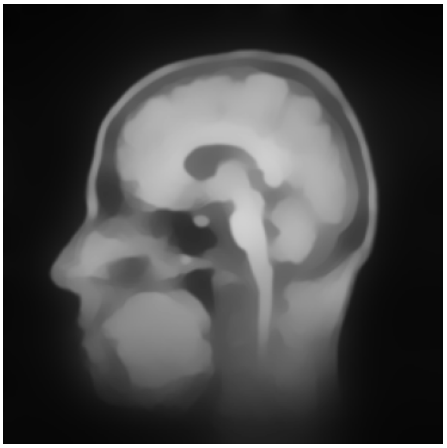
(A) Original image.



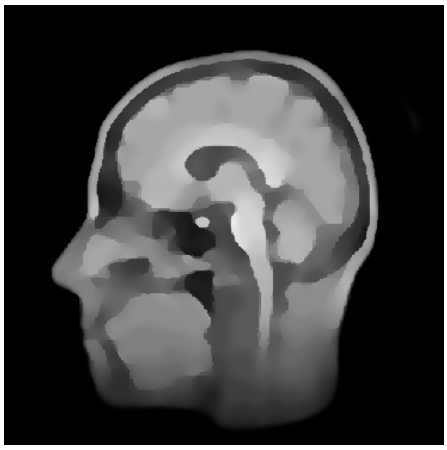
(B) Gradient magnitude of the original image. Values shifted to $[0,255]$ for better visibility.



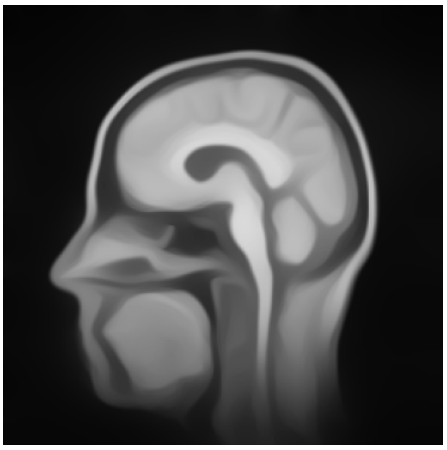
(C) Linear diffusion ($t = 10$).



(D) Perona-Malik diffusion ($t = 200$, $\varepsilon = 7$).



(E) Edge-enhancing anisotropic diffusion ($t = 200$, $\varepsilon = 5$).



(F) Coherence-enhancing anisotropic diffusion ($t = 200$, $\varepsilon = 5$).

FIGURE 1.2: Different diffusion methods applied to the MRI slice of a head.

can also be obtained for the general diffusion equation as given in Equation (1.3). Weickert et al. [WWW13] again use the diffusion tensor D and get

$$E(u) = \frac{1}{2} \int_{\Omega} \nabla^{\top} u \, D \, \nabla u \, \mathrm{d}x \, \mathrm{d}y. \tag{1.22}$$

Now, if we replace the diffusion tensor with the unit matrix, we obtain linear diffusion as in Section 1.2 and the energy functional becomes

$$E(u) = \frac{1}{2} \int_{\Omega} (\nabla u_1)^2 + (\nabla u_2)^2 \, \mathrm{d}x \, \mathrm{d}y. \tag{1.23}$$

The similarity between the energy functional of diffusion in Equation (1.23) and the smoothness term in Equation (1.21) is obvious. More sophisticated diffusion methods can thus be implemented directly in the method of Horn and Schunck to refine the assumption of a smooth solution. In the choice of regularizer, a distinction can be made between image-driven ones, which include the isotropic and anisotropic diffusion methods presented earlier, and flow-driven ones, which take into account discontinuities of the unknown flow field by preventing smoothing at or across flow discontinuities [Wei+03]. In the following chapter, we will discuss variational methods for estimating the optical flow in general as well as more recent methods.

2 Optical Flow

In this chapter, we will take a closer look at the basics and development of different methods in estimating the optical flow of an image sequence. This is helpful in understanding why and how the blocking artifacts considered in this thesis are created in the first place.

In computer vision, a variety of methods have been developed and refined over the last decades that enable machines to perceive their environment by detecting certain objects in an image and assigning them to a class using object recognition. However, many well studied methods in this area consider the data of only a single image. In the case of video inputs, this leads to valuable temporal information being lost. Objects could be identified even better by additionally considering their motion. This is where optical flow has several advantages, since it is used to represent such motion information in image sequences.

The optical flow of an image sequence is a vector field representing the movement of a point between consecutive images of the sequence. This movement is usually caused by independent object motion in the scene, but also by the relative motion between the observer, usually a camera, and the environment. First descriptions of optical flow go back to studies of the visual perception of the environment by animals in motion [Gib50]. A more recent example is environment sensing in autonomous driving, where both the camera and surrounding objects move. Basic methods for determining optical flow go back several decades to works by Lucas and Kanade [LK81] and Horn and Schunck [HS81]. We have already mentioned the method of Horn and Schunck in Chapter 1, where an iterative algorithm is used to determine the optical flow pattern. They assume that the apparent velocity of brightness patterns in the image vary smoothly almost everywhere and require several constraints on smoothness and flow velocity. Meanwhile, there are more advanced methods, some of which we present in Section 2.1.

Even to this day, determining the optical flow is a major challenge in image processing. Problems such as *motion blur* and the *aperture problem* play a crucial role. Motion blur results from the movement of an object during the exposure time of a camera. It increases proportionally to the exposure time and the angular velocity of the object relative to the camera. The blur causes edges and patterns in the image to be much harder to detect. The aperture problem describes constraints imposed by viewing objects through a bounded aperture. For example, the motion of a simple one-dimensional structure such as an edge cannot be uniquely determined if its ends are not visible through the aperture. Being able to estimate the optical flow is useful for a variety of tasks in computer vision like object and motion detection.

2.1 Methods

In this section, we will briefly review the development of methods for determining optical flow. Beauchemin and Barron [BB95] provide a summary of some of the main methods in the last century. These include variational methods, multiconstraint methods, frequency-based methods and more. Modern methods, on the other hand, are mostly learning-based, which means that they are based on artificial intelligence approaches and deep neural networks.

Variational Methods

Variational methods have long been considered successful, since they allow transparent modeling and produce dense and accurate results [MSB17]. As already shown in Section 1.4, they are based on the minimization of an energy functional. It is composed of a data term and a smoothness term reflecting several assumptions on the image sequence. These may include a brightness constancy assumption, a gradient constancy assumption or a smoothness constraint to preserve discontinuities. The data term is usually strongly non-convex, which is why several variational methods linearize it. Since this makes the estimation of large motion difficult, Brox et al. [Bro+04] propose a warping strategy. They present coarse-to-fine warping that produces smaller angular errors and is very robust under noise while avoiding linearisations in the data terms. Modern approaches to optical flow estimation partially adopt ideas from variational methods, but are mostly learning-based methods, which we examine in more detail in the following.

Learning-Based Approaches

In recent years, the greatest successes in optical flow estimation have been achieved by learning-based methods. These usually involve training a Convolutional Neural Network (CNN). CNNs are a class of deep neural networks and are mostly used in automated processing of image or audio data. We will briefly introduce a few of the most popular methods.

FlowNet FlowNet [Dos+15] was the first successful CNN-based method for calculating optical flow. It introduces two core concepts for learning-based methods. The first is feature matching, which involves not only identifying features in one image, but also recognizing them elsewhere in another image. The other is the introduction of a correlation layer that has such matching capabilities and can operate at different scales. FlowNet was able to achieve best ratings on common benchmarks at the time and outperformed conventional methods.

SPyNet Ranjan and Black [RB17] implement a classical spatial-pyramid formulation into a deep learning method. They show that this can help to estimate optical flow with less model parameters. Combining spatial pyramids with deep learning in contrast to FlowNet, which uses only deep learning, has some advantages. One is that SPyNet is significantly smaller and simpler than FlowNet. The movements in each layer of the pyramid are rather small, which makes the method more effective than previous ones.

PWC-Net One of the more recent methods is presented by Sun et al. [Sun+18] and is called PWC-Net. They implement a pyramid scheme and warping strategies. In addition, the method uses features of the first image together with warped features to construct a cost volume, which is finally processed by a CNN to estimate the optical flow. The combination of these three building blocks *pyramid*, *warping* and *cost volume* (PWC) have proven to be very effective. PWC-Net is significantly smaller and easier to train even than the improved FlowNet 2.0 of Ilg et al. [Ilg+17]. Recently, Sun et al. presented an empirical investigation of

different CNN-based models for optical flow estimation [Sun+20]. There, they also provide a very clear scheme for PWC-Net, which can be seen in Figure 2.1. On the left you can see the feature pyramid, which calculates a cost volume using both images. For this purpose, the most similar pixels in features of one image are found in the other image. In the second step, a CNN is used as flow estimator to calculate the optimal flow for the lowest pyramid layer based on the cost volume and the features of the first image. Then the flow is upsampled (shown as an arrow pointing upwards in the diagram) and the features of the second image are warped onto it. This procedure is repeated for all layers of the pyramid until finally the full resolution optical flow is obtained.

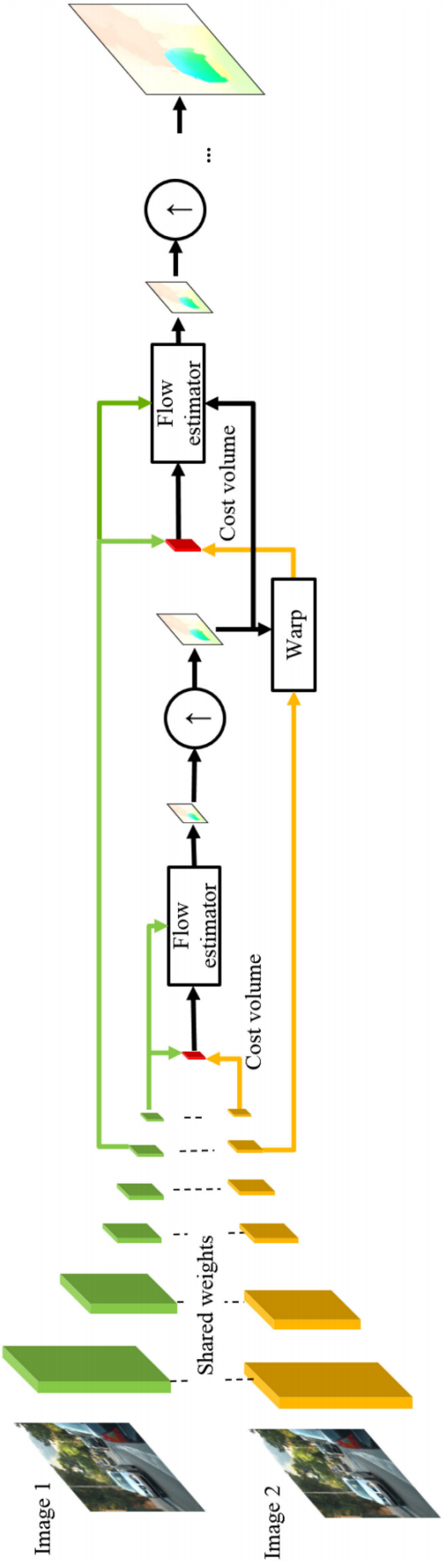


FIGURE 2.1: Schematic overview of PWC-Net. Image adapted from [Sun+20].

2.2 RAFT

The Recurrent All-Pairs Field Transforms (RAFT) method presented by Teed and Deng [TD20] is currently one of the best methods for determining optical flow. It is based on a deep network architecture with iterative updating of the flow field. One drawback of this method is the emergence of blocking artifacts. Since the goal of this work is to reduce the error of the RAFT method, we will examine the origin of these artifacts in more detail. For this purpose, the approach of RAFT will be briefly explained.

Figure 2.3 shows a schematic overview of how the algorithm works. Starting from two consecutive frames of an image sequence, the feature encoder is used to extract features from both images. To this end, each dimension of the images is first reduced to one-eighth of its original size. After this reduction step, a vector is calculated for each pixel in both images. Using these feature vectors, all pixels in the first image are correlated pairwise with those of the second image. This results in a four-dimensional correlation volume, whose last two dimensions are pooled to create multi-scale volumes. Together with a context encoder that only extracts features from the first image, the flow field is then iteratively updated using a gated recurrent unit (GRU). In each step, values are selected in the correlation volume based on the current flow. The result is the optical flow between the two images at one-eighth of the original resolution.

Due to the pooling, the network calculates the optical flow at one-eighth of the original resolution. However, the final output is of the same resolution as the original image. Therefore, there is an upsampling step in between. In this upsampling process, the vector values of the high-resolution flow are determined using a weighted combination of the nine neighboring low-resolution pixels. For this, weights are used for each neighbor, which are an additional output of the update iterator.

Even though the upsampling yields good results, it creates blocking artifacts of size 8×8 pixels. To understand this a little better, consider Figure 2.2. It shows the upsampling process for two high-resolution pixels located inside two neighboring low-resolution pixels. While for the orange pixel in the left image the left nine large neighboring blocks (light orange) are used for calculation, the blue pixel on the right uses the right nine blocks (light blue). In general, the values of the smaller high-resolution blocks can be chosen in a way that a smooth junction is created between the coarser blocks by weighting the neighboring values appropriately. However, in extreme cases (which can be easily devised), visible edges may still appear at the borders of the 8×8 blocks, that do not appear in the correct optical flow.

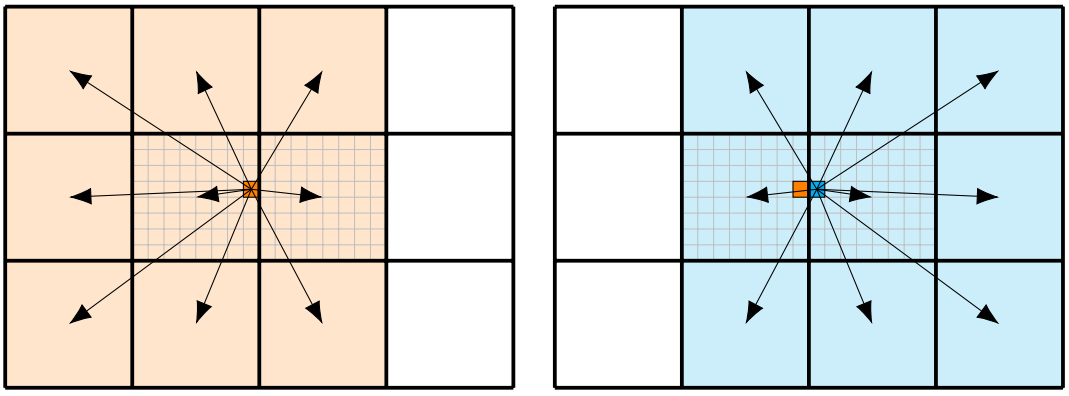


FIGURE 2.2: Upsampling for two neighboring pixels on the edge of two 8×8 blocks. The orange pixel on the left uses the weighted values of 9 coarser blocks, of which 3 are different from the 9 coarse blocks of the blue pixel on the right.

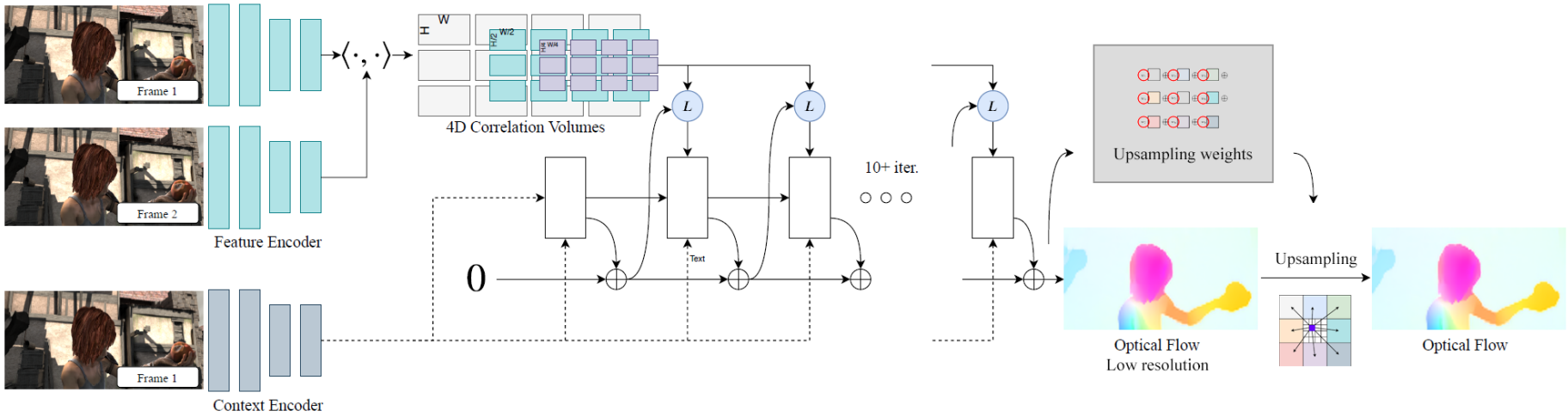


FIGURE 2.3: Schematic overview of the RAFT algorithm (adapted from [TD20]).

3 Implementation

In this chapter, we present details on the implementation of a general diffusion method. To this end, we first introduce a theoretical formulation of the input data and our conventions. Since we want to test and develop a variety of different diffusion methods with several parameters, it is beneficial to establish a framework to facilitate this task. This requires a general numerical implementation of diffusion algorithms. We then adapt this implementation specifically to vector fields such as optical flow.

3.1 Prerequisites

We start by specifying the theoretical foundations of our problem. The goal is to develop a framework to apply various diffusion algorithms to images. To do this, we start by considering continuous gray-scale images. These are a mapping u from a rectangular domain, the image domain, to a one-dimensional domain of real values and we obtain

$$u : \mathbb{R}^2 \supset \Omega \to \mathbb{R} \quad \text{with} \quad \Omega := (0, a_x) \times \big(0, a_y\big), \tag{3.1}$$

where a_x and a_y are the width and height of the image, respectively. Next, we discretize this image domain, since image data is usually given only on a regular grid consisting of pixels. Our image is then a set

$$\big\{u_{i,j} \mid i = 1, \dots, W; j = 1, \dots, H\big\}, \tag{3.2}$$

where each (i, j) corresponds to one pixel with a value $u_{i,j} \in \mathbb{R}$. In general terms, this corresponds to an image with resolution $W \times H$. Unless otherwise indicated, we always specify our image in x- and y-directions and the corresponding grid widths are h_x and h_y, whereby both in theory and in practice one often sets $h := h_x = h_y$.

3.2 Numerical Scheme for Diffusion

In Chapter 1, we described the theoretical diffusion process in image processing. We will now consider a numerical scheme for its practical implementation. One of the main issues with implementing anisotropic diffusion filters numerically is that they often suffer from dissipative artifacts and have difficulty approximating rotation invariances. Weickert et al. [WWW13] approach this problem by using finite differences to discretize a gradient descent of a quadratic energy. This results in a general (3×3)-stencil that provides more stable results than conventional methods. We will therefore now take a closer look at their approach.

The first step is to consider the evolution equation of the diffusion process on an image u. It is a gradient descent of the quadratic energy and reads

$$E(u) = \frac{1}{2} \int_\Omega \nabla^\top u \, D \, \nabla u \, \mathrm{d}x \mathrm{d}y, \tag{3.3}$$

where D is the diffusion tensor as introduced in Chapter 1. D is time-invariant but can be chosen space-variant and - while making use of its symmetry - we write its entries as

$$D = \begin{pmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{pmatrix}. \tag{3.4}$$

In order to understand why we use Equation (3.3), one may consider the gradient descent

$$\frac{du_{i,j}}{dt} = -\frac{\partial E(u)}{\partial u_{i,j}} \tag{3.5}$$

and observe that the right-hand side is equal to the right-hand side of the general diffusion equation (1.3) which reads $\operatorname{div}(D \cdot \nabla u)$. Using a regular grid $\{1,\dots,W\} \times \{1,\dots,H\}$ with square grid cells of size $h \times h$ to discretize the two-dimensional image domain in both directions, we can discretize Equation (3.3) to get

$$E(u) = \frac{1}{2} \sum_{i=0}^{W} \sum_{j=0}^{H} \left(a u_x^2 + 2b u_x u_y + c u_y^2 \right)_{i+\frac{1}{2}, j+\frac{1}{2}}. \tag{3.6}$$

Note that a, b and c are assumed to have approximations in $\left(i+\frac{1}{2}, j+\frac{1}{2}\right)$ following the ideas in [WWS06]. There it is illustrated that in the case of nonlinear diffusion, the center of a four-pixel cell is well suited to discretize locally. This same assumption is also applied to u_x^2, u_y^2 and $u_x u_y$. In addition, a, b, c, u_x and u_y must satisfy Neumann boundary conditions at the edges of the image.

To discretize our derivatives u_x and u_y, we use forward differences and get

$$\begin{aligned}
\boxed{\cdot\!-\!\cdot} &:= D_x u_{i,j} &&:= \frac{u_{i+1,j} - u_{i,j}}{h}, \\
\boxed{\cdot\!-\!\cdot} &:= D_x u_{i,j+1} &&:= \frac{u_{i+1,j+1} - u_{i,j+1}}{h}, \\
\boxed{\cdot\,|\,\cdot} &:= D_y u_{i,j} &&:= \frac{u_{i,j+1} - u_{i,j}}{h}, \\
\boxed{\cdot\,|\,\cdot} &:= D_y u_{i+1,j} &&:= \frac{u_{i+1,j+1} - u_{i+1,j}}{h}.
\end{aligned} \tag{3.7}$$

Using these discretizations, we can achieve approximations of u_x^2, u_y^2 and $u_x u_y$ with second-order of consistency. To this end, one can apply affine combinations of the arithmetic mean and the geometric mean, which leads to

$$\begin{aligned}
u_x^2\big|_{i+\frac{1}{2},j+\frac{1}{2}} &\approx \frac{1}{2h^2} \cdot \Big(\Big(1 - \alpha_{i+\frac{1}{2},j+\frac{1}{2}}\Big) \cdot \big(\boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\!-\!\cdot} + \boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\!-\!\cdot} \big) \\
&\quad + 2\alpha_{i+\frac{1}{2},j+\frac{1}{2}} \cdot \boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\!-\!\cdot} \Big), \\
u_y^2\big|_{i+\frac{1}{2},j+\frac{1}{2}} &\approx \frac{1}{2h^2} \cdot \Big(\Big(1 - \alpha_{i+\frac{1}{2},j+\frac{1}{2}}\Big) \cdot \big(\boxed{\cdot\,|\,\cdot} \cdot \boxed{\cdot\,|\,\cdot} + \boxed{\cdot\,|\,\cdot} \cdot \boxed{\cdot\,|\,\cdot} \big) \\
&\quad + 2\alpha_{i+\frac{1}{2},j+\frac{1}{2}} \cdot \boxed{\cdot\,|\,\cdot} \cdot \boxed{\cdot\,|\,\cdot} \Big), \\
u_x u_y\big|_{i+\frac{1}{2},j+\frac{1}{2}} &\approx \frac{1}{2h^2} \cdot \Big(\frac{1 - \beta_{i+\frac{1}{2},j+\frac{1}{2}}}{2} \cdot \big(\boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\,|\,\cdot} + \boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\,|\,\cdot} \big) \\
&\quad + \frac{1 + \beta_{i+\frac{1}{2},j+\frac{1}{2}}}{2} \cdot \big(\boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\,|\,\cdot} + \boxed{\cdot\!-\!\cdot} \cdot \boxed{\cdot\,|\,\cdot} \big) \Big),
\end{aligned} \tag{3.8}$$

where $\alpha_{i+\frac{1}{2},j+\frac{1}{2}}$ and $\beta_{i+\frac{1}{2},j+\frac{1}{2}}$ are arbitrary weights. These provide another way to influence the diffusion process by steering the weighting of arithmetic and geometric mean. The

parameter α is used for both the x- and y-directions, since there is no need to distinguish between them. The weights can be chosen as constants. For example, if both values are set to zero, a standard discretization of average forward differences is obtained. However, we can also integrate a spatial variance. Then the weights can be made dependent on the entries of the diffusion tensor.

Finally, we can apply the derivative approximations in Equation (3.8) to the discrete energy in Equation (3.6) to calculate the right hand side in Equation (3.5). The result gives the desired discretisation of $\operatorname{div}(D \cdot \nabla u)$ and can be represented by the (3×3)-stencil, which can be found in Appendix A.1.

3.3 Diffusion on Vector-Valued Images

The previously presented methods for diffusion have always referred to two-dimensional images with only one value per pixel. A naive approach to apply these concepts to multi-valued images, e.g. color images or optical flow, is processing each channel separately and reassemble the results of each channel afterwards. However, merging the outputs of the seperated edge detectors is mostly a heuristic combination without any theoretical background. In doing so, influences of the channels among each other are lost that can be helpful in localizing edges. For this reason, the method of Sapiro and Ringach [SR96] is presented, which extends the ideas of the previous Section to vector-valued data such as optical flow.

First, we let $x = (x_1, x_2) \in \mathbb{R}^2$ and restrict ourselves to two-dimensional, two-valued images $f(x) : \mathbb{R}^2 \to \mathbb{R}^2$, such as the optical flow. We denote the value of each component (or channel) as $f_i(x) : \mathbb{R}^2 \to \mathbb{R}$, so $f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix}$. Using this, we define

$$f^{(k)}(x) := \left(\frac{\partial f_1}{\partial x_k}, \frac{\partial f_2}{\partial x_k} \right) \tag{3.9}$$

where the ith component represents the partial derivative $\frac{\partial f_i}{\partial x_k}$ at $x \in \mathbb{R}^2$.

If we then consider the values of two points $p = \left(x_1^0, x_2^0\right)$ and $q = \left(x_1^1, x_2^1\right)$ of our image, their difference is given by $\Delta f = f(p) - f(q)$. If the Euclidean distance $d(p,q)$ between p and q approaches zero, the difference becomes

$$\mathrm{d}f = \sum_{i=1}^{2} f^{(i)} \, \mathrm{d}x_i. \tag{3.10}$$

We are now interested in two quantities of our image: the direction of maximum change of $f(x)$ and the rate or rather the absolute value of this change. We therefore want to maximize the squared norm of Equation (3.10)

$$\left(\mathrm{d}f\right)^2 = \sum_{i=1}^{2} \sum_{j=1}^{2} f^{(i)} \cdot f^{(j)} \, \mathrm{d}x_i \, \mathrm{d}x_j \tag{3.11}$$

under the condition

$$\sum_{i=1}^{2} \mathrm{d}x_i \, \mathrm{d}x_i = 1. \tag{3.12}$$

Equation (3.11) is also called the first fundamental form. If we now define

$$g_{ij}(x) := f^{(i)}(x) \cdot f^{(j)}(x), \tag{3.13}$$

where $\cdot$ is the scalar product in $\mathbb{R}^2$, we get the matrix representation

$$(\mathrm{d}f)^2 = \begin{pmatrix} \mathrm{d}x_1 \\ \mathrm{d}x_2 \end{pmatrix}^T \underbrace{\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}}_{=:G} \begin{pmatrix} \mathrm{d}x_1 \\ \mathrm{d}x_2 \end{pmatrix}. \tag{3.14}$$

This form allows easy determination of the directions of changes in the image as well as their strength. One obtains the extrema of $(\mathrm{d}f)^2$ via the eigenvectors of the matrix G, and the values obtained here are the corresponding eigenvalues. Since $g_{ij} = g_{ji}$, G is symmetric and its characteristic polynomial with eigenvalues λ is

$$\lambda^2 - (g_{11} + g_{22})\lambda + (g_{11}g_{22} - g_{12}^2). \tag{3.15}$$

A simple calculation shows that the eigenvalues are then

$$\lambda_\pm = \frac{g_{11} + g_{22} \pm \sqrt{(g_{11} - g_{22})^2 + 4g_{12}^2}}{2}. \tag{3.16}$$

Following the procedure in [DZ86], the problem stated in Equation (3.11) can be reformulated to finding θ maximizing $F(\theta) = g_{11}\cos^2\theta + 2g_{12}\cos\theta\sin\theta + g_{22}\sin^2\theta$. By using some convenient substitutions, the solution of $\frac{\mathrm{d}F}{\mathrm{d}\theta} = 0$ is given by the angles θ_+ and θ_- with (modulo π)

$$\begin{aligned} \theta_+ &= \tfrac{1}{2}\arctan\tfrac{2g_{12}}{g_{11}-g_{22}}, \\ \theta_- &= \theta_+ + \tfrac{\pi}{2}. \end{aligned} \tag{3.17}$$

This leads to the corresponding eigenvectors

$$v_\pm = (\cos\theta_\pm, \sin\theta_\pm). \tag{3.18}$$

Let us briefly clarify the meaning of these quantities. The eigenvectors $v_\pm$ indicate the direction of maximum and minimum change, while the eigenvalues $\lambda_\pm$ indicate the corresponding maximum and minimum rates of change. Specifically, θ_+ is the direction of maximum change and λ_+ is the maximum rate of change.

While $\lambda_+ \equiv \|\nabla f\|^2$ and $\lambda_- \equiv 0$ is always obtained for monovalent images, the minimum rate of change λ_- can be nonzero in our multivalued case. It is therefore important to note that the strength of an edge is not only given by the maximum rate of change λ_+, but rather the difference between λ_+ and λ_- is decisive. Sapiro and Ringach therefore propose to detect discontinuities in the image via a function $h(\lambda_+, \lambda_-)$ that reflects the difference between λ_+ and λ_-. With this, one can also reformulate the diffusion equation to

$$\frac{\partial f}{\partial t} = h(\lambda_+, \lambda_-)\frac{\partial^2 f}{\partial \theta_-^2}, \tag{3.19}$$

where a suitable choice for h is a decreasing function in the difference $\lambda_+ - \lambda_-$. However, this is not of any further relevance to us.

For our purposes, the insights from Equations (3.16) and (3.18) are particularly important. They provide us with an edge detector for vector-valued images that considers all dimensions of the vectors equally. Especially the direction of largest change will be very helpful in analyzing the error of the RAFT method in Section 4.2.

4 Methods for Diffusing Optical Flow

In this chapter, we present the methods we developed and tested for diffusion of optical flow and improvement of the RAFT method. We first apply conventional linear and nonlinear methods mainly for comparison reasons. However, since a systematic error is expected in the RAFT method, we analyze this error in detail and visualize it. Using the knowledge gained from the analysis, we then develop diffusion algorithms specifically adapted to RAFT's error.

4.1 Conventional Methods

With the developed framework, well-known methods of diffusion can be applied easily. As an example, we can simply insert the unit matrix as diffusion tensor and specify a number of time steps as well as a time step size to get linear diffusion. Afterwards, we can compare the results for different values of the parameters we devised in our methods. Linear diffusion does not require any further discussion, since we don't have great influence on the diffusion process.

With nonlinear methods, we can exert greater influence. The Perona-Malik model introduces first approaches to edge detection. For this purpose, we multiply our image gradient with a diffusivity function as presented in Section 1.3. This diffusivity depends on the magnitude of the largest change in the image, which we can estimate as presented in Section 3.3.

If we consider anisotropic diffusion, we compute a diffusion tensor by specifying its eigenvalues and eigenvectors. To do this, we again use the magnitude of the largest changes in the image but additionally consider its direction as presented in Section 3.3. We can choose the first eigenvector of the diffusion tensor the same as the direction of the largest change and the second eigenvector orthogonal to it. This allows distinguishing between diffusion across edges and along edges.

More crucial is the choice of eigenvalues. In the context of this work, it is of great importance for all methods to preserve edges of objects in the optical flow. For this purpose, we will mainly rely on diffusivity functions. We therefore want to use the standard anisotropic diffusion method to find the best of three common functions for our purposes. These include the Regularized Linear variant

$$g_{\mathrm{RL}}\left(s^2\right) = \frac{1}{2\sqrt{s^2+\varepsilon^2}}, \tag{4.1}$$

the Charbonnier function

$$g_{\mathrm{CH}}\left(s^2\right) = \frac{1}{\sqrt{1+s^2/\varepsilon^2}} \tag{4.2}$$

and Perona-Malik's method

$$g_{\mathrm{PM}}\left(s^2\right) = \frac{1}{1+s^2/\varepsilon^2}. \tag{4.3}$$

There are of course numerous other variants [Bla+98; TP13], but we are not able to compare all of them here.

4.2 Error Analysis of RAFT

General Error Analysis

We now want to get a better understanding of the error that occurs when calculating optical flow using the RAFT method to be able to adapt our methods to it. To this end, let us first consider an example. Figure 4.1a shows two frames of the Sintel data set overlaid with each other. This makes the motion between the frames fairly visible. Figure 4.1b shows the optical flow calculated by RAFT using a color representation, where no systematic errors can be discerned. In addition, Figure 4.1c shows the difference between the optical flow of RAFT and the actual ground truth of the optical flow from Sintel. This corresponds to the error made by RAFT. Again, no pattern in the error is apparent at first. It is noticeable that the error due to blocking artifacts does not seem to have a large influence on the overall error, since errors seem to occur mainly at edges.

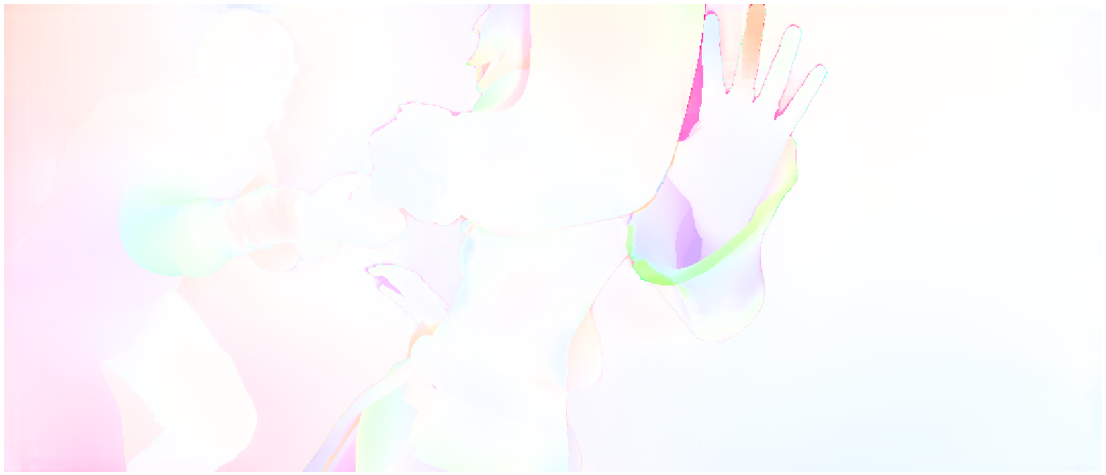
If we enlarge the view and look at the hand in the upper right corner as shown in Figure 4.2, then we can make the artifacts become more visible. There are slight blocking artifacts visible in the scaled optical flow, which are even more noticeable in the error plot. So it seems that such artifacts are created at least in certain parts of the image.

We can additionally consider the method for calculating the largest changes in the image based on the image gradient in Section 3.3, where the direction of this change was calculated as well. Figure 4.3 shows the angle of this largest change mapped to a cyclic colormap. Here we can clearly see a regular grid throughout the image. Not surprisingly, this grid coincides with the position of the blocking artifacts. The same result is obtained for other images in the data set.

However, one thing should be noted. The error caused by the blocking artifacts is relatively small. Especially in Figure 4.1c, it is clearly visible that the biggest errors are caused by inaccurate edge detection and other common problems like occlusions. Blocking artifacts are only detectable when the image is enlarged and the coloring is adjusted.

(A) Original frame and the following frame overlaid with 50% opacity.

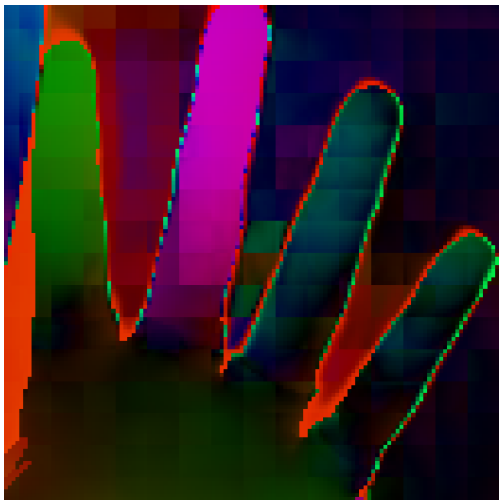
(B) Color representation of the optical flow of the frame as calculated by RAFT.



(C) Color representation of the difference between the optical flow of RAFT and the actual ground truth of the optical flow. Bright values represent a low error and dark, saturated values a higher error.

FIGURE 4.1: Analysis of the optical flow resulting from the RAFT method applied to a sample frame in the Sintel data set.

(A) Zoomed-in and enhanced area of the optical flow in Figure 4.1b.



(B) Zoomed-in and enhanced area of the error of RAFT in Figure 4.1c. Note that the meaning of the color has been reversed. Dark color now represents a low error. Also, a logarithmic transformation has been applied for better visibility.

FIGURE 4.2: Enhanced close-up of RAFT's optical flow and its error from Figure 4.1.

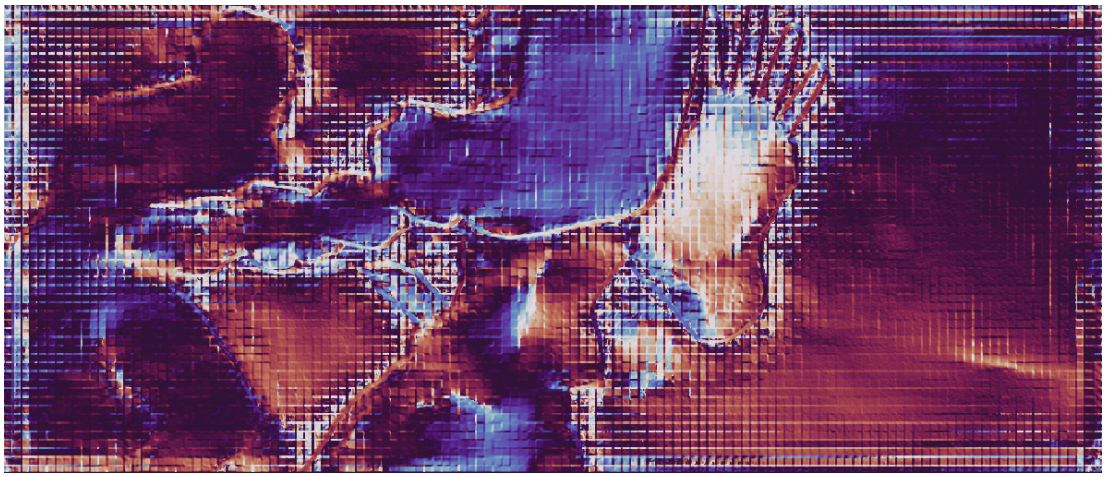


FIGURE 4.3: Direction of largest change in the optical flow. The colormap is cyclic and represents the angle $\theta_1 \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Blocking Artifacts in Detail

In our general examination of the error, we noticed blocking artifacts. Let's examine these in a bit more detail. First, it is easily noticeable that the artifacts are arranged in a regular grid consisting of 8×8 pixel sized cells. The grid's position follows a regular pattern as well. In the case of this data set, all images are of the same size and therefore the grid is always in the same location. This is consistent with the downscaling and upscaling in RAFT to one-eighth of the original image size presented in Chapter 2, as this is expected to produce exactly such regular artifacts. The regularity of the artifact's positions is very helpful in developing our methods in Section 4.3.

We can now use this knowledge to additionally investigate the error within the blocking artifacts. To do this, the error can be calculated individually for each pixel in an 8×8 block. This can be thought of as removing all pixels except the one currently under consideration from all blocks in the image, and calculating the average endpoint error (AEE) only for the pixels that remain. See Section 5.2 for an explanation of how the AEE is calculated. To be more precise, we calculate the AEE 64 times and each time for $1/64$ of the original amount of pixels. This can be averaged for all blocks in the optical flow and across all images in the data set. The result can be seen in Figure 4.4, where a pattern is visible. In average, the error in the upper left corner is smaller than in the lower right corner, and there seems to be a linear gradient from the upper left to the lower right corner. Although this difference is relatively small, it can definitely be taken into account in the development of the methods, since it is clearly a systematic error and we have already noted that the expected overall improvement is quite small as well. Note that the colormap used can be seen to the right of the plot in Figure 4.4 and will be used several times in the course of this work, where purple always represents low values and yellow high values.

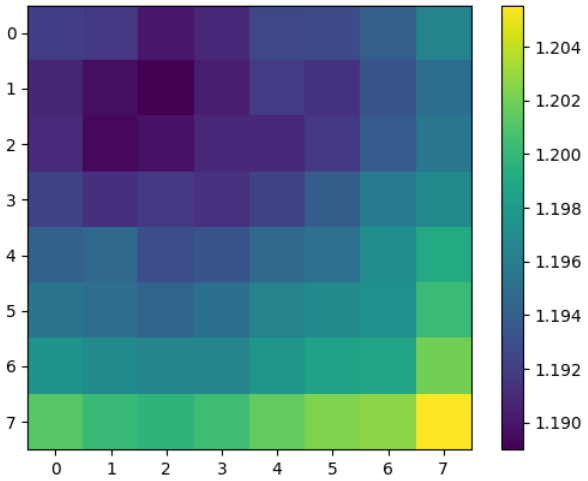


FIGURE 4.4: Average endpoint error of each pixel in an 8×8 block in all blocks of the RAFT results on Sintel.

4.3 RAFT Adapted Methods

In this section, the most promising methods we developed to improve the error are presented. We will mainly focus on smoothing the blocking artifacts. This is mainly due to the fact that in this work only the optical flow and not the original image sequences are considered. Thus, it would be surprising if diffusion and smoothing could be used to fix other problems such as poor edge detection that do not follow any obvious regularity. In a different approach, one might also use the original frames as input.

Diffusion Direction Customization (DDC)

We can control the direction and strength of diffusion ourselves. The easiest way to do this is to modify the eigenvalues and eigenvectors of the diffusion tensor. The idea now is that we can reduce the blocking artifacts by smoothing over the edges of the artifacts. To this end, we choose the eigenvectors pointing outward at the edge of an artifact. This can be seen in Figure 4.5. We will call this method *Diffusion Direction Customization* (DDC).

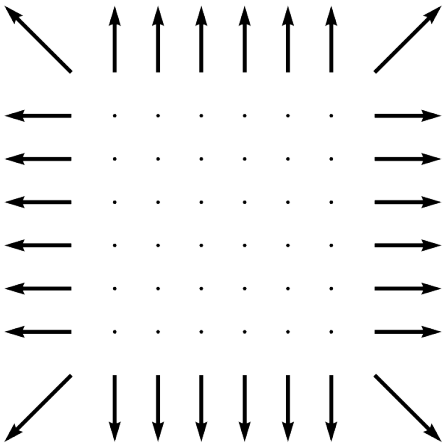


FIGURE 4.5: Custom eigenvectors of the diffusion tensor for an 8×8 block.

We have already found that the positions of the artifacts are known. Thus, we can spread the blocks of eigenvectors over the whole image according to these positions. This pattern can be see in Figure 4.6. The second eigenvector is always othogonal to the first.

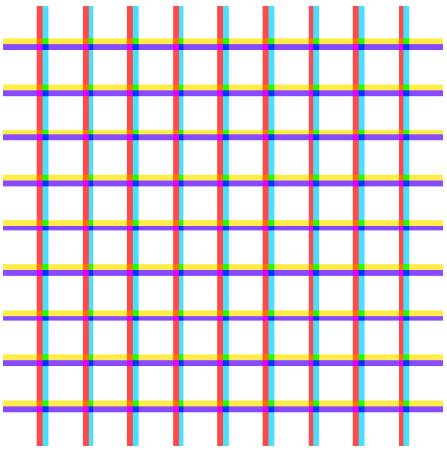


FIGURE 4.6: Custom eigenvectors of the diffusion tensor on a grid. The coloring corresponds to the direction of the vectors.

The choice of eigenvalues is a bit more complicated. We can naively set the first eigenvalue to 1 and the second to a value close to 0. This corresponds to strong diffusion across the edges of the blocking artifacts and almost none along the edges. The problem is that edges of objects in the flow are then smoothed as well, which of course should be avoided. So we have to distinguish between edges of objects and edges of blocking artifacts. This is one of the main challenges in developing these methods.

One approach is to again use classical edge detectors as presented in Section 4.1. Based on the image gradient, these yield large values for small changes in the image and vice versa. We can therefore use these values directly as eigenvalues and expect little diffusion at edges. We will address the problem of distinguishing edges of blocking artifacts from edges of objects later using a different method.

Remove and Interpolate (RI)

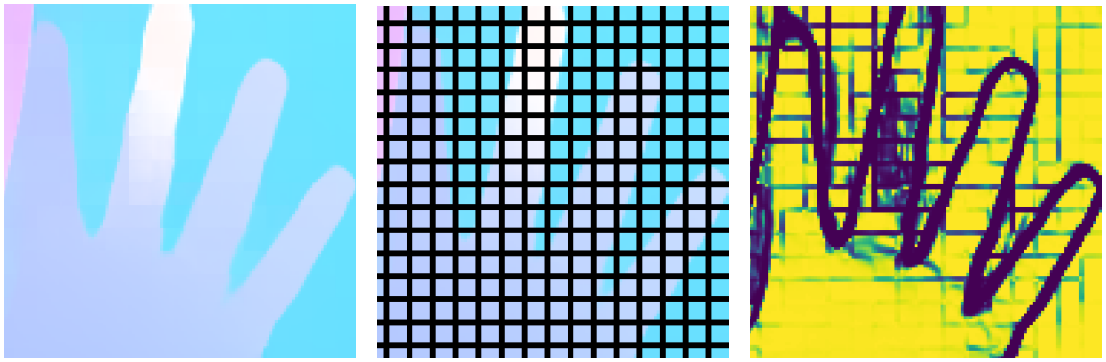
Another concept we've been exploring is what we'll call *Remove and Interpolate* (RI). This sums up what the method is supposed to do: we remove the unwanted edges of blocking artifacts and interpolate the missing values based on the surrounding ones. This can be applied to different quantities.

The first and most obvious quantity to apply RI to is the optical flow itself. This process is shown in Figure 4.7. We remove the vectors at the edge of each 8×8 block in the image. This can be seen in Figure 4.7b. We then interpolate the resulting grid of missing values. For this we use the interpolate function for grid data from SciPy[1]. It provides three possible types of two-dimensional interpolation: cubic, linear and nearest. SciPy's documentation explains these in a bit more detail, but we want to present at least the basic functionality. The cubic interpolation uses a piecewise cubic, continuously differentiable, and approximately curvature-minimizing polynomial surface to determine the missing values (see Figure 4.7d). The linear method decomposes the input point set into N-D simplexes and interpolates linearly on each simplex (see Figure 4.7e). "Nearest" simply uses the value of the closest pixel (see Figure 4.7f). In the images, we can see that the nearest interpolation produces quite hard edges. With the cubic method, clearly wrong directions are calculated in some places, as can be seen in the example on the middle finger. The linear method does not preserve the edges cleanly. We will evaluate these methods further in Chapter 5. In this first variant, edges are not taken into account yet.

However, we can additionally apply an edge detector to this method to preserve important edges of objects. For this, we consider the values of the largest change in the image and can again calculate a diffusivity (see Figure 4.7c). If the diffusivity is smaller than a predefined threshold, then the associated vector in the flow is not removed before interpolating.

Another quantity which we can apply RI to is the eigenvalue of the diffusion tensor from the DDC method. We have already found that edges of blocking artifacts are also preserved by our edge detection. However, by applying this method, we can remove these edges and interpolate the missing values. This should allow diffusion to still preserve edges of objects, but smooth edges of blocking artifacts. Again, we examine the three previously mentioned interpolation methods in the results.

[1] https://www.scipy.org/



(A) Original optical flow. (B) Removing values on the grid. (C) Diffusivity for edge detection.

(D) Cubic interpolation. (E) Linear interpolation. (F) Nearest interpolation.

FIGURE 4.7: Remove and Interpolate method on the close-up of a sample frame comparing different interpolation methods.

Template Matching (TM)

We found that our edge detectors also detect the edges of blocking artifacts. This is problematic since our diffusion method preserves or even sharpens edges. Using *Template Matching* (TM), we attempt to explicitly detect the edges of blocking artifacts without detecting the edges of objects. For this purpose, we use the Template Matching function of OpenCV[2]. Using this function, it is possible to find predefined templates in the form of smaller images within larger images. This involves sliding the template over the image, similar to a convolution, and calculating the similarity between template and respective image area. There are six different methods in OpenCV to calculate similarities. Either via the square difference between template and input image at the respective position, their cross correlation, their correlation coefficients or the respective normalized variant of these methods. If the input image has a size of $(W \times H)$ and the template of $(w \times h)$, then the method yields an output image of size $(W - w + 1 \times H - h + 1)$ where each pixel value corresponds to the calculated similarity at the pixel's position. We have compared the methods and for our purposes, the normalized correlation coefficient is clearly superior to the other methods.

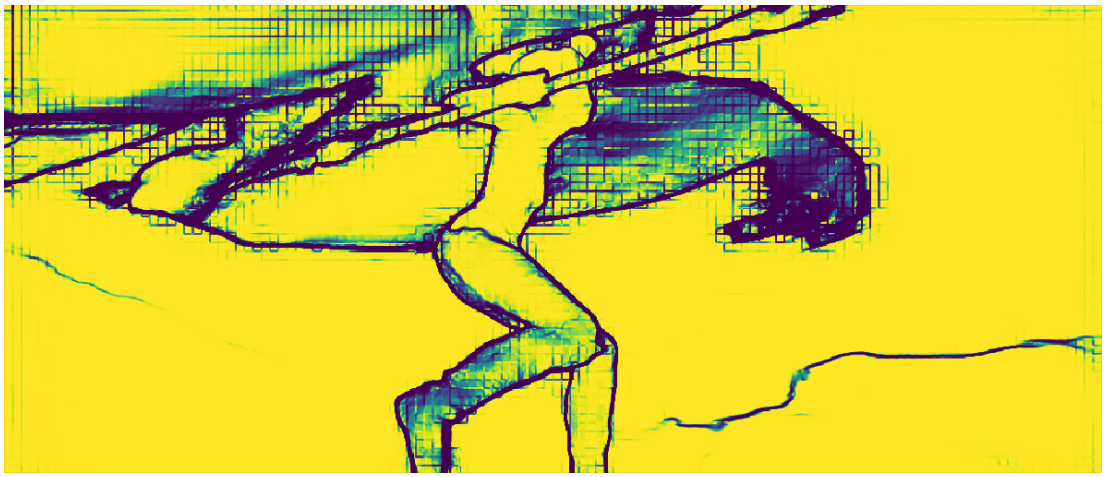
Two questions arise: Which quantity is most suitable for edge detection and what should the templates look like then? To answer the first question, we will first have a look at Figure 4.2b again. There, the artifacts are clearly visible to humans in both images. However, it is quite difficult to create a concrete template that would allow a machine to recognize all artifacts. In Figure 4.7c on the other hand, the edges of blocking artifacts are clearly separated from the background and look fairly uniform. Also, the values are always between 0 and 1 with edges having values close or equal to 0. We will therefore focus mainly on the diffusivity for template matching.

To generate the templates, it is convenient to examine some sample images and note the most common patterns. One of the biggest differences between edges of blocking artifacts and of objects is that in most cases the inner values of a blocking artifact are quite large. Figure 4.8 shows a possible choice of templates. These are all possible combinations of at least two edges in the blocking artifact. If there is only one edge, the method already gives good results. In several trials, these templates have proven to be good detectors.

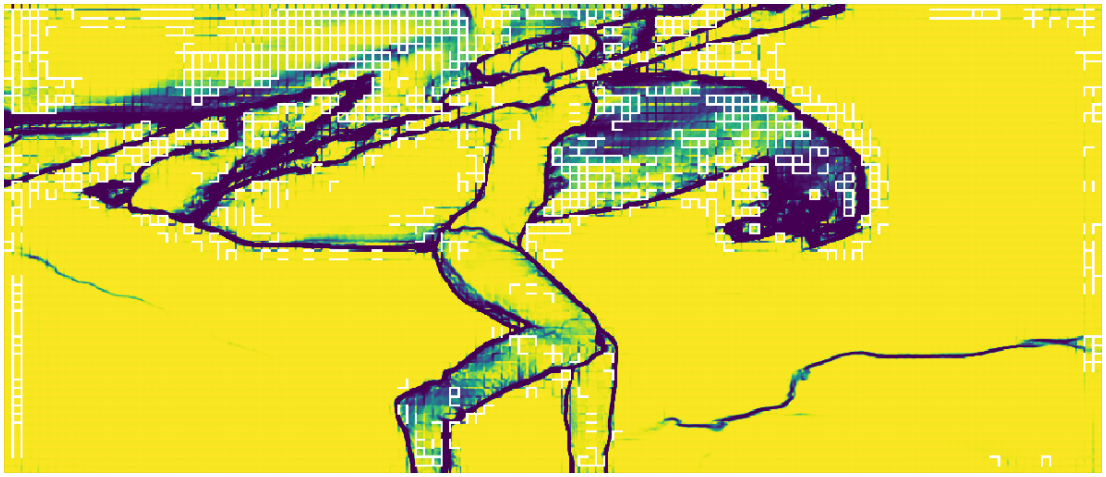
FIGURE 4.8: Match templates.

The functionality of Template Matching quickly becomes clear with the help of an example. Figure 4.9 shows the main steps of the method. First, Figure 4.9a represents the diffusivity as introduced before. Using the Template Matching algorithm, the correlation with each possible image area is calculated for each template. We then select which positions should actually be removed. To do this, we specify two conditions. The first is that the match found is actually on the grid of blocking artifacts. We can conveniently assume this because we know their positions. The second condition is that the calculated correlation must be larger than a predefined threshold. A larger threshold leads to less accepted matches, a threshold of zero removes all values on the grid and we get the same result as in Figure 4.7b. The result after applying these two conditions is shown in Figure 4.9b. Note that only those pixels were marked that were also an edge in the given template. Tests have shown that this gives slightly better results than selecting the entire edge of a matched block, since potentially correct pixel values could be removed in the latter variant.

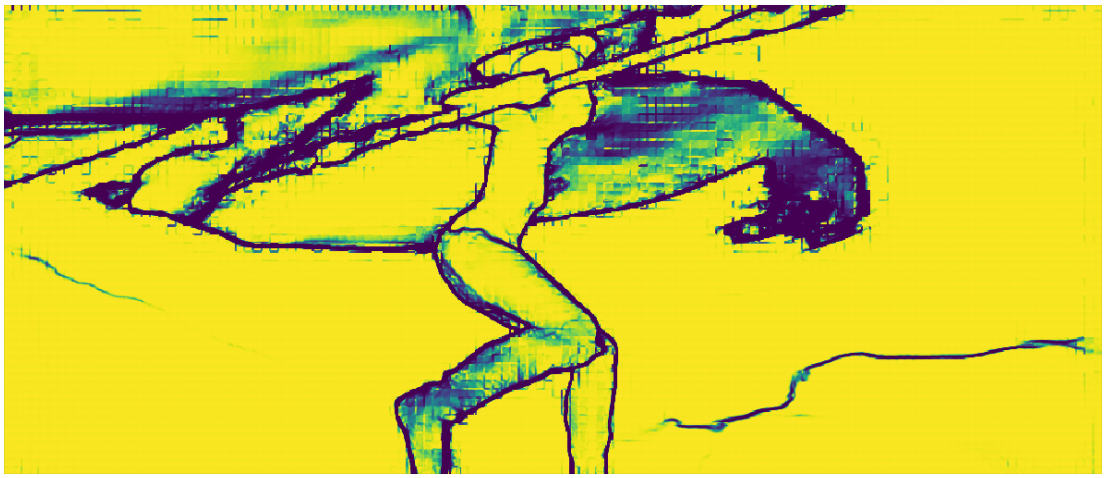
[2] https://opencv.org/



(A) Original diffusivity calculated using the largest changes in the image.



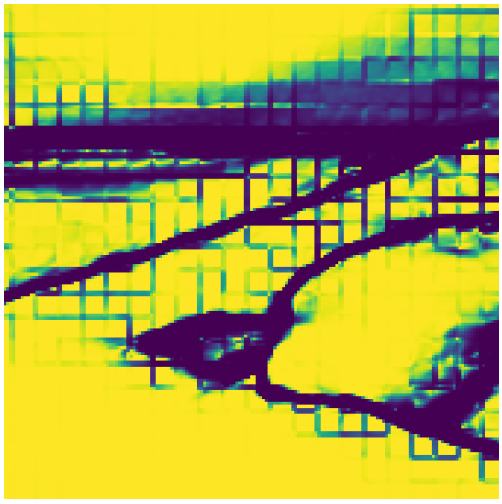
(B) Edges of blocking artifacts detected by Template Matching are marked in white.



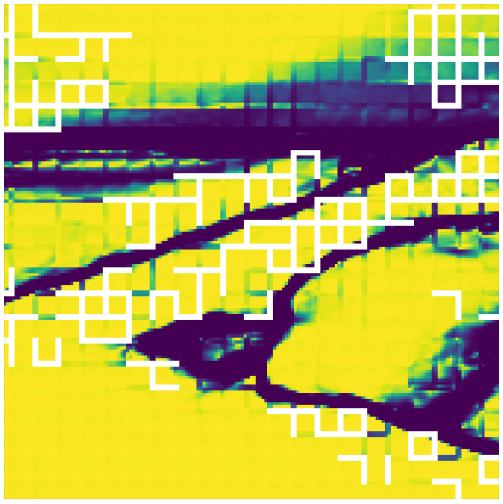
(C) Detected edges interpolated using neighboring values.

FIGURE 4.9: Template Matching applied to the Diffusivity.

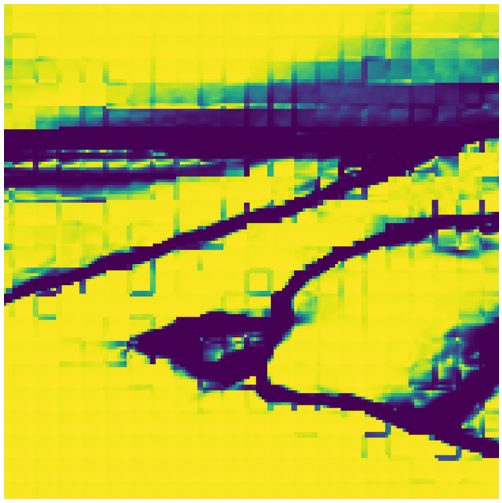
Finally, there is the question of what to do with the edges found. For this, we can again use our Remove and Interpolate method. We remove the found edges from the diffusivity and interpolate the missing values. In Figure 4.10, the whole process can be seen in a close-up. There the interpolation of the values is better visible. It is clearly visible that mainly edges of blocking artifacts are detected and removed, while the values removed from edges of objects are mostly restored by the interpolation afterwards. The interpolation method used is "nearest" in the example. In addition, Figure 4.10d shows the results of the Template Matching, i.e. the calculated correlations.



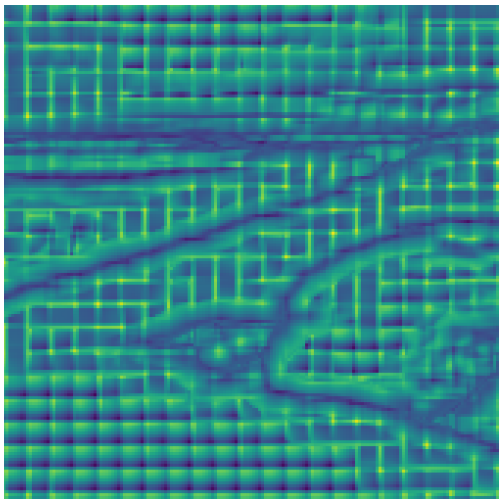
(A) Original diffusivity.



(B) Matched templates in white rarely lie on edges that belong to objects.



(C) Resulting diffusivity by interpolating the matched template edges.



(D) Result of Template Matching.

FIGURE 4.10: Close-up view of Figure 4.9.

Variable Time Steps

The optical flow differs greatly depending on the sequence. For example, in some scenes there is a lot of motion and correspondingly large flow values. With large flow values, we expect a larger error in the calculation. Accordingly, it would be practical to apply more time steps of diffusion in case of a larger error. Of course, calculating the AEE is not possible without knowing the ground truth flow, so we have to rely on other quantities. The y-axis of the graph in Figure 4.11 shows the average absolute value and the AEE of all flows of RAFT in the final pass of Sintel (listed chronologically on the x-axis). The two quantities are scaled in order to be comparable in the graph. Large flow values seem to be related to a larger error. This confirms our expectation.

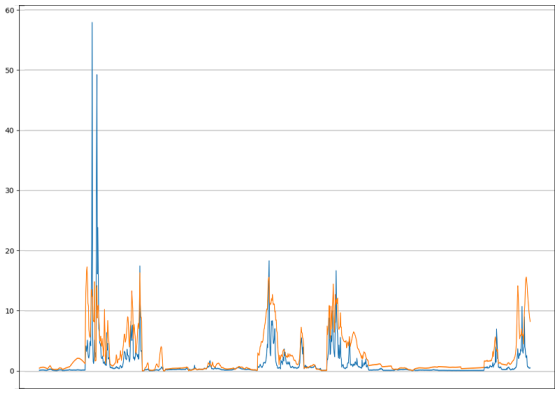


FIGURE 4.11: Relation between mean flow (orange) and AEE (blue).

Figure 4.12 shows the relation between the AEE and the average values of the diffusivity on the edges of the 8×8 grid of blocking artifacts for a sample of frames from the Sintel data set. Again, the values are scaled to make comparison easier. They seem to correlate as well.

The idea now is to make the number of time steps variable and dependent of the average absolute flow or the diffusivity. First, we choose the flow as a measure for the number of time steps. If u is the flow with size $W \times H$, we can calculate the number of time steps as

$$t(u) = \left\lfloor \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \left\| u_{i,j} \right\|_2 \right\rfloor \cdot \tau_{\text{scale}}. \tag{4.4}$$

We therefore calculate the arithmetic mean of the absolute values of all vectors in the flow and round it to a whole number. In this case, we use the floor function $\lfloor \cdot \rfloor$ for rounding. Note that the absolute value of the vectors refers to their Euclidean norm $\|\cdot\|_2$, so in our case the length of the vectors. Finally, we can scale the result by a factor τ_{scale} to gain more control. Of course, it is possible to transform the absolute flow values differently as well. In fact, our tests have shown that even better results are obtained if the rounded arithmetic mean is squared additionally.

A similar transformation can be performed using the diffusivity. However, the diffusivity values of different frames are pretty similar, which makes a useful transformation to the number of time steps difficult. We will therefore not discuss this in more detail.

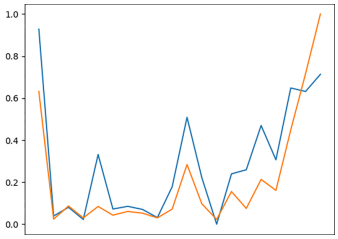


FIGURE 4.12: Relation between AEE (orange) and diffusivity values on 8×8 block grid edges (blue, scaled).

4.4 Parameter Overview

In the previous chapters, we presented a variety of different parameters that allow influencing the diffusion process. The following list provides an overview and briefly explains their respective function.

t	Number of time steps, i.e. how often diffusion is applied.
τ	Time step size per diffusion step. In practice, the stencil is convolved with the image in each time step and the result weighted with τ is subtracted from the result of the previous time step.
σ	Standard deviation of the Gaussian kernel that is applied to the input image to enable more reliable edge detection.
ε	Contrast parameter for the diffusivity g.
α, β	Weights used in the numerical diffusion scheme.
t_{RI}	Threshold parameter used in the Remove and Interpolate method. Based on this threshold, the diffusivity is used to decide whether an edge is interpolated or not. For a value of zero, all edges are removed; for a value of one, none are removed.
t_{TM}	Threshold parameter for Template Matching. A small value means that matches with low correlation are accepted as well and therefore more templates are matched. A value of one means that no templates are matched.
τ_{scale}	A scaling parameter for the variable time step size.

5 Evaluation

In Chapter 3 and Chapter 4, we developed a framework for applying diffusion and presented several approaches to reducing errors in the form of blocking artifacts such as the one arising in the RAFT method. The purpose of this chapter is to present the evaluation of our methods and compare their effectiveness. To this end, we first present the data set on which the evaluation took place. We will then evaluate the results mainly using the difference in average endpoint error between our methods and the results of RAFT.

5.1 Data Set

We evaluate the implemented methods on the MPI Sintel[1] data set [But+12]. It consists of 23 sequences and a total of over 1000 frames that were generated from the animated open source short movie Sintel. Since these are purely computer-generated images, the actual ground truth optical flow is available as well. We apply our methods to the final Sintel results of RAFT trained on FlyingChairs [Dos+15] and FlyingThings [May+16] with data set specific finetuning. So these are the results generated by an overfitted RAFT and we abbreviate them as CTS. Later, our results for the data set without finetuning are also presented, which are abbreviated with CT. In the Sintel data set, two additional variants are offered, namely the clean and final pass. In the clean pass, various lighting effects such as shading, reflections and mirror effects are already applied. In the final pass, other difficulties such as motion blur and atmospheric effects are added.

For calculating the results for our various methods, we choose a sample of twenty flows for reasons of convenience and computational cost. The sample was chosen based on the frames' AEE. They are intended to mimic the distribution of the AEE of the entire data set. This should ensure that the results for the samples are representative of the entire data set, which has been confirmed in most cases of our tests.

5.2 Error Evaluation

To evaluate the developed methods, we compare the AEE of the respective frames of RAFT with the AEE after applying our methods to the RAFT results. The AEE is calculated using the ground truth of the optical flow, which is contained in the Sintel data set. It is defined as the average distance between the endpoint positions of the ground truth flow vectors and the estimated flow vectors. If we consider images of size $W \times H$ with ground truth flow u^{gt}, then the AEE of the estimated flow u^{est} is calculated as

$$\text{AEE}\left(u^{\text{gt}}, u^{\text{est}}\right) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \left\| u_{i,j}^{\text{gt}} - u_{i,j}^{\text{est}} \right\|_2 . \tag{5.1}$$

In addition, we provide the change in percentage of AEE compared to RAFT. To this end, we subtract the AEE of the respective method of the AEE of the original results of RAFT and divide by RAFT's AEE. So if u^{est} is the resulting flow of our method and u^{RAFT} is that of RAFT,

[1] http://sintel.is.tue.mpg.de/

then the change is given by

$$\delta_{\mathrm{AEE}}\left(u^{\mathrm{gt}}, u^{\mathrm{est}}, u^{\mathrm{RAFT}}\right) = \frac{\mathrm{AEE}\left(u^{\mathrm{gt}}, u^{\mathrm{RAFT}}\right) - \mathrm{AEE}\left(u^{\mathrm{gt}}, u^{\mathrm{est}}\right)}{\mathrm{AEE}\left(u^{\mathrm{gt}}, u^{\mathrm{RAFT}}\right)}. \tag{5.2}$$

Note that a positive value for δ_{AEE} corresponds to a decrease of the AEE of our method compared to the one of RAFT, so our goal is to maximize this value. In contrast, we want to minimize the AEE.

5.3 Results

Parameter Optimization

The summary in Section 4.4 shows a variety of parameters. When computing the results, we cannot optimize all of these parameters, especially since this would not be possible in a real world application. Therefore, we simplify the parameter space by specifying some values in advance. The large dependence of anisotropic diffusion on several parameters such as the gradient threshold parameter and the number of time steps is also addressed by Tsiotsios and Petrou [TP13].

The number of time steps depends strongly on the respective method and is therefore not fixed. However, the time step size per diffusion step is set to $\tau = 0.05$. This value guarantees stability, since it is smaller than 0.25 (see [WWW13] for details), and allows a fine resolution of the time steps. The standard deviation σ is fixed as well, since optimizations on several different methods have mostly led to similar optima for σ. This optimum is approximately $\sigma = 0.1$. The same applies to the contrast parameter of the diffusivity. Its optimum is usually around $\lambda = 20$. The fact that these values attain similar optima for different methods is not surprising, since it is mainly the edge detection that is optimized which is used similarly for all methods.

For the weights of the numerical scheme, we compared the variants presented by Weickert et al. in Table 2 of [WWW13]. The best results were obtained for their nonstandard stencil, so we set $\alpha = 0.44$ and $\beta = 0.118 \operatorname{sign}\left(b(x,y)\right)$, where b is the space-dependent diffusion tensor entry as introduced in Section 3.2. The other parameters are analyzed in more detail with the help of the results.

Figure 5.1 shows the optimization process using the contrast parameter λ of the diffusivity. The AEE has been calculated for different values of λ at regular intervals. If necessary, one could resolve the intervals even smaller, but in this case the accuracy is sufficient. A unique minimum is attained. Similar results are obtained for the other parameters.

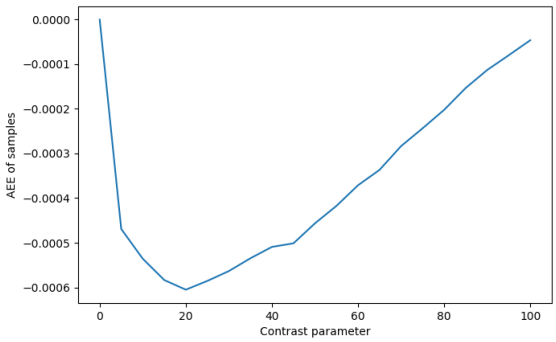


FIGURE 5.1: Optimization of the contrast parameter in the diffusivity.

Conventional Methods

To get a sense of the change in error and to examine the influence of the number of time steps, we consider the results of the three conventional methods presented in Chapter 4. These are listed for three different numbers of time steps in Table 5.1. In the table, it can be seen well that with linear diffusion the error becomes larger with increasing number of time steps. This is not surprising, since no features are retained in the flow field and information is only lost. In the Perona-Malik model, it can be seen that a slight improvement was achieved after only one time step. However, if more time steps are applied, the error quickly increases again. With the more sophisticated anisotropic diffusion, the error can actually already be improved quite a bit, especially considering that no adaptation to the RAFT method has yet been applied. Therefore, diffusion in general seems to be able to have a positive influence on the error. However, the improvement is still quite small.

A close-up of the respective errors of some of our methods applied to a sample frame can be seen in Figure 5.3. There you can see how the methods change the error and if blocking artifacts are still visible. Figure 5.3a shows the initial error of the RAFT method, in which the blocking artifacts are clearly visible. Linear diffusion reduces these artifacts (Figure 5.3b), but the image and especially edges become brighter and the error therefore increases. Figure 5.3c shows the same for anisotropic diffusion, which preserves edges better, but does not reduce blocking artifacts significantly.

Method	Time steps	AEE	δ_{AEE} (%)
Original	($t = 0$)	0.71382	0.000
Linear	($t = 1$)	0.71535	-0.215
	($t = 5$)	0.72250	-1.217
	($t = 10$)	0.73108	-2.418
Perona-Malik	($t = 1$)	0.71373	+0.012
	($t = 5$)	0.71391	-0.014
	($t = 10$)	0.71431	-0.069
Anisotropic	($t = 1$)	0.71372	+0.014
	($t = 5$)	**0.71348**	**+0.047**
	($t = 10$)	0.71352	+0.040

TABLE 5.1: Results of conventional diffusion methods. Bold indicates the largest improvement of the error.

We now want to use the best value of the anisotropic diffusion to find the optimal variant for calculating the diffusivity. To do this, we compare the three variants we presented in Section 4.1. The results can be seen in Table 5.2, where the optimal number of time steps was determined individually for each diffusivity function. It is quite clear that the Perona-Malik diffusivity is superior to the other two.

Diffusivity	AEE	δ_{AEE} (%)
Regularized Linear (g_{RL})	0.71381	+0.001
Charbonnier (g_{CH})	0.71358	+0.033
Perona-Malik (g_{PM})	**0.71348**	**+0.047**

TABLE 5.2: Results of different diffusivities on anisotropic diffusion.

RAFT Adapted Methods

We will now compare the results of our adapted methods *DDC, RI* and *TM* as well as combinations of those. We also present the results using a variable number of time steps. Finally, we verify whether the results of the sample data set can be generalized to the whole Sintel data set.

Diffusion Direction Customization Table 5.3 shows the results of DDC for different numbers of time steps. By choosing custom eigenvectors of the diffusion tensor, the standard anisotropic diffusion could be improved as can be seen from the value marked in bold, albeit only slightly. We can clearly see that the AEE increases when applying more time steps after the minimum is reached, which is in line with our expectation.

Time steps	AEE	δ_{AEE} (%)
($t = 0$)	0.71382	0.000
($t = 1$)	0.71372	+0.013
($t = 5$)	0.71351	+0.042
($t = 10$)	0.71344	+0.054
($t = 15$)	**0.71341**	**+0.056**
($t = 20$)	0.71344	+0.054
($t = 50$)	0.71362	+0.028

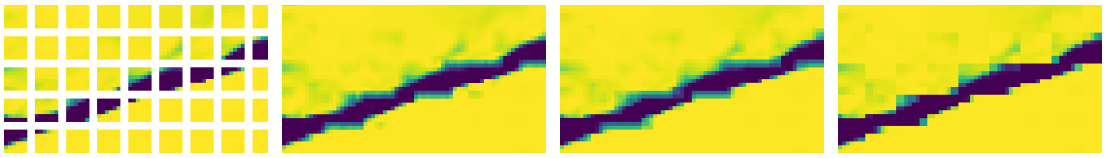
TABLE 5.3: Results of the Diffusion Direction Customization.

Remove and Interpolate The results of the RI method in Table 5.4 on the other hand do not look quite as promising. We see a significant increase in the error when applying the method to the optical flow itself. Note that no diffusion was used in this process. However, this result is not surprising since we remove all values on the 8×8 grid without considering features in the flow.

If we apply the method to the diffusivity as a refinement for anisotropic diffusion, a slight improvement of the error is possible, especially using the "nearest" interpolation method. To better understand why it is the simplest of the three interpolation methods that yields the best results, consider Figure 5.2. There you can see nicely that cubic and linear interpolation create a smooth transition at locations where edges between large and small diffusivity have been removed. The nearest interpolation, on the other hand, produces sharp edges. This better resembles the original sharp edges that the diffusivity usually produces in such areas. Figure 5.3d shows the error of this variant. Notice that the edges of the grid and thus the blocking artifacts are slightly smoothed, but the artifacts are still clearly visible.

RI applied to...	Interpolation method	AEE	δ_{AEE} (%)
Optical Flow	cubic	0.72312	-1.304
	linear	0.72444	-1.488
	nearest	**0.72290**	**-1.273**
Diffusivity	cubic	0.71376	+0.008
	linear	0.71401	-0.027
	nearest	**0.71371**	**+0.014**

TABLE 5.4: Results of the Remove and Interpolate method comparing different interpolation methods.



(A) Diffusivity with removed grid in white. (B) Cubic interpolation. (C) Linear interpolation. (D) Nearest interpolation.

FIGURE 5.2: Comparison of different interpolation methods applied to the diffusivity.

If we take into account the values of the diffusivity in the RI method, then we can improve the results slightly. This is shown in Table 5.5. The threshold controls how large the diffusivity must be to preserve edges. A threshold of 0 would correspond to the values in Table 5.4, as all edges on the 8×8 grid would be removed. With increasing threshold, the error decreases for both variants. A threshold of 1 implies that no values are removed and interpolated, since the diffusivity can take at maximum the value 1. This means that for a threshold of 1 and thus the best values in the table, the RI method is not applied at all. Consequently, this method only leads to a deterioration of the error when applied to the optical flow directly and also when used as a preliminary step of anisotropic diffusion.

RI applied to...	Threshold t_{RI}	AEE	δ_{AEE} (%)
Optical Flow (nearest)	0.1	0.71498	-0.164
	0.5	0.71467	-0.120
	0.9	0.71445	-0.088
	(1.0)	**(0.71382)**	**(0.000)**
Diffusivity (nearest)	0.1	0.71354	+0.039
	0.5	0.71351	+0.043
	0.9	0.71350	+0.044
	(1.0)	**(0.71348)**	**(+0.047)**

TABLE 5.5: Results of the Remove and Interpolate method keeping values with low diffusivity.

Template Matching By being a bit more selective in choosing edges to remove using TM, it is possible to improve the AEE further. This can be seen from the bold marked result in Table 5.6. Note that the threshold parameter of $t_{\text{TM}} = 0.8$ is nearly the optimal value and not chosen at random. Of course, for a threshold of $t_{\text{TM}} = 1.0$, no match can be found because the correlation coefficient is normalized and thus cannot be greater than 1 and we get the same result as with anisotropic diffusion. Additionally, the average number of matches per flow is given in the table. At the optimal value, these are around 160. To give some context to this value: The images underlying the flows have dimensions of 1024×436 pixels. So, in total, there can be approximately $\frac{1,024}{8} \cdot \frac{436}{8} \approx 7,000$ blocking artifacts in the image. Since the number of matches is calculated for each of the eleven templates individually and the total number is then the sum of these, a maximum of $11 \cdot 7,000 = 77,000$ matches could be found in total. Hence, although only few matches are found, there is already an improvement. This is not clearly visible in Figure 5.3e, where the edges of objects are well preserved, but the blocking artifacts are still visible. But this is different when applying the following method.

Threshold t_{TM}	AEE	δ_{AEE} (%)	Average Matched Templates
0.1	0.71371	+0.015	7133
0.5	0.71366	+0.022	2275
0.8	**0.71340**	**+0.058**	**160**
1.0	0.71348	+0.047	0

TABLE 5.6: Results of Template Matching.

Variable Time Steps The best method so far has been DDC combined with TM as a preliminary step. This variant can be further refined with the help of a variable number of time steps, as can be seen in Table 5.7. The time step scaler τ_{scale} is always chosen nearly optimal. The best variant is clearly choosing the time steps proportional to the square of the arithmetic mean of the absolute flow values, whereas using the diffusivity did not perform as well in several trials. Once again, a noticeable improvement of the AEE could be achieved. This can also be seen in the error representation in Figure 5.3f. The blocking artifacts are significantly reduced and the image is slightly darker overall, what corresponds to a reduction in error.

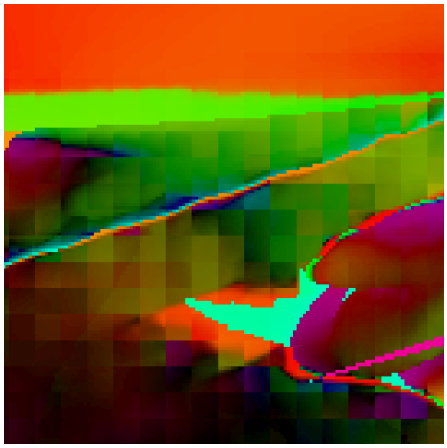
Time steps proportional to...	Time step scaler τ_{scale}	AEE	δ_{AEE} (%)
Mean of absolute flow	8	0.71328	+0.075
Squared mean of absolute flow	1	**0.71308**	**+0.103**
Diffusivity on block grid	125	0.71344	+0.053

TABLE 5.7: Results of Template Matching comparing variants of a variable number of time steps.

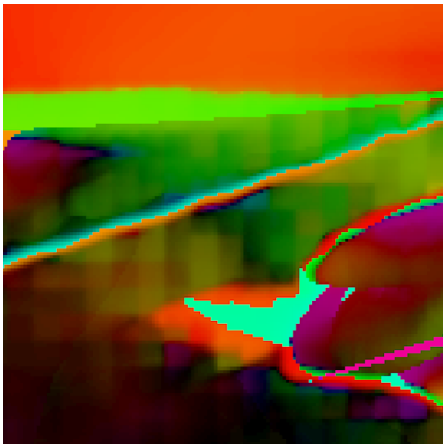
Generalization The previous results were all referring to the sample of 20 frames of the data set. To verify whether these results generalize to the entire Sintel data set, we calculated the error values for two different variants of RAFT, namely RAFT CT without finetuning of RAFT on Sintel and CTS with finetuning on Sintel, applied to the clean and final passes of Sintel and also averaged the four variants. The five results are presented in Table 5.8. Our sample frames are part of RAFT CTS final, and although the overall improvement in error is slightly less than for the sample, the average error of all frames could be improved as well. The other variants also show a reduction in error, albeit a slightly smaller one.

Data set		AEE	δ_{AEE} (%)
RAFT CT clean	Original	1.43041	
	Diffused	1.43015	+0.018
RAFT CT final	Original	2.71342	
	Diffused	2.71211	+0.049
RAFT CTS clean	Original	0.76768	
	Diffused	0.76764	+0.006
RAFT CTS final	Original	1.21736	
	Diffused	1.21632	+0.085
Whole Sintel data set	Original	1.53222	
	Diffused	1.53155	+0.043

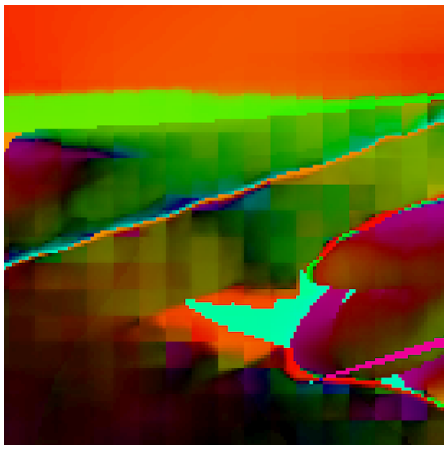
TABLE 5.8: Results of Template Matching with variable number of time steps.



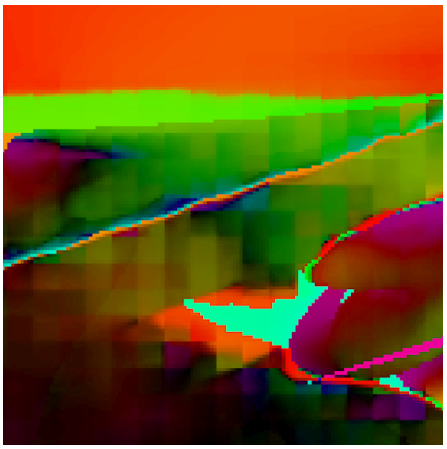
(A) Original error of RAFT without any methods applied.



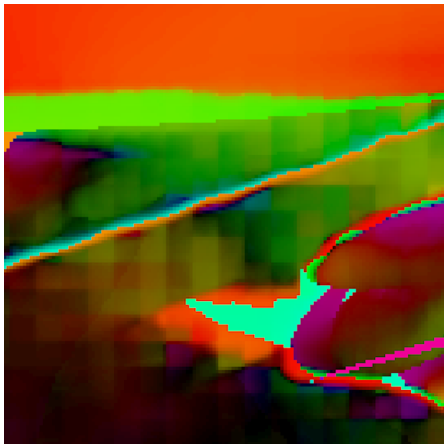
(B) Linear diffusion applied.



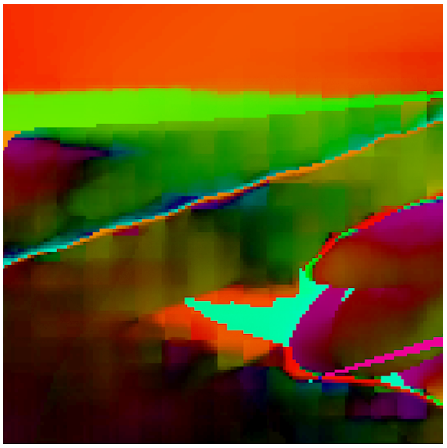
(C) Anisotropic edge-enhancing diffusion applied.



(D) Remove and Interpolate applied.



(E) Template Matching applied.



(F) Template Matching with variable time steps applied.

FIGURE 5.3: Close-up of the error of a frame of the Sintel data set in the form of a color representation of the difference between the respective calculated optical flows and the ground truth optical flow. Dark values represent a smaller error, bright values a larger error.

6 Conclusion

This thesis deals with the refinement of optical flow using diffusion-based methods. More precisely, the results of one of the best methods for computing the optical flow of an image sequence, namely RAFT, are examined for a systematic error and improved primarily with respect to this error. We want to emphasize that in all the methods, only the resulting optical flow was used as an input and therefore no information was derived from the original images themselves.

The analysis of the error that occurs in the RAFT method revealed that a regular error occurs in the form of blocking artifacts in the flow. Based on this, we have developed several approaches to reduce this error using diffusion. Since these approaches differ mainly in the choice of the diffusion tensor or its eigenvalues and eigenvectors, we have developed a framework to simplify adapting the diffusion tensor and also apply the diffusion afterwards. A special feature is that we can not only diffuse regular images, but the framework also works for diffusing optical flow. Parts of the framework are therefore adapted specifically to vector-valued images.

We found that, while conventional diffusion methods have partly already achieved slight improvements on the Sintel data set, methods adapted to the analyzed error perform even better. We first chose the diffusion tensor itself to obtain diffusion over edges of blocking artifacts. In addition, we specifically detected blocking artifacts in the image and diffused primarily at these locations. To have more control over the number of diffusion steps, the time steps can be chosen individually for each flow. A dependence of time steps on the average absolute flow has proven to be useful.

Our results have shown that there is a systematic error in RAFT and that it can be reduced using only the resulting optical flow as input and by applying diffusion. A reduction in AEE was achieved on the entire Sintel data set. However, only minor improvements were achieved. This is probably due to the fact that the error caused by blocking artifacts is small as well. The optimization of a large number of parameters has posed some difficulties. Further approaches that could lead to a greater improvement of the error will be discussed in the next section.

6.1 Outlook

Although these initial implementations of diffusion-based refinement methods did not lead to a significant reduction of the error, the methods showed that further development could still lead to general improvements of the RAFT method. We present a some possible approaches in the following.

Additional Input Data

The underlying premise of the methods presented is that they consider only a single optical flow. Advantages can result from additionally considering the flows of the following frames or the underlying image data itself. Generally using more input data might help to determine whether a blocking artifact in one frame still occurs in the following frame to distinguish which edges result from blocking artifacts and which are object edges. For example, edges of

blocking artifacts will certainly remain on the regular grid in the flow, while edges of objects visible in the optical flow are likely to move.

Consideration of Image Structures

In Chapter 1, more sophisticated methods for detecting structures in images have already been mentioned, such as detecting dominant directions in an image using a structure tensor. When considering the color representative of the direction of largest change in Figure 4.3, the blocking artifacts become clearly visible. Thus, a further advancement of our methods could be to take these directions into account when determining the diffusion tensor. An example would be choosing the eigenvectors to direct diffusion at edges of blocking artifacts in the direction of motion in the area surrounding the edges.

Blocking Artifact Detection

To detect blocking artifacts, we mainly consider the diffusivity in our methods and apply Template Matching to it. There are more advanced methods for pattern recognition in images that may produce more accurate results. A better method for detecting unwanted edges could additionally incorporate other quantities such as the direction of the largest change.

A Appendix

A.1 Diffusion Stencil

In Section 3.2 we derived a numerical scheme for the diffusion process. This results in a 3×3-stencil that can be convolved with an image to approximate the diffusion process $\operatorname{div}(D \cdot \nabla u)$. For the sake of clarity, we introduce the following notations.

The diffusion tensor D is defined as

$$D = \begin{pmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{pmatrix},$$

where we will only write a, b and c for its entries. For the discretization, we require these entries as well as the parameters α and β at positions $\left(i+\frac{1}{2}, j+\frac{1}{2}\right)$. The respective positions are denoted by the index of the parameter. For readability, we write only (i,j) for $\left(i+\frac{1}{2}, j+\frac{1}{2}\right)$ and accordingly for all other positions. We obtain these intermediate values by simply averaging over the four surrounding pixels.

To account for the boundary values, one can implement the indicator function χ in the discretization to mimic Neumann boundary conditions. Here, $\chi_{i,j}$ takes the value 0 if i or j are outside the image domain and 1 otherwise. This allows us to write the stencil entries as follows.

Stencil weight for $u_{i+1,j+1}$:

$$-\frac{\alpha_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot c_{i,j}}{h^2} - \frac{\alpha_{i,j}\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot a_{i,j}}{h^2} - \frac{0.5\cdot\beta_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2}$$
$$-\frac{0.5\cdot\beta_{i,j}\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2} - \frac{0.5\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2} - \frac{0.5\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2}$$

Stencil weight for $u_{i+1,j}$:

$$\frac{\alpha_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot c_{i,j-1}}{h^2} + \frac{\alpha_{i,j-1}\cdot\chi_{i+1,j}^2\cdot a_{i,j-1}}{h^2} + \frac{\alpha_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot c_{i,j}}{h^2}$$
$$+\frac{\alpha_{i,j}\cdot\chi_{i+1,j}^2\cdot a_{i,j}}{h^2} + \frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2} + \frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i+1,j}^2\cdot b_{i,j-1}}{h^2}$$
$$+\frac{0.5\cdot\beta_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2} + \frac{0.5\cdot\beta_{i,j}\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2} + \frac{0.5\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2} - \frac{\chi_{i+1,j}^2\cdot a_{i,j-1}}{h^2} - \frac{\chi_{i+1,j}^2\cdot a_{i,j}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i+1,j}^2\cdot b_{i,j-1}}{h^2} + \frac{0.5\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2}$$

Stencil weight for $u_{i+1,j-1}$:

$$-\frac{\alpha_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot c_{i,j-1}}{h^2} - \frac{\alpha_{i,j-1}\cdot\chi_{i+1,j-1}\cdot\chi_{i+1,j}\cdot a_{i,j-1}}{h^2} - \frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j-1}\cdot b_{i,j-1}}{h^2}$$

$$-\frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i+1,j}^2\cdot b_{i,j-1}}{h^2}+\frac{0.5\cdot\chi_{i,j}\cdot\chi_{i+1,j-1}\cdot b_{i,j-1}}{h^2}+\frac{0.5\cdot\chi_{i+1,j}^2\cdot b_{i,j-1}}{h^2}$$

Stencil weight for $u_{i,j+1}$:

$$\frac{\alpha_{i-1,j}\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot a_{i-1,j}}{h^2}+\frac{\alpha_{i-1,j}\cdot\chi_{i,j+1}^2\cdot c_{i-1,j}}{h^2}+\frac{\alpha_{i,j}\cdot\chi_{i,j+1}^2\cdot c_{i,j}}{h^2}$$
$$+\frac{\alpha_{i,j}\cdot\chi_{i+1,j}\cdot\chi_{i+1,j+1}\cdot a_{i,j}}{h^2}+\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}+\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i,j+1}^2\cdot b_{i-1,j}}{h^2}$$
$$+\frac{0.5\cdot\beta_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2}+\frac{0.5\cdot\beta_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2}+\frac{0.5\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j+1}^2\cdot b_{i-1,j}}{h^2}-\frac{\chi_{i,j+1}^2\cdot c_{i-1,j}}{h^2}-\frac{\chi_{i,j+1}^2\cdot c_{i,j}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2}+\frac{0.5\cdot\chi_{i,j+1}\cdot\chi_{i+1,j+1}\cdot b_{i,j}}{h^2}$$

Stencil weight for $u_{i,j}$:

$$-\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j}^2\cdot a_{i-1,j-1}}{h^2}-\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j}^2\cdot c_{i-1,j-1}}{h^2}-\frac{\alpha_{i-1,j}\cdot\chi_{i,j}^2\cdot a_{i-1,j}}{h^2}$$
$$-\frac{\alpha_{i-1,j}\cdot\chi_{i,j+1}^2\cdot c_{i-1,j}}{h^2}-\frac{\alpha_{i,j-1}\cdot\chi_{i,j}^2\cdot c_{i,j-1}}{h^2}-\frac{\alpha_{i,j-1}\cdot\chi_{i+1,j}^2\cdot a_{i,j-1}}{h^2}$$
$$-\frac{\alpha_{i,j}\cdot\chi_{i,j+1}^2\cdot c_{i,j}}{h^2}-\frac{\alpha_{i,j}\cdot\chi_{i+1,j}^2\cdot a_{i,j}}{h^2}-\frac{\beta_{i-1,j-1}\cdot\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}$$
$$-\frac{\beta_{i-1,j}\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}-\frac{\beta_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2}-\frac{\beta_{i,j}\cdot\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2}$$
$$+\frac{\chi_{i,j}^2\cdot a_{i-1,j-1}}{h^2}+\frac{\chi_{i,j}^2\cdot a_{i-1,j}}{h^2}+\frac{\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}$$
$$+\frac{\chi_{i,j}^2\cdot c_{i-1,j-1}}{h^2}+\frac{\chi_{i,j}^2\cdot c_{i,j-1}}{h^2}-\frac{\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}$$
$$-\frac{\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2}+\frac{\chi_{i,j+1}^2\cdot c_{i-1,j}}{h^2}+\frac{\chi_{i,j+1}^2\cdot c_{i,j}}{h^2}$$
$$+\frac{\chi_{i,j+1}\cdot\chi_{i+1,j}\cdot b_{i,j}}{h^2}+\frac{\chi_{i+1,j}^2\cdot a_{i,j-1}}{h^2}+\frac{\chi_{i+1,j}^2\cdot a_{i,j}}{h^2}$$

Stencil weight for $u_{i,j-1}$:

$$\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot a_{i-1,j-1}}{h^2}+\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j}^2\cdot c_{i-1,j-1}}{h^2}+\frac{\alpha_{i,j-1}\cdot\chi_{i,j}^2\cdot c_{i,j-1}}{h^2}$$
$$+\frac{\alpha_{i,j-1}\cdot\chi_{i+1,j-1}\cdot\chi_{i+1,j}\cdot a_{i,j-1}}{h^2}+\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}+\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}$$
$$+\frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j-1}\cdot b_{i,j-1}}{h^2}+\frac{0.5\cdot\beta_{i,j-1}\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2}+\frac{0.5\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}-\frac{\chi_{i,j}^2\cdot c_{i-1,j-1}}{h^2}-\frac{\chi_{i,j}^2\cdot c_{i,j-1}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j}\cdot\chi_{i+1,j-1}\cdot b_{i,j-1}}{h^2}+\frac{0.5\cdot\chi_{i,j}\cdot\chi_{i+1,j}\cdot b_{i,j-1}}{h^2}$$

Stencil weight for $u_{i-1,j+1}$:

$$-\frac{\alpha_{i-1,j}\cdot\chi_{i-1,j+1}\cdot\chi_{i,j+1}\cdot c_{i-1,j}}{h^2}-\frac{\alpha_{i-1,j}\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot a_{i-1,j}}{h^2}-\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i-1,j+1}\cdot\chi_{i,j}\cdot b_{i-1,j}}{h^2}$$
$$-\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i,j+1}^2\cdot b_{i-1,j}}{h^2}+\frac{0.5\cdot\chi_{i-1,j+1}\cdot\chi_{i,j}\cdot b_{i-1,j}}{h^2}+\frac{0.5\cdot\chi_{i,j+1}^2\cdot b_{i-1,j}}{h^2}$$

Stencil weight for $u_{i-1,j}$:

$$\frac{\alpha_{i-1,j-1}\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot c_{i-1,j-1}}{h^2}+\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j}^2\cdot a_{i-1,j-1}}{h^2}+\frac{\alpha_{i-1,j}\cdot\chi_{i-1,j+1}\cdot\chi_{i,j+1}\cdot c_{i-1,j}}{h^2}$$
$$+\frac{\alpha_{i-1,j}\cdot\chi_{i,j}^2\cdot a_{i-1,j}}{h^2}+\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}+\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}$$
$$+\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i-1,j+1}\cdot\chi_{i,j}\cdot b_{i-1,j}}{h^2}+\frac{0.5\cdot\beta_{i-1,j}\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}+\frac{0.5\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i-1,j+1}\cdot\chi_{i,j}\cdot b_{i-1,j}}{h^2}-\frac{\chi_{i,j}^2\cdot a_{i-1,j-1}}{h^2}-\frac{\chi_{i,j}^2\cdot a_{i-1,j}}{h^2}$$
$$-\frac{0.5\cdot\chi_{i,j}^2\cdot b_{i-1,j-1}}{h^2}+\frac{0.5\cdot\chi_{i,j}\cdot\chi_{i,j+1}\cdot b_{i-1,j}}{h^2}$$

Stencil weight for $u_{i-1,j-1}$:

$$-\frac{\alpha_{i-1,j-1}\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot c_{i-1,j-1}}{h^2}-\frac{\alpha_{i-1,j-1}\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot a_{i-1,j-1}}{h^2}-\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}$$
$$-\frac{0.5\cdot\beta_{i-1,j-1}\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}-\frac{0.5\cdot\chi_{i-1,j}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}-\frac{0.5\cdot\chi_{i,j-1}\cdot\chi_{i,j}\cdot b_{i-1,j-1}}{h^2}$$

Bibliography

[BB95] S. S. Beauchemin and J. L. Barron. “The computation of optical flow”. In: *ACM Computing Surveys* 27.3 (1995), pp. 433–466.

[BF+95] L. Blanc-Feraud, P. Charbonnier, G. Aubert, and M. Barlaud. “Nonlinear image processing: modeling and fast algorithm for regularization with edge detection”. In: *Proc. International Conference on Image Processing*. Vol. 1. IEEE, 1995, 474–477 vol.1.

[Bla+98] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger. “Robust anisotropic diffusion”. In: *IEEE Transactions on Image Processing* 7.3 (1998), pp. 421–432.

[Bro+04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. “High accuracy optical flow estimation based on a theory for warping”. In: *Proc. European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer, 2004.

[But+12] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. “A naturalistic open source movie for optical flow evaluation”. In: *Proc. European Conf. on Computer Vision (ECCV)*. Part IV, LNCS 7577. Springer-Verlag, 2012, pp. 611–625.

[BWS05] A. Bruhn, J. Weickert, and C. Schnörr. “Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods”. In: *International Journal of Computer Vision* 61.3 (2005), pp. 1–21.

[Cat+92] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. “Image Selective Smoothing and Edge Detection by Nonlinear Diffusion”. In: *SIAM Journal on Numerical Analysis* 29.1 (1992), pp. 182–193.

[Cha+02] J.-Y. Chang, W.-F. Hu, M.-H. Cheng, and B.-S. Chang. “Digital image translational and rotational motion stabilization using optical flow technique”. In: *IEEE Transactions on Consumer Electronics* 48.1 (2002), pp. 108–115.

[Dos+15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. “FlowNet: Learning Optical Flow With Convolutional Networks”. In: 2015, pp. 2758–2766.

[DZ86] S. Di Zenzo. “A note on the gradient of a multi-image”. In: *Computer Vision, Graphics, and Image Processing* 33.1 (1986), pp. 116–125.

[Gei+13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. “Vision meets robotics: The KITTI dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[Gib50] James J. Gibson. *The perception of the visual world*. The perception of the visual world. Houghton Mifflin, 1950.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361.

[Guo+12] Z. Guo, J. Sun, D. Zhang, and B. Wu. “Adaptive Perona–Malik Model Based on the Variable Exponent for Image Denoising”. In: *IEEE Transactions on Image Processing* 21.3 (2012), pp. 958–967.

[Hö83] K. Höllig. “Existence of infinitely many solutions for a forward backward heat equation”. In: *Transactions of the American Mathematical Society* 278.1 (1983), pp. 299–316.

[HS81] B.K.P. Horn and B.G. Schunck. “Determining Optical Flow”. In: *Proc. Techniques and Applications of Image Understanding*. Vol. 0281. International Society for Optics and Photonics, 1981, pp. 319–331.

[Ilg+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. “FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks”. In: 2017, pp. 2462–2470.

[KK98] B. Kawohl and N. Kutev. “Maximum and comparison principle for one-dimensional anisotropic diffusion”. In: *Mathematische Annalen* 311.1 (1998), pp. 107–123.

[LK81] B. Lucas and T. Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Proc. Seventh International Joint Conference on Artificial Intelligence*. Vancouver, Canada, 1981.

[Mae+96] Y. Mae, Y. Shirai, J. Miura, and Y. Kuno. “Object tracking in cluttered background based on optical flow and edges”. In: *Proc. of 13th International Conference on Pattern Recognition*. Vol. 1. 1996, 196–200 vol.1.

[May+16] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”. In: 2016, pp. 4040–4048.

[MG15] M. Menze and A. Geiger. “Object scene flow for autonomous vehicles”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3061–3070.

[MSB17] D. Maurer, M. Stoll, and A. Bruhn. “Order-Adaptive and Illumination-Aware Variational Optical Flow Refinement”. In: *Proc. of the British Machine Vision Conference 2017*. British Machine Vision Association, 2017, p. 150.

[NFD97] M. Nielsen, L. Florack, and R. Deriche. “Regularization, Scale-Space, and Edge Detection Filters”. In: *Journal of Mathematical Imaging and Vision* 7.4 (1997), pp. 291–307.

[PM90] P. Perona and J. Malik. “Scale-space and edge detection using anisotropic diffusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.7 (1990), pp. 629–639.

[PSM94] P. Perona, T. Shiota, and J. Malik. “Anisotropic Diffusion”. In: *Geometry-Driven Diffusion in Computer Vision*. Computational Imaging and Vision. Springer Netherlands, 1994, pp. 73–92.

[RB17] A. Ranjan and M.J. Black. “Optical Flow Estimation Using a Spatial Pyramid Network”. In: 2017, pp. 4161–4170.

[Shi+05] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi. “Optical flow-based real-time object tracking using non-prior training active feature model”. In: *Real-Time Imaging*. Special Issue on Video Object Processing 11.3 (2005), pp. 204–218.

[SLZ98] Y. Q. Shi, S. Lin, and Y.-Q. Zhang. “Optical flow-based motion compensation algorithm for very low-bit-rate video coding”. In: *International Journal of Imaging Systems and Technology* 9.4 (1998), pp. 230–237.

[SR96] G. Sapiro and D.L. Ringach. “Anisotropic diffusion of multivalued images with applications to color filtering”. In: *IEEE Transactions on Image Processing* 5.11 (1996), pp. 1582–1586.

[Sun+18] D. Sun, X. Yang, M. Liu, and J. Kautz. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: 2018, pp. 8934–8943.

[Sun+20] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. “Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.6 (2020), pp. 1408–1423.

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[TD20] Z. Teed and J. Deng. “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow”. In: *Proc. Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Springer International Publishing, 2020, pp. 402–419.

[TP13] C. Tsiotsios and M. Petrou. “On the choice of the parameters for anisotropic diffusion in image processing”. In: *Pattern Recognition* 46.5 (2013), pp. 1369–1381.

[Wei+03] J. Weickert, A. Bruhn, N. Papenberg, and T. Brox. “Variational optic flow computation: From continuous models to algorithms”. In: *IWCVIA* 3 (2003), pp. 1–6.

[Wei95] J. Weickert. “Multiscale texture enhancement”. In: *Proc. Computer Analysis of Images and Patterns*. Ed. by V. Hlaváč and R. Šára. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1995, pp. 230–237.

[Wei96] J. Weickert. “Theoretical Foundations of Anisotropic Diffusion in Image Processing”. In: *Proc. Theoretical Foundations of Computer Vision*. Computing Supplement. Vienna: Springer, 1996, pp. 221–236.

[Wei99a] G.W. Wei. “Generalized Perona-Malik equation for image restoration”. In: *IEEE Signal Processing Letters* 6.7 (1999), pp. 165–167.

[Wei99b] J. Weickert. “Coherence-Enhancing Diffusion Filtering”. In: *International Journal of Computer Vision* 31.2 (1999), pp. 111–127.

[WWS06] M. Welk, J. Weickert, and G. Steidl. “From Tensor-Driven Diffusion to Anisotropic Wavelet Shrinkage”. In: *Computer Vision – ECCV 2006*. Vol. 3951. Springer Berlin Heidelberg, 2006, pp. 391–403.

[WWW13] J. Weickert, M. Welk, and M. Wickert. “L2-Stable Nonstandard Finite Differences for Anisotropic Diffusion”. In: *Proc. Scale Space and Variational Methods in Computer Vision*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 380–391.

[YA02] Y. Yu and S.T. Acton. “Speckle reducing anisotropic diffusion”. In: *IEEE Transactions on Image Processing* 11.11 (2002), pp. 1260–1270.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Druck-Exemplaren überein.

Datum und Unterschrift:

Declaration

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

Date and Signature: