

SAARLAND UNIVERSITY
FACULTY OF NATURAL SCIENCES AND TECHNOLOGY I
DEPARTMENT OF COMPUTER SCIENCE
MASTER'S PROGRAM IN COMPUTER SCIENCE

Master's Thesis

Object Tracking using Variational Optic Flow Methods

submitted by

Timo Florian Adam

August 26, 2014



UNIVERSITÄT
DES
SAARLANDES

Supervisor:

Prof. Dr. Joachim Weickert,
Mathematical Image Analysis Group

First Reviewer:

Prof. Dr. Joachim Weickert

Second Reviewer:

Prof. Dr. Antonio Krüger

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken, August 26, 2014

Jonni Adm

Statement in Lieu of an Oath

I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

Saarbrücken, August 26, 2014

A handwritten signature in blue ink, reading "Jonni Adm". The signature is written in a cursive style with a long, sweeping tail on the final letter.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Saarbrücken, August 26, 2014

Jonni Adm

Acknowledgements

I would shortly like to acknowledge those people who supported me in completing this thesis.

I thank Prof. Joachim Weickert for offering me the opportunity to work on this interesting topic and for creating a motivating working environment for his students. I also want to thank Prof. Antonio Krüger for agreeing to review my thesis as a second reviewer.

I also want to mention the excellent research on Variational Optic Flow Computation by Prof. Andrés Bruhn on which this work is based upon. I am also thanking the team of the Mathematical Image Analysis Group for their friendly support, especially Markus Mainberger and Oliver Demetz for introducing me to the topic of correspondence problems in computer vision in their lectures at Saarland University. I also thank Oliver Barth and Merlin Lang for taking the time to have some insightful discussions on the topic.

Last, but certainly not least, I thank my family for their constant and unconditional support and their encouragement.

Abstract

The estimation of motion and the tracking of objects in image sequences are two of the key problems in the field of computer vision.

We will first introduce the reader to the basic concepts of object tracking by providing some examples and laying the mathematical foundation. To be able to detect and track objects throughout an image sequence, it is vital to extract motion information from the sequence. To this end, we will consider three variational approaches for the computation of the optic flow and research their properties with regard to object tracking. Variational optic flow methods belong to the class of high-accuracy optic flow methods and offer many advantages for our particular purpose, especially when applied in the spatiotemporal setting.

Since an optic flow field only provides us with point-to-point correspondences between time instances of an image sequence, we will take a closer look at how to convert the obtained optic flow field to a higher-level representation by means of segmentation. Finally, we are going to explore some ideas on how to exploit the combination of object tracking and variational optic flow computation to evaluate the quality of an optic flow field or, beyond that, to assess if it is possible to use the information obtained from the tracking algorithm to improve the estimation of the optic flow.

In this work, we are going to assume that our input data consists of gray value image sequences acquired in the spectrum of visible light using a monocular camera setup, but most of the concepts carry over easily to multichannel- or multicamera setups.

Contents

1	Introduction	3
1.1	Motivation	3
1.2	Target Audience	3
1.3	An Introduction to Object Tracking	5
1.4	Basic Definitions	6
1.5	Target Representations	7
1.6	Difficulties and Challenges in Object Tracking	8
2	Variational Optic Flow Computation	11
2.1	Fundamentals	12
2.2	The Variational Method of Horn and Schunck	14
2.3	Variational Optic Flow for Large Displacements	20
2.4	The Spatiotemporal Method of Brox et al	24
3	Segmentation of Displacement Fields	29
3.1	Thresholding	29
3.2	3D Watershed on Vector Fields	31
3.3	Region Merging of 3D Volumes	33
4	Results	37
4.1	Tracking Score	38
4.2	Performance Evaluation of the Optic Flow Result	39
4.3	Future Work	40
4.4	Conclusions	41
4.5	Sources	41
A	Documentation of Accompanying Software	43
A.1	Contents of the Accompanying Medium	43
A.2	Sources	44
A.3	Keyboard Commands	44

List of Figures

1	Examples of object tracking in practice	4
2	Target representations for object tracking	8
3	Optic flow in practice	11
4	Visualization of vector fields using a color-coded representation	14
5	Three scenarios illustrating the aperture problem	17
6	Variational optic flow: Results of the method of Horn and Schunck	20
7	Error measure functions: The L_1 - and L_2 -Norm	21
8	Illustration of the coarse-to-fine warping scheme	22
9	Results for variational optic flow computation for large displacements	23
10	Results of the spatiotemporal method of Brox et al.	26
11	Optic flow with spatial and spatiotemporal smoothness assumption	27
12	Optic flow under illumination changes	28
13	Segmentation of displacement fields	30
14	Illustration of a segmentation using the watershed	31
15	The watershed segmentation on a gray value image	32
16	Postprocessing of the optic flow field	35
17	Tracking result (Contour representation)	37
18	Tracking result (Rectangular representation)	38
19	Tracking result (Point representation)	39
20	Evaluation of several optic flow computation results	40

List of Tables

1	Tracking scores for the three optic flow methods presented in this work	41
---	---	----

1 Introduction

1.1 Motivation

The tracking of objects in image sequences is a heavily researched topic in the realm of computer vision with many possible fields of application. The increased availability of digital video content, such as on online video platforms, and the need for automated video analysis in fields such as media, robotics or automotive industries have spurred a lot of research on this topic recently.

Starting with the development of infrared radar trackers for airspace monitoring in the early 20th century, the invention of the *Kalman filter* [Kal60] revolutionized the field of object tracking in the 1960s. The article of Grewal and Andrews [GA10] is a very nice read on the history of object tracking and its early applications in the aerospace industry and its role in space exploration. Instead of focusing on statistical filtering techniques which are especially suited for real-time tasks, this work will be concerned with higher-accuracy optic flow methods that allow to precisely track the motion of objects through a scene.

The goal of this thesis is to explore the combination of variational optic flow methods in the spatial and also spatiotemporal domain with concepts from object tracking. We will also propose a way of evaluating the quality of optic flow computation results using the trajectory information obtained through object tracking.

This section will give a concise introduction to object tracking and highlight some of the challenges in the field. In Section 2, three different variational optic flow algorithms will be introduced, accompanied by a discussion on their advantages and disadvantages in terms of applicability to object tracking. Section 3 will be concerned with segmentation and postprocessing of the optic flow field. In Section 4, we will look at how object tracking and variational optic flow interact, discuss some results and give an outlook on possible future research.

1.2 Target Audience

Readers should be familiar with general computer vision literature and basic image processing techniques as well as numerical analysis, discretization methods and the calculus of variations. We will also revert to some ideas from the fields of machine learning and information retrieval.

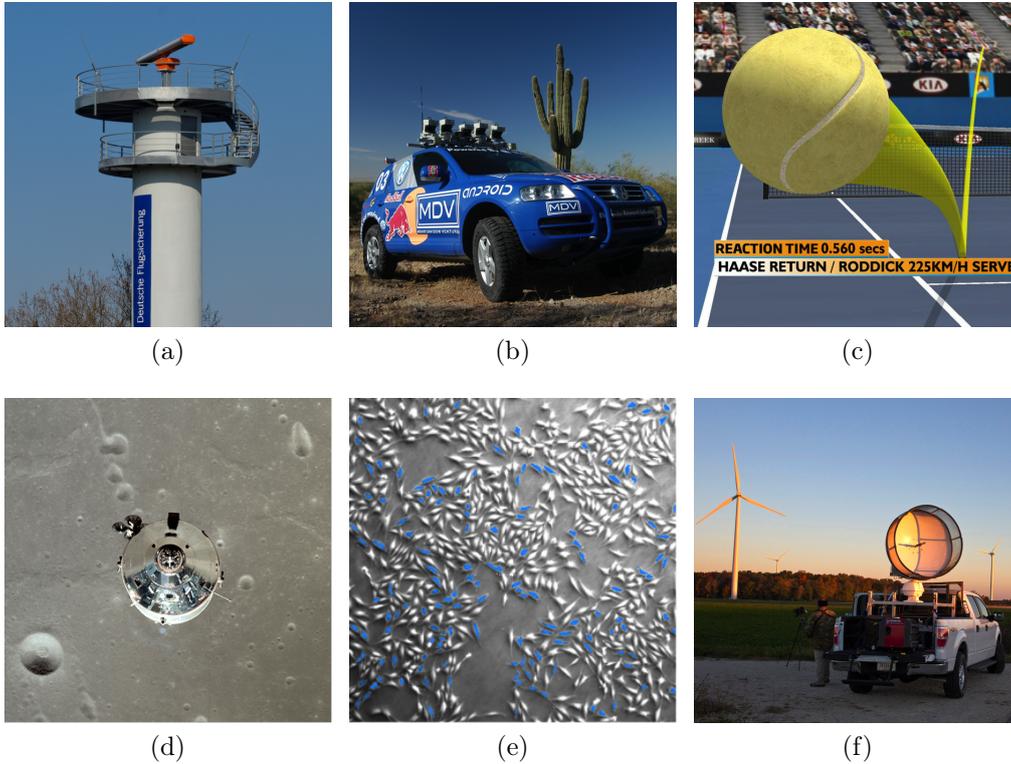


Figure 1: Examples of object tracking in practice: a) Radar tower at Frankfurt Airport: Radar measurements are used to track airplanes during approach and departure [GA10]. b) Stanley: An autonomous ground vehicle equipped with lidar and video sensors, using an unscented Kalman filter for collision avoidance. [Thr+06] c) HawkEye: Game ball tracking technology used as an officiating aid in sports competitions, such as tennis tournaments. d) Apollo 11 command module in orbit of moon: The trajectory estimations are controlled by software using Kalman filters [MS85]. Research to ensure correctness of the trajectory estimation gave rise to extended Kalman filters and Monte Carlo methods. e) Tracking of pancreatic stem cells: The paths of the cells marked in blue have been validated as correctly tracked by an algorithm over a sequence of 23.5 hours with 4 frames/hour. [Rap+11] f) Researchers set up radar equipment near Horicon National Wildlife Refuge in Eastern Wisconsin to track birds [Lar05].

Image Credit: a) Norbert Nagel. License: © ⓘ ⓘ b) Stanford Racing Team. License: educational c) Andy Ingham. License: © ⓘ ⓘ ⓘ d) NASA/JSC. License: ⓘ e) Fraunhofer Institute for Marine Biotechnology and Institute for Neuro- and Bioinformatics, University of Lübeck. License: © ⓘ ⓘ f) United States Geological Survey, J. Bartholmai. License: ⓘ

1.3 An Introduction to Object Tracking

Object tracking refers to the problem of using sensor measurements to determine the location, path and characteristics of objects of interest. Typical types of sensors that are used to acquire the input data include radar, sonar, lidar, infrared, microphone or ultrasound but can also be based on a variety of other image acquisition methods. When using regular cameras, operating in the spectrum of visible light, object tracking is also often referred to as *video tracking* in the literature.

The objectives of object tracking are the determination of the *number of objects*, their *states* (such as position, shape, velocity or acceleration), also possibly their identities or any other features that might be of interest. Judging from the vague definition, one can already tell that object tracking is a highly application-driven problem that may take various forms. Depending on the area of application, the task of object tracking is usually simplified by imposing certain constraints, such as assumptions on the motion of objects (smooth object motion, constant velocity or acceleration, prior knowledge about movement patterns), the number or size of objects or by exploiting other contextual information.

Tracking methods can be classified by the amount of interaction with the user. *Manual* and *semi-automated tracking* methods (where certain markers set by a human operator are already provided to the algorithm) are able to provide a very high accuracy but are usually expensive and time-consuming and often not applicable in real-time. An example of such a scenario are movie productions. *Automated tracking* algorithms on the other hand are in general applicable in real-time and require no interaction with the user. Instead, they rely on a priori information about targets that are fitted to the algorithm.

Applications of object tracking include weather- and airspace monitoring (Figure 1a), medical imaging applications [Sma+08], wildlife biology [RPP12], [ZR98] (Figure 1f), cell biology (Figure 1e), vehicle navigation [She+09] and driver assistance systems [LMB02], unmanned aerial vehicle navigation [Ken+09; MS85], or motion based recognition, such as automatic object detection and human identification. One of the main applications for tracking is robotics since, for being able to understand and react to the environment, it is essential for machines to have the capability to visually detect and track objects. ([GMM02], [MLB07]). Tracking methods are also used in automated surveillance systems (to detect rare events or to redirect the attention of human operators to events of interest), in business- or retail intelligence or for analyzing traffic flux to prevent congestions by redirecting traffic flow. Video indexing is also becoming a very important field of

application for tracking since there is a huge amount of video material on online platforms and in private collections and the need for tagging, annotating, understanding and retrieving video contents is becoming more and more important [AKM02; SK11]. In applications such as human computer interaction, video tracking is used for gesture recognition [LL01; HB01] or eye gaze tracking. Object tracking also has applications in the field of augmented reality, entertainment- and media productions [Yao+10; Lu+13] as well as art installations and public performances.

1.4 Basic Definitions

Objects of interest are usually referred to as *targets*. Consequently, the first appearance of an object in a sequence is referred to as *target birth*. This event occurs at image boundaries (when an object enters the field of view), at specific entry areas (doors,...), from the far-field of the camera (approaching from distance), or if a target spawns from another target. When an object leaves the sequence, this is referred to as *target death*. Consequently, this situation occurs when an objects leaves the field of view of the camera, at specific exit areas (doors,...), when it disappears at a distance or when it disappears inside another object (building,...). The identification and the management of detected objects is referred to as *track management* (or *trajectory management*).

An *image sequence* is a function $f(x, y, t)$ (also sometimes simply referred to as f from now on) from $\Omega_3 \rightarrow \mathbb{R}$ with $\Omega_3 \subset [0, n_x] \times [0, n_y] \times [0, T] \subset \mathbb{R}^3$ where $(x, y)^\top$ denote the coordinates in a spatial image domain $\Omega_2 \subset [0, n_x] \times [0, n_y] \subset \mathbb{R}^2$ and t denotes time in a certain interval $[0, T] \subset \mathbb{R}$. The domain of the function $f(x, y, t)$ is referred to as *image sequence domain*, while the co-domain is referred to as the *range*. Since we are focusing on gray value image data, high values in the co-domain correspond to a high brightness/gray value of the image, while low values correspond to low brightness/gray value.

As a first step of object tracking, the sequence is converted from the image domain Ω_2 onto a *feature space* (also called *observation space*) which is often defined as $\Omega_O \subset [0, n_x] \times [0, n_y] \subset \mathbb{R}^2$. This process is referred to as *feature extraction*. Examples of such features are the brightness, the color, edges, textures or even higher level representations of the image data such as the optic flow.

Next, the sequence is converted from the feature domain Ω_O into a *state space*, usually represented as $\Omega_S \subset [0, n_x] \times [0, n_y] \subset \mathbb{R}^2$. The co-domains of the feature- and the state spaces can take arbitrary form, depending on the model used. A vector $x_t \in \Omega_S$ represents the state of the target at time $t \in [0, T] \subset \mathbb{R}$. The sequence of states over time $\mathbf{x} = x_t : t \in [0, T]$ is called

the *trajectory*.

Object Tracking is usually performed with some predefined appearance features (either parametric or nonparametric), such as the probability density, certain predefined templates, sets of landmarks or some arbitrarily defined appearance vectors. To determine features in a scene, it is also possible to use principal- or independent component analysis or machine learning approaches such as support vector machines or Bayesian networks.

The automated tracking of objects generally consists of five main logical steps:

- The definition of a method to extract relevant information from the data (using motion classification, change detection, object classification, low-level features such as the color or the gradient or mid-level features like edges or interest points)
- The definition of a representation for encoding the appearance and the shape of a target (the state) (This step is usually a trade-off between accuracy and invariance, as will be seen in Section 1.5)
- The definition of a method to propagate the state of the target over time (a method that links different instances of the same object over time)
- The definition of a strategy to manage targets appearing and disappearing from the imaged scene (as part of the trajectory management)
- The extraction of meta-data from the state (video annotation, scene understanding, behavior recognition)

Since we are not interested in extracting any additional data besides the motion information of the objects themselves in this work, we can easily neglect the last step.

1.5 Target Representations

Objects can be represented as points or sets of points $x_t = (u_t, v_t)$ on the image plane, where x_t is typically defined as the centroid of the target object (Figure 2a). This representation is usually chosen for small objects or when the shape or size of objects is not important. Objects can also be represented as primitive geometric shapes, like rectangles $x_t = (u_t, v_t, w_t, h_t)$, where (u_t, v_t) denotes the location of the object on the image plane while w_t and h_t denote the width and height of the surrounding shape (Figure 2b). Ellipses $x_t = (u_t, v_t, w_t, h_t, \theta_t)$ are also commonly used, where the additional

parameter θ_t represents the angle of rotation. Such representation are mostly suitable for larger rigid objects whose size may be of interest.

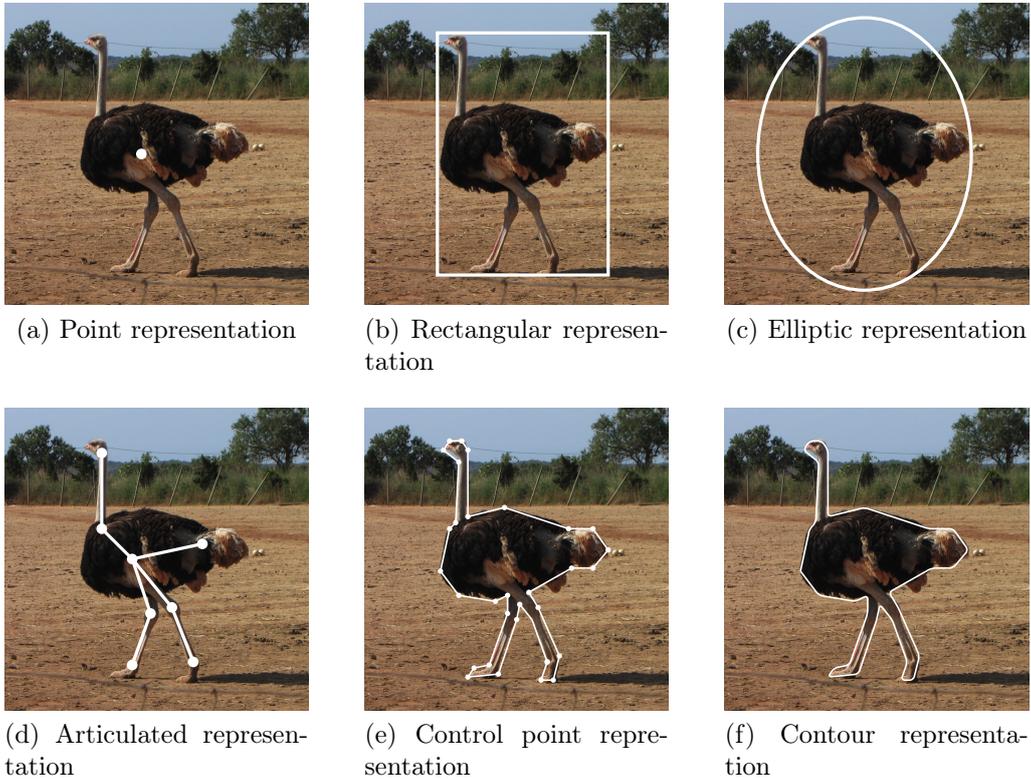


Figure 2: Common target representations for object tracking

A more expressive way to represent objects is to display their silhouette (the area of the image that is covered by the object) or contour (the boundary of the silhouette). This representation is suitable for tracking complex non-rigid shapes, as displayed in Figure 2f. In applications such as movie productions, one often uses articulated shape models, where parts of an object are put in relation through joints that connect them. The relationships between these parts are governed by kinematic motion models. Such a representation is displayed in Figure 2d. Then, there are also representations using control points (Figure 2e) or skeletal models. Such representations can be obtained by applying a medial axis transform to the object silhouette.

1.6 Difficulties and Challenges in Object Tracking

Besides the usual problems one encounters in image processing, such as noise, sampling- or compression artifacts and camera motion, one additionally has

to deal with many other problems that are specific to optic flow computation and object tracking. Such problems include abrupt or complex object motion patterns (tracking of a fly), scene illumination- or ambient illumination changes (changes in the direction, the intensity or the color of light sources), non-rigid or articulated object structures, changes in pose (legs of a person while walking) and motion patterns such as rotations, translations, deformations, scaling or divergent motion. Additional major problems are partial or full occlusions and *clutter*¹. Challenges also include multi-camera tracking, where one tries to follow an object over time by using different camera perspectives or by extracting and combining tracking data from different image acquisition sources.

¹clutter: features extracted from non-target image areas are difficult to discriminate from features that one expects the target to generate

2 Variational Optic Flow Computation

Motion is an important source of visual information. This section investigates several methods to compute motion patterns in image sequences, the *optic(al) flow*. Optic flow computation is an essential tool for tasks such as object detection, automated navigation [Enk91; ON92] and scene understanding [Wed+11], which are key problems in machine vision. In addition, it has plenty of other applications such as video compression [WKC94], particle image velocimetry [Ruh+05] and medical image registration [HW13].

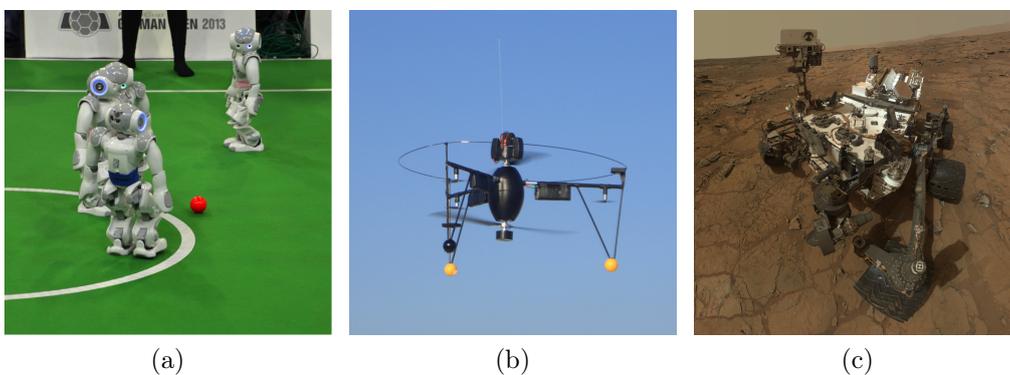


Figure 3: Optic flow in practice: a) German Robocup: Robots competing in a soccer tournament using optical flow algorithms to track the movement of the game ball. b) A miniature unmanned aerial vehicle attaining its precise position above ground using a visible light optic flow sensor directed to the ground. c) Mars rover Curiosity: Because of its distance to earth, an automated vehicle on planet Mars relies upon autonomous navigation using visual odometry. [Pap12]

Image Credit: a) Torsten Maue; License:  b) EMT Fancopter. License:  c) NASA/JPL-Caltech/MSSS. License: 

There exist lots of very different strategies for computing the optic flow, such as *feature-based methods* that try to find correspondences between characteristic features in images [Low99; WAB06; WWB88], *region- or area-based matching techniques* that try to match entire regions based on their contained information [Sin90; Ana89], *frequency-domain or phase-based approaches* that try to perform a filtering in the Fourier Domain [FJ90; Hee87] and *differential methods* that are based on formulating the problem in a variational model. Differential optic flow methods can again be classified into *local methods* such as the ones by Lukas and Kanade [LK81] and Bigün et al. [BGW91] and *global methods* like Horn and Schunck [HS81], Brox et al. [Bro+04] and Zimmer et

al. [ZBW11]. There also exist combinations between global and local differential approaches such as [BWS05]. Local methods are usually more robust under noise, but can only produce sparse flow fields, while global methods yield dense flow fields. The three methods investigated in this section belong to the class of global differential methods.

For a purpose such as object tracking, where one is interested in consistency and stability of the results over time, It can also be helpful to extend the modeling into the *spatiotemporal domain*. Black [Bla92] formulated a temporal constraint that assumes a constant acceleration of image patches over time. Weickert and Schnörr [WS01b] investigated the extension of smoothness terms into the temporal dimension while Salgado and Sánchez [SS06] present a temporal regularizer. Barron and Beauchemin [BFB94; BB95] provide a nice review on different optic flow techniques.

In the following, we are going to investigate three different *variational optic flow methods* along with their properties regarding object tracking. Variational optic flow methods currently rank among the best optic flow computation approaches available. Based on the calculus of variations, these methods are *mathematically well founded*, and allow for a *transparent and coherent modeling*, since all assumptions are incorporated into the mathematical model and can therefore easily be understood and modified. The research introduced in this section is largely based upon the excellent work on modeling and numerics of variational optic flow methods by Bruhn et al. [Bru06].

2.1 Fundamentals

The goal of *optic flow computation* is to estimate the apparent² motion patterns of an image sequence f_0 , and/or the motion of the camera itself. The desired result can mathematically be represented as a vector field (in this context also called *displacement field*, *optic(al) flow field* or *image velocity field*) $(u(x, y, t), v(x, y, t), \Delta t)^\top$ that models the displacement of objects in a sequence between points in time. When working with real world data, the image sequence is usually discretized, and one obtains a sequence of still images (also called *frames*) $(f_0(x, y, t_0), f_0(x, y, t_1), \dots)$.

For all the following methods that we are going to examine, we will assume that the image sequence has been presmoothed by means of a Gaussian

²Motion patterns must not necessarily be caused by actual objects moving through the scene, but can also be caused by e.g., moving reflection patterns on a surface caused by the movement of the sun.

convolution

$$f(x, y, t) = (K_\sigma * f_0)(x, y, t) := \int_{\Omega} K_\sigma(\tilde{x}, \tilde{y}, \tilde{t}) f_0(x - \tilde{x}, y - \tilde{y}, t - \tilde{t}) d\tilde{x} d\tilde{y} d\tilde{t} \quad (1)$$

where $K_\sigma = K_{\sigma_x} \cdot K_{\sigma_y} \cdot K_{\sigma_t}$ is the product of three one-dimensional Gaussian kernels

$$K_{\sigma_i}(i) := \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{i^2}{2\sigma_i^2}\right) \quad (2)$$

with standard deviation σ_i in all three directions. Of course, such a low-pass filtering always entails a certain loss of information, but it also removes high frequency structures such as noise and makes the method more stable and makes the function $f(x, y, t)$ infinitely many times differentiable. The standard deviation σ_i is therefore also referred to as *noise scale*.

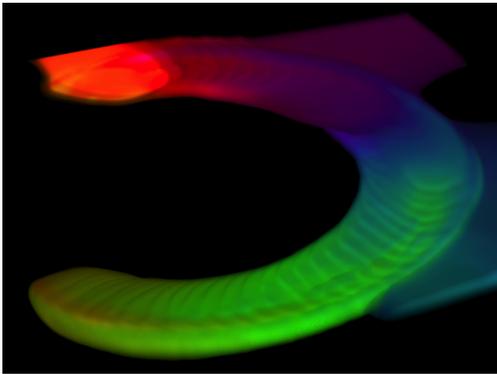
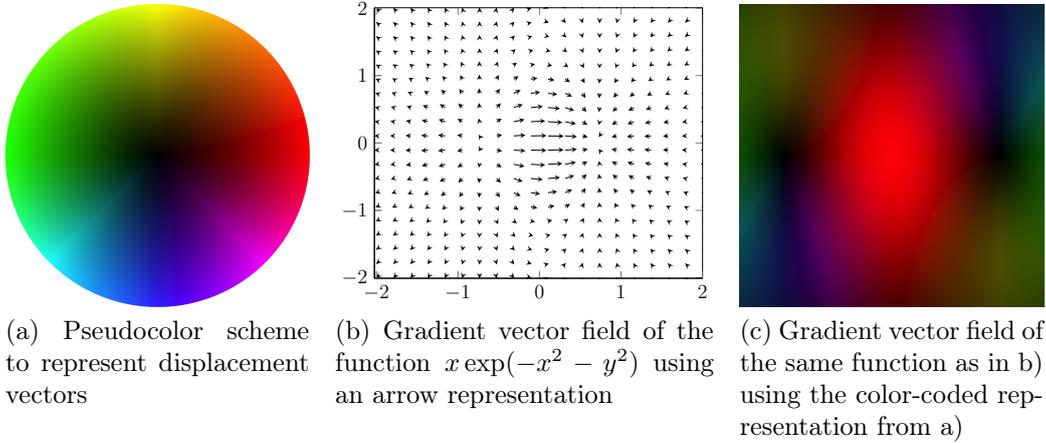
We will use the following equivalent notations when referring to (partial) derivatives.

$$\frac{\partial}{\partial x} f = \partial_x f = f_x \quad (3)$$

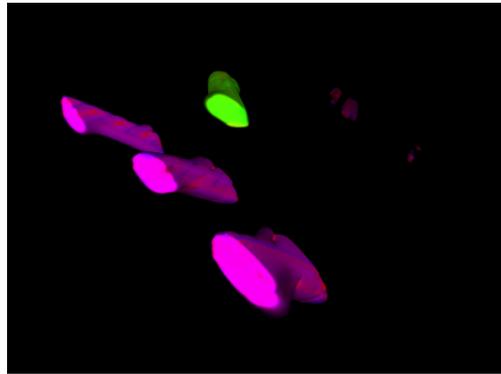
Variables in bold print such as \mathbf{x} are used as a short notation, for when a mathematical concept is applied either in a spatial ($\mathbf{x} = (x, y)$) or spatiotemporal ($\mathbf{x} = (x, y, t)$) context. Which domain the variable refers to should be clear from the context.

To represent a displacement field graphically, instead of the *arrow plot* that is commonly used in physics (also called *vector plot* or *quiver plot*), shown in Figure 4b, we are going to use a pseudocolor representation scheme from Bruhn [Bru06], where the hue represents the direction of the vector $(u(x, y, t), v(x, y, t))^\top$ at location $(x, y)^\top$ and time t , while the saturation represents its magnitude. A distinct advantage of the pseudocolor representation is that it allows to display motion information in the vector field much more precisely (especially at motion discontinuities) compared to the coarse representation of the arrow plot.

In this work we will only focus on the *monocular case* for optic flow computation (using only one camera). Therefore, we will not be able to extract any actual depth information from the scene which means that the flow field will be computed from the projection of the actual motion onto the camera plane.



(d) 3D volume representation: the color-coded displacement field of a scene depicting a ball rolling in a circle



(e) 3D volume representation: the displacement field of the sequence from Figure 10

Figure 4: Visualization of vector fields using a color-coded scheme and 3D volume models

2.2 The Variational Method of Horn and Schunck

To find correspondences between objects at different points in time of an image sequence, one needs to extract certain features from the image sequence and then search for correspondences between these features. The first feature that comes to mind is the *gray value* (also brightness, intensity). If we assume the gray value of an object to remain fairly constant over time, we are able to formulate the following constraint, also known as the *brightness constancy assumption*:

$$\begin{aligned}
 f(x, y, t) &= f(x + u, y + v, t + 1) && \Leftrightarrow \\
 0 &= f(x + u, y + v, t + 1) - f(x, y, t) && (4)
 \end{aligned}$$

From now on, for the sake of simplicity we will assume that the temporal displacement between two subsequent frames of an image sequence is one ($\Delta t = 1$).

Having the unknowns u and v in the argument of the function is very inconvenient from a mathematical point of view. By performing a linearization (using the first order term of a Taylor expansion) around point f , we can approximate the value of f at the displaced location via:

$$\begin{aligned} f(x + u, y + v, t + 1) &\approx f + \left(\begin{pmatrix} x + u \\ y + v \\ t + 1 \end{pmatrix} - \begin{pmatrix} x \\ y \\ t \end{pmatrix} \right)^\top \nabla f \\ &= f(x, y, t) + (u, v, 1) \begin{pmatrix} f_x(x, y, t) \\ f_y(x, y, t) \\ f_t(x, y, t) \end{pmatrix} \end{aligned} \tag{5}$$

Plugging the result of Equation 5 into the brightness constancy assumption, we obtain the *linearized optic flow constraint* (also *brightness constancy constraint equation* or *optic flow constraint equation*):

$$\begin{aligned} f + f_x u + f_y v + f_t - f &= 0 \quad \Leftrightarrow \\ f_x u + f_y v + f_t &= 0 \end{aligned} \tag{6}$$

The linearization of the constancy assumption is however a very severe restriction in the model, as we will see later. With this formula at hand, it is now possible to solve the equation for the unknowns u and v . Since we have only one equation with two unknowns, the system is underdetermined and there may also occur situations, where we obtain infinitely many correct solutions to Equation 6. This is referred to as the *aperture problem* [MU79], illustrated in Figure 5. If the image gradient is zero ($\nabla f = 0$), the flow cannot be determined at all, since all flow vectors are valid solutions to the brightness constancy constraint equation as seen in Figure 5 Scenario 3. In cases where the image gradient in one dimension is non-zero ($\nabla f \neq 0$), all the points that lie on the line described by equation 6 are valid solutions to our problem. The question in such a situation is, which of all these valid displacement vectors is the most descriptive one for the actual motion and which vector $(u, v)^\top$ should be accepted as a solution. To this end, we can decompose the flow vector $(u, v)^\top$ into the normal flow $(u, v)_n^\top$ and the tangential flow $(u, v)_t^\top$ by reformulating it using the basis given by the two vectors ∇f

and ∇f^\perp

$$\begin{pmatrix} u \\ v \end{pmatrix} = \underbrace{\left(\begin{pmatrix} u \\ v \end{pmatrix} \frac{\nabla f}{\|\nabla f\|} \right) \frac{\nabla f}{\|\nabla f\|}}_{\text{normal flow} =: (u,v)_n^\top} + \underbrace{\left(\begin{pmatrix} u \\ v \end{pmatrix} \frac{\nabla f^\perp}{\|\nabla f\|} \right) \frac{\nabla f^\perp}{\|\nabla f\|}}_{\text{tangential flow} =: (u,v)_t^\top} \quad (7)$$

when rewriting Equation 6 in vector notation as

$$-f_t = (u, v) \nabla f \quad (8)$$

we can now reformulate the normal flow as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix}_n = \frac{-f_t \nabla f}{\|\nabla f\| \|\nabla f\|} \quad (9)$$

Since we are not able to determine the flow uniquely, the *normal flow* at least provides us with a canonical solution which in a lot of situations gives the most meaningful result. By plugging Equation 9 into the brightness constancy constraint equation, we can verify that the normal flow is indeed a valid solution to the optic flow problem:

$$\begin{aligned} \nabla f \begin{pmatrix} u \\ v \end{pmatrix}_n + f_t &= 0 \\ \nabla f \frac{-f_t \nabla f}{\|\nabla f\| \|\nabla f\|} + f_t &= 0 \\ -f_t + f_t &= 0 \end{aligned} \quad (10)$$

Such a situation is depicted in Figure 5 Scenario 2.

To compute the unknown flow field $(u(x, y), v(x, y), 1)^\top$, Horn and Schunck use a variational ansatz. The previously introduced constraint is modeled in form of an *energy functional* of the unknown variables u and v . The goal is to penalize deviations from the model assumptions by attempting to minimize this energy.

As explained above, at many locations it is not possible to compute a flow. To still obtain a dense solution, an additional *smoothness term* is introduced. This term models the assumption, that the flow field only varies smoothly in spatial direction. This behavior is achieved by assuming homogeneous smoothness on the flow field (controlled by a weighting parameter α , referred to as *regularization parameter*, or *smoothness parameter*)³. This model now

³the larger the parameter α is chosen, the denser and smoother the flow field will become

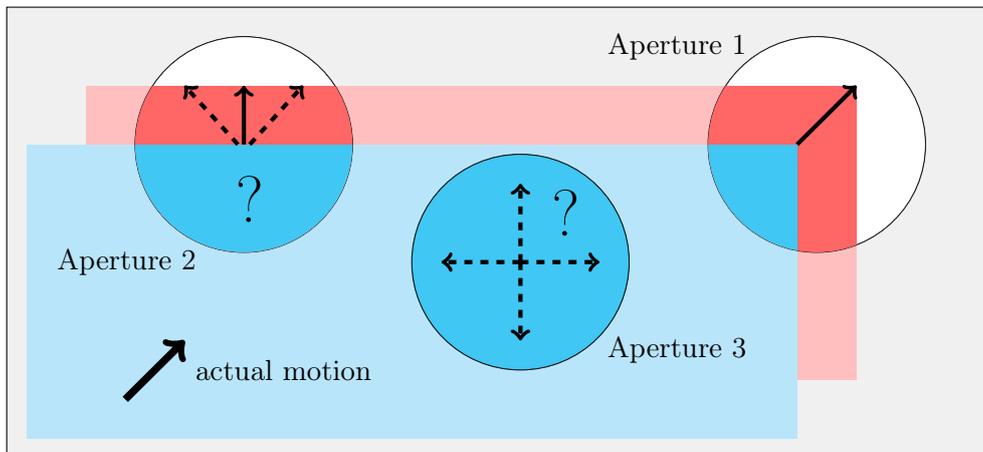


Figure 5: The aperture problem: The rectangle has moved in-between two frames of a sequence in northeastern direction (from the location depicted in cyan to the new location depicted in red) Aperture scenario 1: Since the gradient in both spatial dimensions is not equal to zero at the location of interest, the flow vector can be uniquely determined. Aperture scenario 2: Only the gradient in one dimension is not equal to zero. Equation 6 is now fulfilled for infinitely many flow vectors (all vectors pointing from the location of interest to the upper red horizontal line are valid solutions). The most meaningful of all possible flow vectors is chosen in this scenario, the one parallel to the direction of the gradient, the normal flow (depicted as a solid arrow). Aperture scenario 3: No gradient information is available at all and the movement of the rectangle cannot be detected. All possible vectors $(u, v)^\top$ are valid solutions to Equation 6.

also leads to the so called *filling-in effect*, which means that locations with missing information in the flow field are compensated by data from regions where flow data is available. The *Horn and Schunck energy functional* [HS81] is now obtained as:

$$E(\mathbf{w}) = \int_{\Omega} (f_x u + f_y v + f_t)^2 + \alpha (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) d\mathbf{x} \quad (11)$$

We can rewrite this energy functional more conveniently using the so called *motion tensor notation* [Far00], [BWS06]:

$$E(\mathbf{w}) = \int_{\Omega} \mathbf{w}^\top \mathbf{J} \mathbf{w} + \alpha (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) d\mathbf{x} \quad (12)$$

$$\text{with } \mathbf{J} = \nabla_3 f \nabla_3 f^\top = \begin{pmatrix} f_x^2 & f_x f_y & f_x f_t \\ f_x f_y & f_y^2 & f_y f_t \\ f_x f_t & f_y f_t & f_t^2 \end{pmatrix} \text{ and } \mathbf{w} = (u, v, 1)$$

If we again rewrite the energy functional more compactly as

$$E(u, v) = \int_{\Omega} F(x, y, u, v, u_x, u_y, v_x, v_y) dx dy \quad (13)$$

the minimizer is now found by computing the *Euler-Lagrange equations* of this energy:

$$0 \stackrel{!}{=} F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} \quad (14)$$

$$0 \stackrel{!}{=} F_v - \frac{\partial}{\partial x} F_{v_x} - \frac{\partial}{\partial y} F_{v_y}$$

assuming *homogeneous Neumann boundary conditions*

$$\mathbf{n}^\top \nabla u = \mathbf{n}^\top \nabla v = 0 \quad (15)$$

where \mathbf{n} is the exterior normal vector on the image boundary $\partial\Omega$.

By plugging the Horn and Schunck energy functional into Equations 14, one obtains:

$$0 = f_x^2 u + f_x f_y v + f_x f_t - \alpha \Delta u \quad (16)$$

$$0 = f_x f_y u + f_y^2 v + f_y f_t - \alpha \Delta v$$

or again in motion tensor notation:

$$0 = J_{11}u + J_{12}v + J_{13} - \alpha \Delta u \quad (17)$$

$$0 = J_{21}u + J_{22}v + J_{23} - \alpha \Delta v$$

These equations can now be discretized with standard discretization methods. One obtains a linear system of equations with $2 \cdot n_x \cdot n_y$ unknowns, where n_x and n_y are the resolutions in both spatial directions of the image. Since this linear system is usually very sparse, one can now efficiently use *iterative numerical solvers* such as the Jacobi or Gauss-Seidel method to find a solution. An iteration step of the Jacobi method reads:

$$\begin{aligned}
u_{i,j}^{k+1} &= \frac{\left(-[f_x]_{i,j}[f_t]_{i,j} - \left([f_x]_{i,j}[f_y]_{i,j}v_{i,j}^k - \alpha \sum_{\mathcal{N}(i,j)} \frac{u_{i,\tilde{j}}^k}{h_d^2} \right) \right)}{[f_x]_{i,j}^2 + \alpha \sum_{\mathcal{N}(i,j)} \frac{1}{h_d^2}} \\
v_{i,j}^{k+1} &= \frac{\left(-[f_y]_{i,j}[f_t]_{i,j} - \left([f_x]_{i,j}[f_y]_{i,j}u_{i,j}^k - \alpha \sum_{\mathcal{N}(i,j)} \frac{v_{i,\tilde{j}}^k}{h_d^2} \right) \right)}{[f_y]_{i,j}^2 + \alpha \sum_{\mathcal{N}(i,j)} \frac{1}{h_d^2}}
\end{aligned} \tag{18}$$

Since the energy functional is designed to be strictly convex, the method has at most one minimum. It was shown by [WS01a] that a unique solution always exists and that this solution depends continuously on the input data and the parameters of the model, which makes the problem *well-posed* (according to the original definition of well-posedness of Hadamard [Had02]).

This is a very desirable property of an optic flow method, especially in the context of object tracking, since this leads to a predictable behavior. The method of Horn and Schunck still gives only poor results, as seen in Figure 6.

The linearization of the constancy assumption in the energy functional makes the method unsuitable for a purpose such as object tracking, since the model only allow to capture very small displacements. In video sequences however, one usually has to deal with fast motion, low framerates and large spatial resolution.

Also, the homogeneous smoothness term is not able to preserve discontinuities in the flow field. If α is chosen too large, the displacement field will just get smoothed too much and motion information will be lost. If however α is chosen too low, the flow field will not be dense enough to reliably track targets.

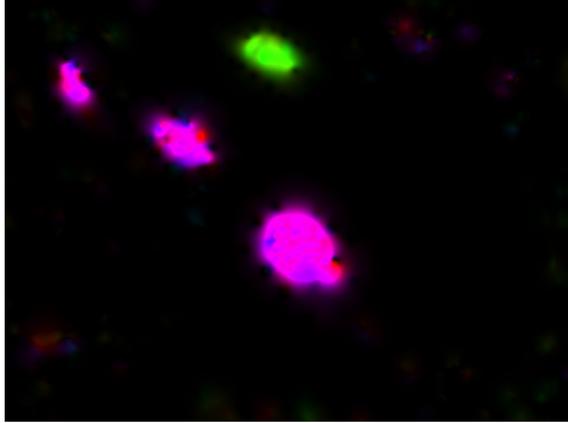
Also, the smoothness term only provides a smoothness of the solution in spatial directions. Since the tracking of objects is a process over time, a spatiotemporal smoothness assumption would be desirable and would lead to a more stable flow field in time direction.



(a) Frame 7



(b) Frame 8



(c) Flow field from frame 7 to 8

Figure 6: Variational optic flow: Result of the method of Horn and Schunck with parameters $\sigma_x = \sigma_y = 3.0$ and $\alpha = 20.0$

2.3 Variational Optic Flow for Large Displacements

Compared to the Horn and Schunck Method, the following method is a much more advanced approach in that it deals with most of the problems that we encountered before in an elegant way. Instead of linearizing the constancy assumption of the data term in our variational model as in Equation 6, we use the non-linearized brightness constancy assumption (Equation 4) and instead postpone the linearization to the numerical scheme. Using this new model we are now able to handle large displacements between successive frames.

Also, instead of penalizing outliers in the data term quadratically, we now reduce the influence of outliers on the result by wrapping the data term in a sub-quadratic function Ψ_D , as introduced by [BA91] and [BWS05]. In general, such a function should have the properties of being strictly convex, differentiable, positive, increasing and usually symmetric along the y-axis (as seen in Figure 7).

We can also apply the same concept to the smoothness term to penalize the gradient of the flow field less severely. If we wrap the smoothness term into a function Ψ_S with the same properties as above, we obtain a so called

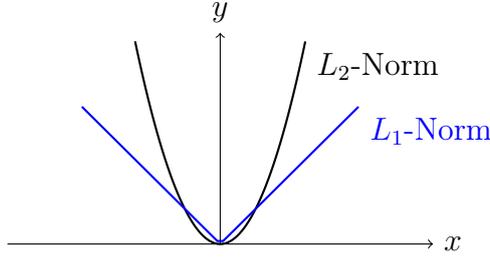


Figure 7: Error measure functions: $|x|$ denotes the deviation from the assumption, while the value of y denotes the applied penalty. As the deviation becomes larger, the quadratic L_2 -Norm penalizes deviations much more severely than the linear L_1 -Norm

flowdriven isotropic smoothness term. The idea of using a regularization for the smoothness term was first introduced by [SH89]. An overview on how to classify different image- and flowdriven smoothness terms in a diffusion based framework is presented in [WS01a]. This non-homogeneous smoothness assumption now also allows to respect discontinuities in the flow field instead of blurring them. The energy functional of the second method which we are going to consider is consequently given as:

$$E(\mathbf{w}) = \int_{\Omega} \Psi_D ((f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w}))^2) + \alpha \Psi_S (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) d\mathbf{x} \quad (19)$$

where as a penalizer, we use the regularized L_1 -Norm $\Psi_D = \Psi_S = \sqrt{s^2 + \epsilon^2}$ with $\epsilon = 0.001$.

Applying again the Euler-Lagrange equations from 14, we now obtain the following equations:

$$\begin{aligned} 0 &= f_x^2 u + f_x f_y v + f_x f_t - \alpha \operatorname{div} (\Psi'_S (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) \nabla \mathbf{u}) \\ 0 &= f_x f_y^2 u + f_y^2 v + f_y f_t - \alpha \operatorname{div} (\Psi'_S (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) \nabla \mathbf{v}) \end{aligned} \quad (20)$$

In motion tensor notation written as:

$$\begin{aligned} 0 &= J_{11} u + J_{12} v + J_{13} - \alpha \operatorname{div} (\Psi'_S (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) \nabla \mathbf{u}) \\ 0 &= J_{21} u + J_{22} v + J_{23} - \alpha \operatorname{div} (\Psi'_S (|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2) \nabla \mathbf{v}) \end{aligned} \quad (21)$$

The same boundary conditions as in Equation 15 apply.

We now have to deal with an energy that is not convex and that does not guarantee us a unique solution anymore. By not linearizing the constancy assumption, we now have to deal with implicit equations, since the unknowns u and v occur in the argument of our function.

One way to deal with the non-convexity is to split the problem into a series of convex optimization problems using a multiresolution coarse-to-fine strategy. The resolution to the next lower level is hereby reduced by a predefined factor of $\eta \in (0, 1)$, also referred to as *downsampling factor* or *warping factor*.

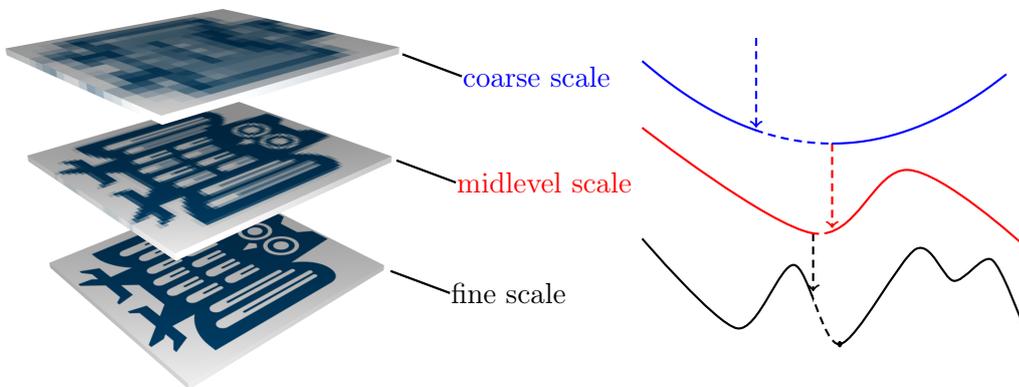


Figure 8: Coarse-to-fine scheme: The original fine scale image data is downsampled to several coarser resolutions. First, the convex problem at the coarsest scale (containing only small displacements) is solved. Then, the obtained flow field is upsampled to the next finer resolution and a difference problem at the next finer scale is solved. Of course, the subsequent image from the sequence has to be warped to compensate for the already computed motion on the previous coarser scale. This process is continued recursively until one reaches the finest scale. Summing up the optic flow result from all scales gives the desired solution.

Starting with the optic flow computation at the coarsest scale, the optic flow field at the next finer scale is obtained by upsampling the already computed flow to the finer resolution and by *warping* the image data by the already computed motion, such that only the flow difference on each scale has to be computed. Since larger displacements become smaller at coarser levels, this scheme allows us to linearize the constancy assumption for the optic flow computation on each single level. Therefore, by approximating our non-convex problem by a series of convex problems, we prevent getting stuck in poor local minima of our energy. The idea behind this approach is

illustrated in Figure 8. A mathematical justification is given in [Bru06] and [Pap+06].

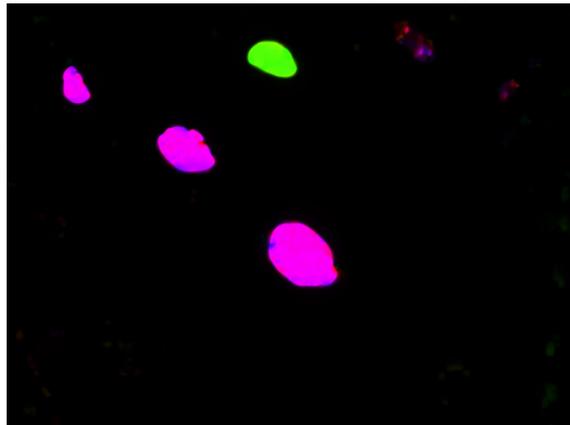
Results for this method are shown in Figure 9. One can easily see that the motion discontinuities of the flow field are now respected instead of just being blurred. Also, the method gives much more stable results for faster moving objects. However, since the results are obtained only in a spatial context between two consecutive frames, this often leads to unsmooth motion boundaries in time direction, which are often not stable enough to allow the computation of a useful trajectory. Besides, the method is also very sensitive to illumination changes since the data term relies only on the brightness constancy assumption. That is why it might be useful to also consider other constraints in the data term and to extend the idea of a smoothness assumption on the flow field from the spatial to the spatiotemporal domain.



(a) Frame 7



(b) Frame 8



(c) Flow field from frame 7 to 8

Figure 9: Variational optic flow for large displacements with parameters $\sigma_x = \sigma_y = 1.6$, $\alpha = 8.0$ and $\eta = 0.83$

2.4 The Spatiotemporal Method of Brox et al

The variational method of Brox et al [Bro+04] takes the ideas discussed in the last subsection even further. Instead of only assuming constancy on the gray values, we now additionally assume constancy on the spatial image gradient in the data term:

$$\begin{aligned} f_x(x, y, t) - f_x(x + u, y + v, t + 1) &= 0 \\ f_y(x, y, t) - f_y(x + u, y + v, t + 1) &= 0 \end{aligned} \tag{22}$$

To make the optic flow computation more robust under illumination changes, the data term now consists of a combined gray value and gradient constancy assumption, with a weighting between the two controlled by a parameter γ . How to choose γ depends mostly on the application at hand. While the gradient constancy assumption is invariant under global additive illumination changes, it may prove to be disadvantageous for situations with rotational motion, since it is inherently built on directional information. Again, we refrain from linearizing these assumptions in the model and instead use the coarse-to-fine strategy presented in Section 2.3. Also, since we now have two equations with two unknowns, the aperture problem becomes less distinct. To reduce the penalty for outliers, as detailed in Figure 7, a subquadratic penalizer for the data term is used. Instead of using a joint robustification strategy, as in the model below, it would also be reasonable to apply a robustification separately to both assumptions of the data term, since both assumptions are usually not correlated. An approach with a separate robustification strategy is presented in [BW05]. Our smoothness term now consists of an isotropic flowdriven smoothness assumption that extends into the spatiotemporal domain⁴. [Nag90; WS01b]. The energy functional is given as:

$$\begin{aligned} E(\mathbf{w}) &= \int_{\Omega \times T} \Psi_D ((f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w}))^2 + \gamma |\nabla f(\mathbf{x}) - \nabla f(\mathbf{x} + \mathbf{w})|^2) d\mathbf{x} \\ &\quad + \alpha \int_{\Omega \times T} \Psi_S (|\nabla_3 \mathbf{u}|^2 + |\nabla_3 \mathbf{v}|^2) d\mathbf{x} \end{aligned} \tag{23}$$

⁴Since one is working with temporal derivatives, it is advisable to not only apply a presmoothing in spatial, but also in temporal direction, as detailed in Equation 1

Consequently, the Euler-Lagrange equations now also extend to the spatiotemporal domain:

$$\begin{aligned}
0 &\stackrel{!}{=} F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} - \frac{\partial}{\partial t} F_{u_t} \\
0 &\stackrel{!}{=} F_v - \frac{\partial}{\partial x} F_{v_x} - \frac{\partial}{\partial y} F_{v_y} - \frac{\partial}{\partial t} F_{v_t}
\end{aligned} \tag{24}$$

with

$$E(u, v) = \int_{\Omega \times T} F(x, y, t, u, v, u_x, u_y, u_t, v_x, v_y, v_t) dx dy dt \tag{25}$$

Since we are now working in a spatiotemporal setting, we now also have to define *homogeneous Neumann boundary conditions* for the spatiotemporal domain

$$\mathbf{n}^\top \nabla_{\mathbf{3}} u = \mathbf{n}^\top \nabla_{\mathbf{3}} v = 0 \tag{26}$$

where n is the exterior normal vector on the image sequence boundary $\partial\Omega_3$.

After plugging in the values, we now obtain the following equations:

$$\begin{aligned}
0 &= \Psi'_D \left((f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w}))^2 + \gamma |\nabla f(\mathbf{x}) - \nabla f(\mathbf{x} + \mathbf{w})|^2 \right) \\
&\quad \cdot \left(f_x(\mathbf{x} + \mathbf{w}) (f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w})) \right. \\
&\quad \left. + \gamma f_{xx}(\mathbf{x} + \mathbf{w}) (f_x(\mathbf{x}) - f_x(\mathbf{x} + \mathbf{w})) \right. \\
&\quad \left. + \gamma f_{yx}(\mathbf{x} + \mathbf{w}) (f_y(\mathbf{x}) - f_y(\mathbf{x} + \mathbf{w})) \right) \\
&\quad + \alpha \operatorname{div} \left(\Psi'_S (|\nabla_{\mathbf{3}} \mathbf{u}|^2 + |\nabla_{\mathbf{3}} \mathbf{v}|^2) \nabla_{\mathbf{3}} \mathbf{u} \right)
\end{aligned} \tag{27}$$

$$\begin{aligned}
0 &= \Psi'_D \left((f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w}))^2 + \gamma |\nabla f(\mathbf{x}) - \nabla f(\mathbf{x} + \mathbf{w})|^2 \right) \\
&\quad \cdot \left(f_y(\mathbf{x} + \mathbf{w}) (f(\mathbf{x}) - f(\mathbf{x} + \mathbf{w})) \right. \\
&\quad \left. + \gamma f_{xy}(\mathbf{x} + \mathbf{w}) (f_x(\mathbf{x}) - f_x(\mathbf{x} + \mathbf{w})) \right. \\
&\quad \left. + \gamma f_{yy}(\mathbf{x} + \mathbf{w}) (f_y(\mathbf{x}) - f_y(\mathbf{x} + \mathbf{w})) \right) \\
&\quad + \alpha \operatorname{div} \left(\Psi'_S (|\nabla_{\mathbf{3}} \mathbf{u}|^2 + |\nabla_{\mathbf{3}} \mathbf{v}|^2) \nabla_{\mathbf{3}} \mathbf{v} \right)
\end{aligned}$$

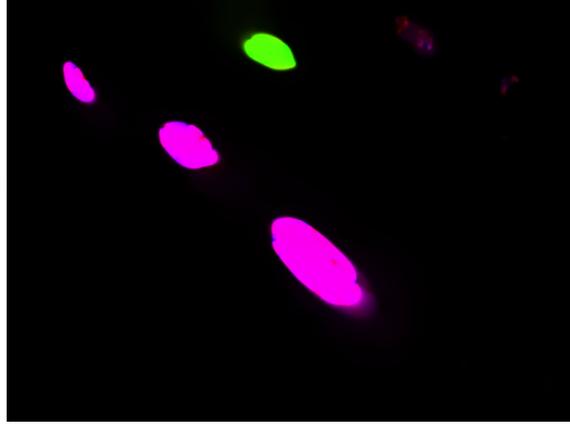
Results are shown in Figure 10. As can be seen, the temporal Gaussian smoothing and the new spatiotemporal smoothness term make the optic flow field more steady and robust across a whole sequence of frames. A spatiotemporal smoothing might however be problematic when dealing with abrupt motion changes and motion boundaries in time direction, since the temporal



(a) frame 7



(b) frame 8



(c) flow field from frame 7 to 8

Figure 10: Results of the spatiotemporal method of Brox et al. with parameters $\sigma_x = \sigma_y = 1.2$, $\sigma_t = 0.2$, $\alpha = 8.0$, $\gamma = 1.0$ and $\eta = 0.83$

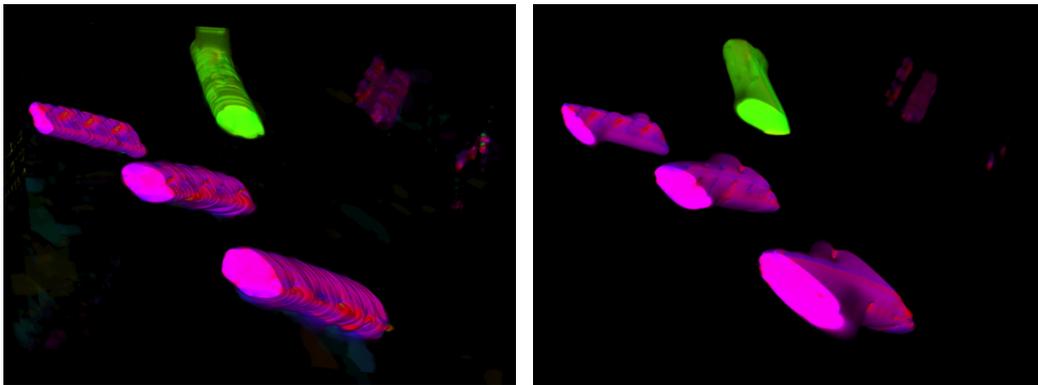
smoothness assumption causes a smearing effect in time direction. This can however be treated by adapting the grid size in time direction h_t . The differences between spatial and spatiotemporal smoothing are illustrated in Figure 11 using a 3D volume representation of the displacement fields. The gradient constancy assumption now also makes the methods more robust under illumination changes as seen in Figure 12.



(a) First frame

(b) Center frame

(c) Last frame



(d) Optic flow with spatial smoothness term (e) Optic flow with spatiotemporal smoothness term

Figure 11: Spatial optic flow compared to spatiotemporal optic flow: The spatiotemporal smoothing causes the flow field to more more consistent over time

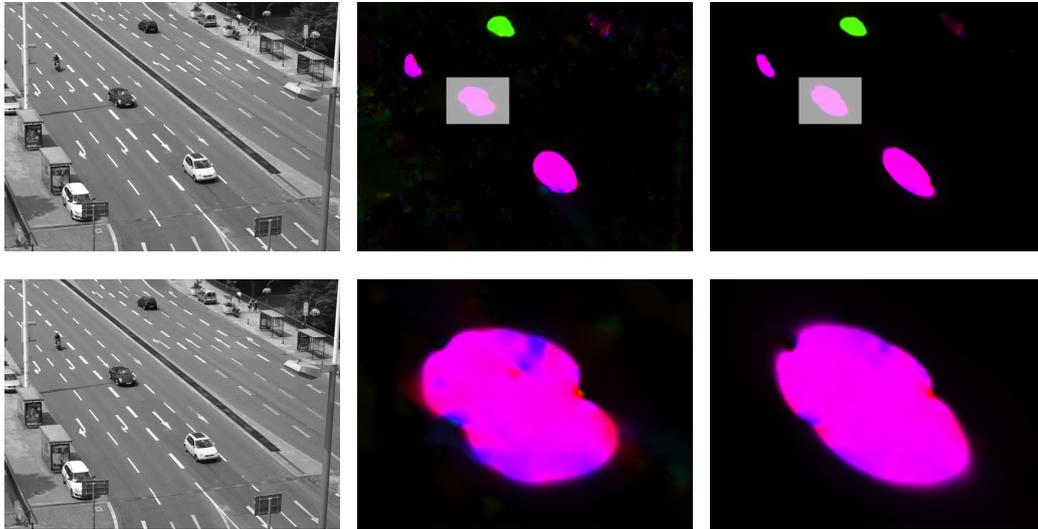


Figure 12: Optic flow under illumination changes: The car in the center of the image passes through the shadow of a lamp post: Left column: frames 21 and 22 from original sequence, Center column: Optic flow with brightness constancy assumption only (enlarged area below), Right column: Optic flow with combined brightness and gradient constancy assumption (enlarged area below)

3 Segmentation of Displacement Fields

Optic flow computation provides us with a mechanism to detect correspondences between pixels in subsequent frames of an image sequence. To be able to track objects throughout an image sequence, it is essential to transform the optic flow displacement field to a higher-level representation. This can for example be achieved by segmenting the displacement field.

Most video tracking methods rely on some sort of segmentation method. Since objects are defined by their edges, edge information is always the most important tool in object detection. Common segmentation methods include mean shift clustering, graph cuts and active contour models. Since we are segmenting a three-dimensional vector field instead of a regular gray value image, our approach will have to be suited to the task.

In the literatur, there exist several different definitions of segmentation problems. Barrow and Tenenbaum [BT78] describe the problem of image segmentation as “the process of partitioning an image into semantically interpretable regions”. A *region (segment)* R_i is a subset of an image. For a set of regions to be a *segmentation*, it must hold that the partitioning is exhaustive:

$$\bigcup_i R_i = I \quad (\text{where } I \text{ is the entire image}) \quad (28)$$

that it is exclusive:

$$R_i \cap R_j = \emptyset \quad \forall i \neq j \quad (29)$$

that the regions are uniform:

$$P(R_i) = \text{true} \quad \forall i \quad (30)$$

and maximal:

$$P(R_i \cup R_j) = \text{false} \quad \forall i \neq j \quad (31)$$

where, given some arbitrary uniformity criterion U , the predicate $P(R)$ evaluates to true if $\exists a \ni \|U(i, j) - a\| < \epsilon, \forall (i, j) \in R$.

3.1 Thresholding

Since the optic flow field only provides us with an approximation of the actual motion in the image, we have several regions in the optic flow field, where the magnitude of flow vectors is larger than zero, but where we do not want to detect any actual motion. These situations can be caused by camera shake, noise, etc. during the acquisition of the sequence. The smoothness term of variational approaches not only provides us with dense data of the actual

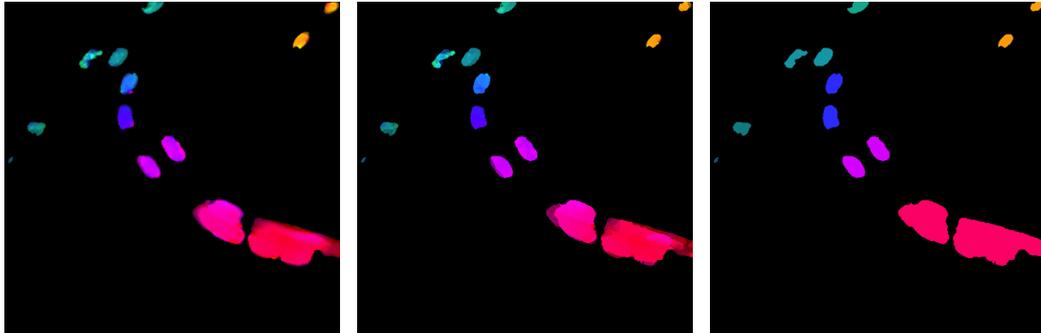
motion, but it also distributes these artifacts. An important observation is that the flow vectors that belong to objects that we actually want to detect usually have a magnitude far larger than the magnitude of those vectors that we want to discard. This gives us a very simple criterion to discern actual movement from noise. One simply introduces a *threshold* τ , such that all regions with gradient magnitude smaller than τ will be treated as background and not considered as a possible object anymore. τ can be chosen as a constant value or as a quantile of the data. An example of how oversegmentation can be treated by applying a threshold is displayed in Figure 13d.



(a) Frame 29 (Gaussian pre-smoothed with $\sigma = 1.0$)

(b) Frame 30 (Gaussian pre-smoothed with $\sigma = 1.0$)

(c) Displacement field between frames 29 and 30



(d) Magnitude thresholded displacement field from (c), using a $\tau = 1\%$ quantile

(e) Watershed segmented displacement field from (d)

(f) Region merged displacement field from (e)

Figure 13: Segmentation of displacement fields

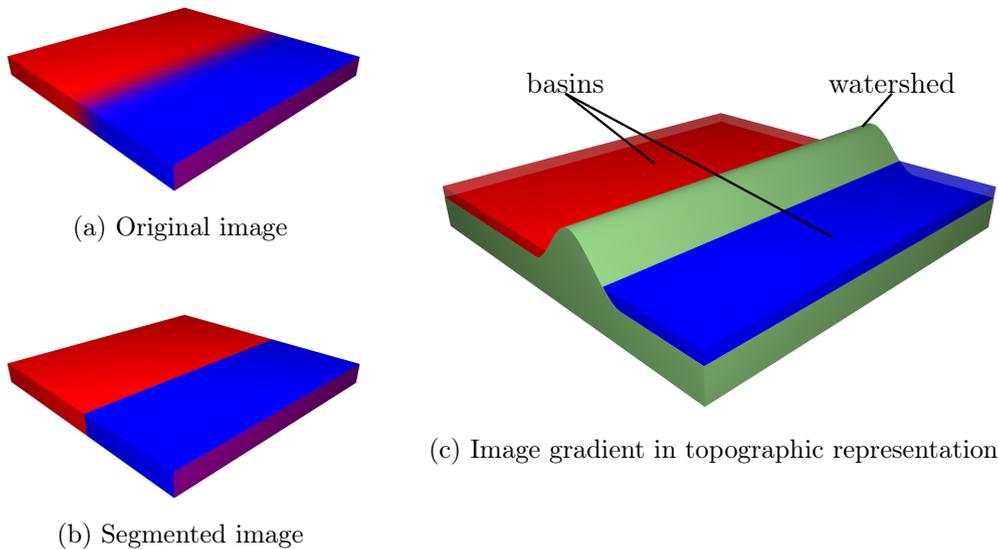


Figure 14: Illustration of the watershed segmentation: a) The original image seen from a 3D perspective. b) The gradient magnitude of the original image is illustrated as a topographic landscape (in green). By performing a gradient descent on the landscape (always taking the path downhill), we obtain two regions (depicted as basins) and separated by the boundary (the watershed). c) The segmentation result.

3.2 3D Watershed on Vector Fields

Since we are working with volume data, we would also like to exploit the third dimension for our segmentation algorithm. There are already a wide variety of volume segmentation algorithms available [Wir07]. For segmentation of the optic flow field, we are going to choose a basic and simple, yet efficient algorithm, called the *watershed transformation*, which belongs to the class of morphological segmentation methods. The principle is very simple and was introduced by Beucher and Lantuejoul [BL79] in 1979. Their original approach, called *watershed by flooding*, follows the following idea: A water source is placed in each regional minimum in the relief⁵. The entire relief is now flooded starting from the sources. Barriers (the segmentation boundaries) are created when water surfaces meet. The resulting set of barriers constitutes a watershed segmentation by flooding. The *toboggan watershed algorithm* that we are going to use differs slightly in the way that the algo-

⁵a relief can be thought of as the gradient magnitude image represented as a topographic height map

rithm progresses. Instead of a flooding, the toboggan watershed algorithm is based on the *topographic distance*:

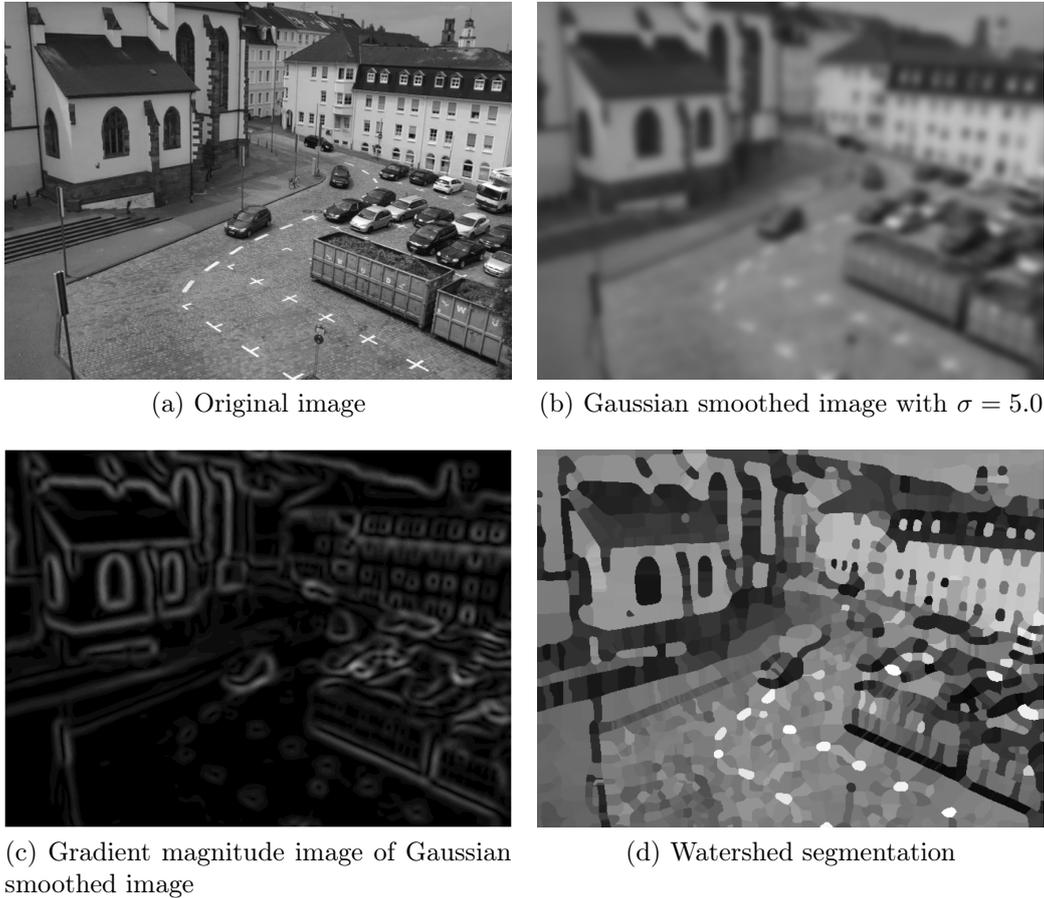


Figure 15: The watershed segmentation on a gray value image

If we imagine the gradient magnitude of a 2D image as a three-dimensional relief, we get a topographical representation of a landscape. When letting a raindrop fall onto this landscape from above, it will always choose the path of steepest descent until it reaches the bed of the valley. All points that the raindrop traversed will be assigned to the same segment. This also means that the algorithm will always produce closed contours which in our case is very desirable. Figure 15 illustrates the concept of a topographic representation of the gradient magnitude. The result of such a watershed algorithm applied to a gray value image is illustrated in Figure 14. A disadvantage of the watershed algorithm is that it is usually prone to oversegmentation of the data. This can be counteracted by applying a Gaussian smoothing to

the computed displacement field. A simple algorithm in pseudocode for a watershed segmentation applied on a 3D vector field is shown in Algorithm 1. The algorithm needs linear processing time and has linear storage space consumption in reference to the number of voxels.

Algorithm 1 3D Watershed Toboggan Algorithm on Vector Fields

```

1: procedure 3D WATERSHED
2:   compute gradient magnitude of 3D vector field
3:    $\text{gradmag}(x, y, t) == \sqrt{u_x^2 + u_y^2 + u_t^2 + v_x^2 + v_y^2 + v_t^2}$ 
4:   for all voxels do
5:     if voxel not yet visited then
6:       follow a path along the smallest derivative
7:       put all points encountered on the path on a stack
8:       if minimum (p,q) is reached then
9:         stop and assign all pixels on stack the value
10:        u(p,q) (from the original vector field)

```

3.3 Region Merging of 3D Volumes

Oversegmentation can be a severe problem for an object tracking algorithm. That is why besides the previous thresholding step, we will have to perform an additional postprocessing of the segmentation result, that is often used in combination with the watershed. The idea of *region merging* is to merge those neighboring segments that have a certain similarity. As mentioned before, we assumed that vectors are similar (and therefore are to be treated as a segment) if the uniformity criterion of the vectors is below a certain threshold. Region merging basically is the process of raising this threshold to a certain extent. This can be achieved by introducing a *similarity measure* between vectors that has values in the range $[0, 1]$.

The *Euclidian distance*, which is usually used in Euclidian space to measure distances⁶ is not appropriate for our case, since it completely discards the direction of the two vectors:

$$\text{sim}_{Euclidian}(\mathbf{x}, \mathbf{y}) = \frac{1}{1 + \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2}} \quad (32)$$

A measure that is commonly used in the field of information retrieval to compare the similarity between two documents represented as vectors is the

⁶since we are looking for a similarity measure instead of a distance measure, we simply use the inverse of the distance

cosine similarity. The cosine similarity measures the cosine of the angle between two vectors:

$$sim_{cosine}(\mathbf{x}, \mathbf{y}) = \frac{1 + \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}}{2} \quad (33)$$

This measure however gives us only a similarity criterion for direction, but discards the magnitude of the vectors.

Following [FJ90], we use a similarity measure based on the *angular error* (also referred to as the *angular similarity*) between two flow vectors $\mathbf{x} = (x_1, x_2)^\top$ and $\mathbf{y} = (y_1, y_2)^\top$ which can be defined as:

$$sim_{angular}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\arccos\left(\frac{x_1 y_1 + x_2 y_2 + 1}{\sqrt{x_1^2 + x_2^2 + 1} \sqrt{y_1^2 + y_2^2 + 1}}\right)}{\pi} \quad (34)$$

After performing the segmentation on a vector field, each segment is naturally described by a single displacement vector. After merging several segments, the direction of the new segment will be defined as the volume weighted average of the individual segments. That means if two neighboring 3D-segments A with a volume of 80 voxels and displacement vector (10, 0) and B with 20 voxels and a displacement vector (0, 10) are to be merged, the new merged segment AB will have a volume of 100 voxels with new direction (8, 2). Since only regions with similar direction are merged, this automatically protects the direction of the new segment from being spoiled by outliers from smaller segments.

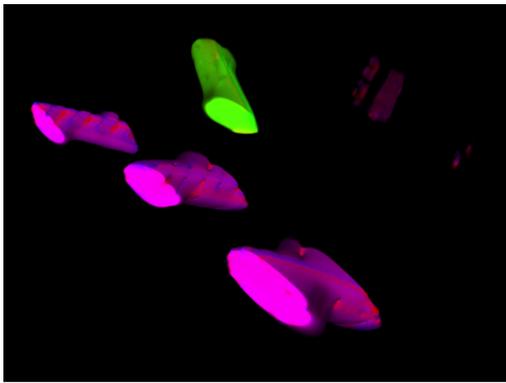
When implemented efficiently, the algorithm has linear processing time and storage space consumption in reference to the number of voxels.



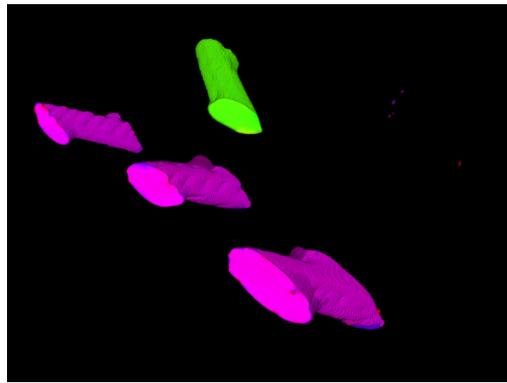
(a) Frame 2

(b) Frame 20

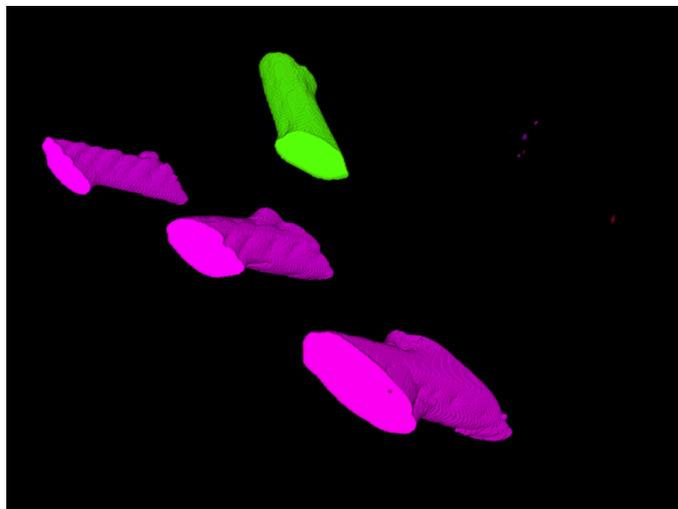
(c) Frame 40



(d) Brox et al displacement field



(e) Displacement field after applying thresholding and a watershed segmentation



(f) Displacement field after applying thresholding and a watershed segmentation with region merging

Figure 16: Illustration of optic flow postprocessing using a 3D volume representation

4 Results

After performing the segmentation of the flow field, the data is now transferred into the state space Ω_s which represents the tracking result. The state space will be defined as a 3D mask $id(x, y, t)$ whose range consists of integer values that constitute the IDs of the targets. All locations that do not belong to a target will be set to zero and treated as background. This representation contains all the information necessary to identify and locate targets in our sequence. In addition to the position of the target which is stored in this mask, the displacement obtained from the region merged vector field contains additional directional- and velocity information of each target. We define the *size of a target* t_i as the number of voxels that the target occupies in the volume, or expressed differently, as the number of points for which $id(x, y, t) = id(t_i)$ holds.

To represent the tracking result visually, we can now superpose the silhouette of the target onto the original image. The color of the overlay corresponds to the pseudocolor-encoded value of the displacement for this target. Figures 17, 18 and 19 show the tracking results of three different image sequences using the target representations introduced in Section 1.5.



Figure 17: Tracking result of a traffic scene with four cars and two pedestrians. The result of the tracking computation is displayed as a translucent overlay on top of the original image sequence, using a contour representation. The colors of the overlay represent the direction of movement of the targets according to the scheme from Figure 4a. The pedestrians in the topright corner were not classified as targets, since they are moving too slow and were below the magnitude threshold.

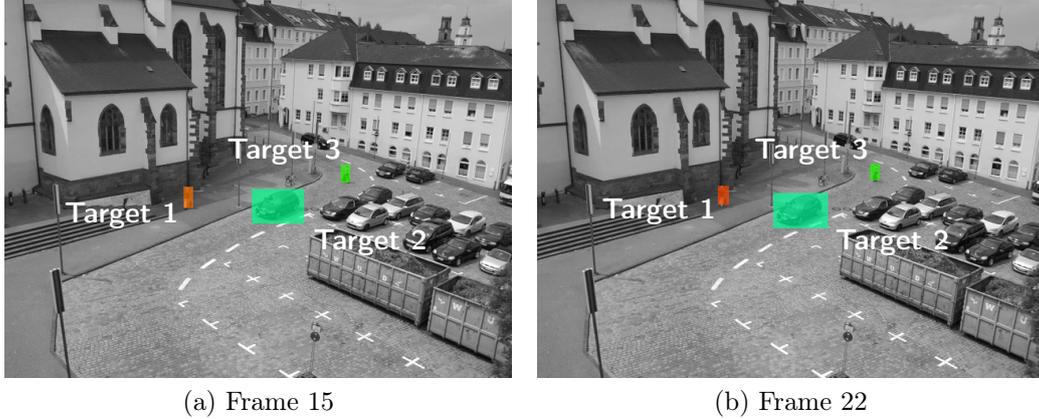


Figure 18: Tracking result of a traffic scene with one car and two pedestrians. The result of the tracking computation is displayed as a translucent overlay on top of the original image sequence, using a rectangular representation. The colors of the overlay represent the direction of movement according to the scheme from Figure 4a. Using the method of Brox et al., the algorithm is even able to track the pedestrian during the transition from sun to shadow (Target 1).

4.1 Tracking Score

Our goal now is to judge the quality of the optic flow result using the tracking information. We introduce a *tracking score* in the range $[0, 1]$ that shall reflect the homogeneity and eventually the quality of the optic flow field. The score is designed in such a way that it leads to a better score if the trajectory is stable and has no abrupt motion discontinuities (excluding the birth and death of the target of course) and leads to a lower score if the trajectory is unstable, which indicates that the optic flow was volatile and possibly erratic.

The score is obtained as follows: For each target t_i that was identified during the tracking process we compute the following score: The displacement $(u, v, 1)$ at each volume location $id(x, y, t)$ points to the new location $id(x + u, y + v, t + 1)$. The number of voxels for which it holds that $id(x, y, t) = id(x + u, y + v, t + 1)$ and $id(x, y, t) = id(t_i)$, will be denoted as $num_{hits}(t_i)$. The score for object t_i is now given as $\frac{num_{hits}(t_i)}{size(t_i)}$. The scores of all objects are now weighted and accumulated to a single score. Obviously, we also have to normalize this score by dividing through the size of each respective target to prevent large objects from automatically obtaining better scores than smaller objects.

Using this measure, a high score will be an indication that the computed

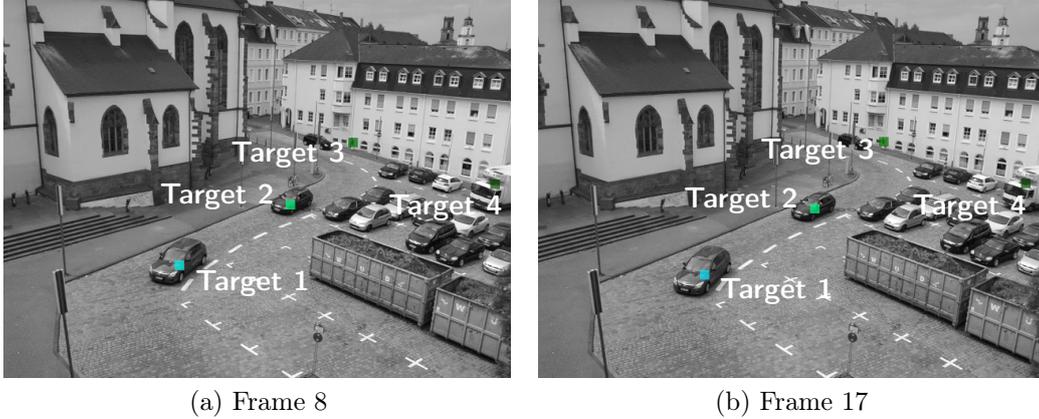


Figure 19: Tracking result of a traffic scene with two moving cars, one slowly moving truck and one pedestrian. The result of the tracking computation is displayed as a translucent overlay on top of the original image sequence, using a point representation. The colors of the overlay represent the direction of movement according to the scheme from Figure 4a. With a lower magnitude threshold, the algorithm is able to track all objects in the scene, even the slow truck in the right corner which is barely moving at all.

flow field is stable. If the tracking algorithm allows us to track a target over a longer sequence of time, without jumps in size or in the motion pattern, this gives us an indication that the optic flow, which was used to compute the tracking data must have been good. Otherwise, if the displacement of many voxels of the target points outside of the object in the next frame, the optic flow can be classified as unreliable and the score is lowered. Of course, one has to consider that motion discontinuities can also occur in real life situations, however, when using a sufficient framerate and a smoothing in temporal direction, such abrupt motions of objects will not influence the score in a negative way.

4.2 Performance Evaluation of the Optic Flow Result

The tracking data can be used to evaluate the performance of an optic flow computation result. The tracking score introduced in the previous subsection gives a useful indication if the optic flow is of good quality. Optic flow results for the different methods introduced in Section 2 are shown in Figure 20 while their corresponding score is displayed in Table 20. The results obtained mostly agree with the visual perception of the quality of the optic flow.

When working with real world video footage, usually there is no *ground*

truth data available to judge the quality of the optic flow computation. That is why one is interested in finding ways to assess the quality of the computed motion information. Since the traffic scenes used in this work also are non-artificial scenes, there is no ground truth available. Only a visual comparison of the results is possible (see Figure 20).

Another interesting application of such a score is the automatic optimization of parameter values for optic flow methods. By optimizing the parameter values, the best score that could be achieved was 0.944503 using the method of Brox et al. with the parameter values $\alpha = 10.5$, $\gamma = 1.0$, $\sigma_x = \sigma_y = 1.8$, $\sigma_t = 1.2$. These settings also give a good result visually, as seen in Figure 20f.

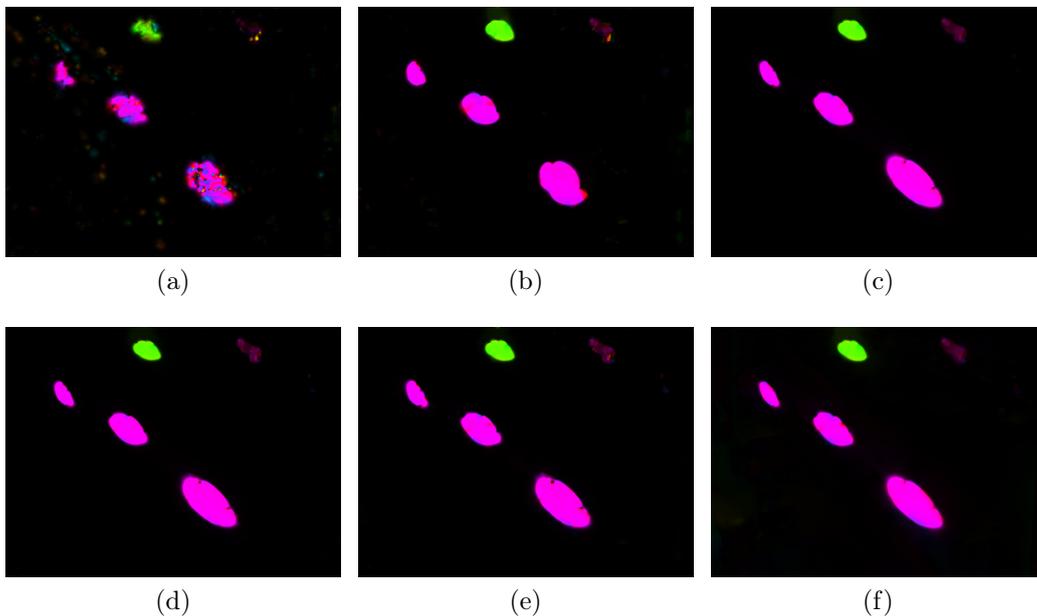


Figure 20: Evaluation of several optic flow computation results using the methods presented in Section 2 with different parameters. The results are shown in Table 1.

4.3 Future Work

There are more advanced variational methods available for computing the optic flow, than the ones used in this work. One possible method that could provide improvements to the optic flow result is the method of Zimmer et al. [ZBW11] which exploits information from the HSV colorspace in its data term

a)	Horn and Schunck: $\sigma_x = \sigma_y = 1.6, \sigma_t = 0.2, \alpha = 8.0$	0.8971
b)	Large Displacements: $\sigma_x = \sigma_y = 1.6, \sigma_t = 0.2, \alpha = 8.0$	0.9066
c)	Brox et al.: $\sigma_x = \sigma_y = 1.6, \sigma_t = 0.2, \alpha = 8.0, \gamma = 1.0$	0.9345
d)	Brox et al.: $\sigma_x = \sigma_y = 0.8, \sigma_t = 0.2, \alpha = 4.0, \gamma = 1.0$	0.9328
e)	Brox et al.: $\sigma_x = \sigma_y = 1.6, \sigma_t = 0.2, \alpha = 8.0, \gamma = 1.0$	0.9329
f)	Brox et al.: $\sigma_x = \sigma_y = 1.8, \sigma_t = 1.2, \alpha = 10.5, \gamma = 1.0$	0.9445

Table 1: Tracking scores for the three optic flow methods presented in Section 2. The letter in the left column refers to the images in Figure 20

and uses an anisotropic smoothness term that includes directional information from the data term. Also, instead of performing the segmentation of the displacement field with a watershed, one could also consider other segmentation approaches, such as the Mumford Shah cartoon model, region growing or Bayesian methods. [Wir07] gives a survey on segmentation methods for 3D volumetric data. By interpreting the segmentation result differently, the trajectory management could also be improved. It would be possible to extract additional information from the optic flow field or to include additional knowledge about the nature of the targets that could then be used to improve the tracking information and hence the tracking score.

4.4 Conclusions

Variational optic flow methods rank among the most precise optic flow computation approaches available. It was shown that the resulting optic flow field can be used straightforward as a feature space for object tracking and that variational optic flow allows precise silhouette tracking out of the box.

The optic flow result also provides additional data such as directional- and velocity information which is available for the tracking process. It was also shown that the tracking data can be used to efficiently evaluate a given optic flow field. A tracking score was introduced which allowed to judge the quality of an optic flow field by interpreting the tracking information. The tracking score even allows to optimize the parameter values of the optic flow method to a certain extent.

4.5 Sources

All images and graphics in this work were created by the author, if not otherwise noted. Parts of the source code written for this thesis are based upon the work of Andrés Bruhn at the Mathematical Image Analysis Group of Saarland University [Bru06]. The 3D volumetric representations of displacement

fields were created using the open-source and license-free software ImageJ ([FR12; BA10]).

A Documentation of Accompanying Software

The software allows to read an image sequence and perform optic flow-, segmentation- and object tracking computations on the sequence. The parameters which are displayed on the terminal can be set interactively while the program is running. The software was implemented in C. The graphical user interface was written using GLUT, a system independent toolkit for writing OpenGL programs. It is based on the materials from the lecture "Advanced Image Analysis" [Sch13] by Christian Schmalz. To compile the program, the source folder contains a makefile that can be executed by typing `make` on the terminal. Depending on the architecture used, the according OpenGL library has to be selected manually in the makefile. The program is then launched by typing the following command:

```
./tracking folder
```

where `tracking` is the name of the program and `folder` is the relative path to a folder containing the image sequence. The folder must contain pgm files with identical dimensions. The files are read as a single image sequence in alphanumerical order according to their filenames.

A.1 Contents of the Accompanying Medium

- The folder `thesis` contains this document in pdf format.
- The folder `src` contains the accompanying source code of this work.
- The subfolders of `src` contain the image sequences used in this work. If a sequence can not be processed because of high memory requirements, it is recommended to decrease the number of frames or to decrease the resolution of the sequence.
 - `whbruecke`: Image sequence of a traffic scene with three cars, a motorcycle and two pedestrians in the far background. Location: Wilhelm-Heinrich Brücke, Saarbrücken
 - `schloss` Image sequence of a traffic scene of a car and two pedestrians. Location: Am Schlossberg, Saarbrücken
 - `schlossberg` Image sequence of a traffic scene of two cars, a truck and a pedestrian. Location: Am Schlossberg, Saarbrücken
 - `ettlingertor` Image sequence from a traffic surveillance camera. Location: Ettlinger Tor, Karlsruhe

- The folder `volumevideos` contains videos of full rotational views of the volume representations of the displacement fields used in this work.

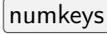
A.2 Sources

The libraries for memory allocation, image resampling, Gaussian filtering, file read- and write operations and vector field visualization are based on source code from Andrés Bruhn [Bru06] and have been partially modified and extended for use with 3D input data. The stack implementation that was used to implement the watershed segmentation- and region merging algorithm is based on source code by Robert I. Pitts [Pit07]. The “Ettlinger Tor” image sequence in Figure 13 was introduced by Kollnig and Nagel [KN97] and is now available on the image sequence server at [Nag14].

A.3 Keyboard Commands

For the following keyboard commands to work, the OpenGL Window must be in focus. All the following parameters can be altered while the program is running. The currently active parameter is indicated by capital letters, colored font and dashes.

Parameters such as the grid size or the boundary size can also be set dynamically, but were not included in the frontend. These parameters can however be changed in the file `main.c`, if desired.

	Go to the previous frame of the sequence.
	Go to the next frame of the sequence.
	Switch to above parameter in the list.
	Switch to parameter below.
	Toggle through options for active parameter. (only applicable to parameters with non-numeric value)
	Input of new value for the active parameter (only applicable to parameters with numerical value).
	
	Performs a computation pass (optic flow, segmentation, tracking) with the currently selected settings and parameter values. After the computation is finished, the OpenGL window will be updated with the resulting sequences.
	Ends the program.

- F1 Write the sequence currently displayed in the topleft frame of the frontend to files.
- F2 Write the sequence currently displayed in the topright frame of the frontend to files.
- F3 Write the sequence currently displayed in the bottomleft frame of the frontend to files.
- F4 Write the sequence currently displayed in the bottomright frame of the frontend to files.
- F5 Toggle content currently displayed in topleft frame: [original input sequence \rightarrow Gaussian blurred input sequence \leftrightarrow]
- F6 Toggle content currently displayed in topright frame: [tracking result with currently selected target representation \rightarrow original sequence \leftrightarrow]
- F7 Toggle content currently displayed in bottomleft frame: [optic flow displacement field \rightarrow Gaussian blurred optic flow displacement field \leftrightarrow]
- F8 Toggle content currently displayed in bottomright frame: [magnitude thresholded and watershed segmented displacement field \rightarrow magnitude thresholded, watershed segmented and region merged displacement field \leftrightarrow]

In addition to being able to switch between the parameters with the ↑ and ↓ keys, the parameters can also be directly selected using the following keys (The following list also contains a short explanation of each parameter):

- z Zoom factor of the OpenGL window. In cases where the sequence is too large to be displayed on a monitor, the frontend is automatically rescaled to a factor < 1.0 . Using a value other than 1.0 leads to aliasing and should only be used if the screen and the image sequence make it necessary.
- t The captioned text in the frontend can either be hidden or displayed using this parameter.

- u** The computation of the optic flow /segmentation
- f** /tracking **v** for the next computation pass can be enabled or disabled. Only those functions that are enabled will be recomputed in the next computation pass using the selected parameter values. If the tracking computation is enabled, the segmentation will also be automatically enabled.
- q** The optic flow method to be used in the next computation pass.
- p** The standard deviation σ for the optic flow pre-smoothing step. (Can be set separately for all three dimensions.)
- a** Parameter α controlling the weight of the optic flow smoothness term.
- g** Controls the weight of the gradient constancy assumption term of the motion tensor. To obtain correct results for the method of Horn and Schunck and the method for large displacements, γ should be set to 0, since these methods do not contain such a term.
- i** The number of inner /outer iterations for the optic flow computation. In the case of the Horn and Schunck method, **i** controls the number of Jacobi iterations while **o** has no effect on the computation.
- o**
- s** The ω parameter for successive over-relaxation. Interval: $[0, 2)$
- e** The warping/refinement factor η . Interval: $]0, 1[$. A high value leads to a more accurate result but also to a longer computation time. The program automatically selects the highest number of warping levels possible, depending on this parameter. This parameter has no effect when the method of Horn and Schunck is selected.
- d** To make large or small displacements better visible, the magnitude of the vectors of the displacement field can be scaled with this factor. The bottomleft and bottomright frames are updated in real-time.

- h To obtain visual information about the range of values in the displacement field, these parameters allows to set a range of values $[min, max]$. All vectors that are outside of this range are set to the value of the background color. See also b.
- j
- b The area of the image that is out of the range defined by h and j blinks in the frontend.
- n The standard deviation σ for the segmentation pre-smoothing step. (Can be set for all three dimensions separately.)
- r Similarity threshold for region merging, to decide if two adjacent segments are to be merged. Normalized value in the interval: $]0, 1[$. A high value will cause the algorithm to merge only segments with high similarity. A low value will merge segments, even if the vectors point in fairly different directions.
- m Parameter for magnitude thresholding. All displacement values with a value below this threshold will be set to 0 in the next computation and assigned as background by the tracking algorithm.
- x The boundaries between segments in the bottomright frame can be hidden or highlighted using white lines.
- k l The tracking algorithm will only consider an object as a possible target if it covers a certain volume $[min, max]$ (normalized to the range $[0, 1]$) of the image.
- w This parameter allows to choose between three different target representations (point representation, rectangle representation, contour representation). The representation in the frontend is update only after the next computation pass.

c

This parameter allows to choose between several different pseudocolor representations for displacement fields. To facilitate the use of the pseudocolor representation for people with color deficiencies, several different color schemes are available:[Bruhn: original color scheme by Andrés Bruhn [Bru06] → RBMY: right: red, down: blue, left: yellow, up: magenta → Bruhn W: Bruhn color scheme on white background ↔] The color scheme is updated in the frontend in real-time.

y

To be able to distinguish different targets and follow their track, it is necessary to highlight certain targets according to their id, assigned by the tracking algorithm. The target with the chosen id is highlighted with a blinking overlay in the frontend.

esc end progr.	write topleft F1	write topright F2	write btm.left F3	write btm.right F4	toggle topleft F5	toggle topright F6	toggle btm.left F7	toggle btm.right F8	F9	F10	F11	F12	power
± \$	1	2	3	4	5	6	7	8	9	0	-	+ =	←
toggle non-numer. parameters →	Q optic flow method	W target represent.	E warping eta	R regionmg. threshold	T display text	Y current target	U recalculate optic flow	I inner iterations	O outer iterations	P Gaussian Smooth OF	{ [}]	perform computat. ↩
•	A smoother. term wght.	S SOR omega	D scale displacem.	F recalculate segment.	G weight for gradient c.	H minimum displacem.	J maximum displacem.	K minimum obj. size	L maximum obj. size	:	;	! /	↩
↩	~	Z zoom factor	X show seg. boundaries	C color scheme	V recalculate tracking	B blink when out of rng.	N Gaussian Smth. SEG	M magnitude threshold	> .	? /			↩
fn	ctrl	alt	cmd						⌘	cmd	alt	prev. param. next param.	prev. frame next frame

References

- [AKM02] Radhakrishna S. V. Achanta, Mohan S. Kankanhalli, and Philippe Mulhem. “Compressed domain object tracking for automatic indexing of objects in MPEG home video”. In: *International Conference in Multimedia and Expo*. 2002, pp. 61–64.
- [Ana89] P. Anandan. “A computational framework and an algorithm for the measurement of visual motion”. In: *International Journal of Computer Vision* 2.3 (1989), pp. 283–310.
- [BA10] Thomas Boudier and Philippe Andrey. *3D Processing and Analysis with ImageJ*. Tech. rep. Domaine de Vilvert, INRA, Jouy-en-Josas, Université Pierre et Marie Curie, Paris, France, 2010. URL: <http://imagejdocu.tudor.lu/lib/exe/fetch.php?media=tutorial:working:workshop3d2010.pdf>.
- [BA91] Michael Julian Black and P. Anandan. “Robust dynamic motion estimation over time”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Maui, Hawaii, USA, 1991, pp. 296–302.
- [BB95] Steven S. Beauchemin and John L. Barron. “The Computation of Optical Flow”. In: *ACM Computing Surveys* 27.3 (1995), pp. 433–467.
- [BFB94] John L. Barron, David J. Fleet, and Steven S. Beauchemin. “Performance of optical flow techniques”. In: *International Journal of Computer Vision* 12.1 (1994), pp. 43–77.
- [BGW91] Josef Bigün, Gösta H. Granlund, and Johan Wiklund. “Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.8 (1991), pp. 775–790.
- [BL79] S. Beucher and C. Lantuejoul. *Use of Watersheds in Contour Detection*. Tech. rep. Centre de Géostatistique et de Morphologie Mathématique, Fontainebleau, France, 1979.
- [Bla92] Michael Julian Black. “Robust Incremental Optical Flow”. PhD thesis. Yale University, Connecticut, USA, Dec. 1992.
- [Bro+04] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *European Conference on Computer Vision*. Vol. 3024. Prague, Czech Republic, 2004, pp. 25–36.

- [Bru06] Andrés Bruhn. “Variational Optic Flow Computation, Accurate Modelling and Efficient Numerics”. PhD thesis. Saarland University, Saarbrücken, Germany, Aug. 2006.
- [BT78] Harry G. Barrow and J. Martin Tenenbaum. *Recovering Intrinsic Scene Characteristics From Images*. Tech. rep. 157. 333 Ravenswood Ave., Menlo Park, California, USA: AI Center, SRI International, Apr. 1978.
- [BW05] Andrés Bruhn and Joachim Weickert. “Towards Ultimate Motion Estimation: Combining Highest Accuracy with Real-Time Performance”. In: *Tenth IEEE International Conference on Computer Vision 1* (2005), pp. 749–755.
- [BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. “Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods”. In: *International Journal of Computer Vision*. Vol. 61. 3. Springer Science + Business Media, 2005, pp. 211–231.
- [BWS06] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. “A Multigrid Platform for Real-Time Motion Computation with Discontinuity-Preserving Variational Methods”. In: *International Journal of Computer Vision* 70.3 (2006), pp. 257–277.
- [Enk91] Wilfried Enkelmann. “Obstacle detection by evaluation of optical flow fields from image sequences”. In: *Image and Vision Computing* 9.3 (1991), pp. 160–168.
- [Far00] Gunnar Farneback. “Fast and Accurate Motion Estimation Using Orientation Tensors and Parametric Motion Models”. In: *International Conference on Pattern Recognition*. 2000, pp. 1135–1139.
- [FJ90] David J. Fleet and Allan D. Jepson. “Computation of component image velocity from local phase information”. In: *International Journal of Computer Vision* 5.1 (1990), pp. 77–104.
- [FR12] Tiago Ferreira and Wayne Rasband. *ImageJ User Guide / Fiji 1.46 Revised Edition*. Tech. rep. National Institutes of Health, U.S. Department of Health and Human Services, 2012. URL: <http://rsbweb.nih.gov/ij/docs/guide/user-guide.pdf>.
- [GA10] M.S. Grewal and A.P. Andrews. “Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]”. In: *IEEE Control Systems Society’s Conference* 30.3 (June 2010), pp. 69–78. ISSN: 1066-033X. DOI: 10.1109/MCS.2010.936465.

- [GMM02] S.B. Goldberg, M.W. Maimone, and L. Matthies. “Stereo vision and rover navigation software for planetary exploration”. In: *Aerospace Conference Proceedings, 2002. IEEE*. Vol. 5. 2002, pages. DOI: 10.1109/AERO.2002.1035370.
- [Had02] Jacques Hadamard. “Sur les problèmes aux dérivées partielles et leur signification physique”. In: *Princeton University Bulletin* 13 (1902), pp. 49–52.
- [HB01] Christian von Hardenberg and François Bérard. “Bare-Hand Human-Computer Interaction”. In: *Proceedings of the ACM Workshop on Perceptive User Interfaces*. 2001, pp. 1–8.
- [Hee87] David J. Heeger. *Model for the Extraction of Image Flow*. Tech. rep. 8. General Robotics and Active Sensory Processing Laboratory, University of Pennsylvania, 1987.
- [HS81] Berthold K. P. Horn and Brian G. Schunck. *Determining Optical Flow*. Tech. rep. Cambridge, Massachusetts, USA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1981.
- [HW13] Simon Hermann and René Werner. “High Accuracy Optical Flow for 3D Medical Image Registration Using the Census Cost Function”. In: *Pacific-Rim Symposium on Image and Video Technology*. 2013, pp. 23–35.
- [Kal60] Rudolf Emil Kálmán. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.
- [Ken+09] Farid Kendoul et al. “An adaptive vision-based autopilot for mini flying machines guidance, navigation and control”. In: *Autonomous Robots* 27.3 (2009), pp. 165–188.
- [KN97] Henner Kollnig and Hans Hellmut Nagel. “3D Pose Estimation by Directly Matching Polyhedral Models to Gray Value Gradients”. In: *International Journal of Computer Vision* 23.3 (June 1997), pp. 283–302. ISSN: 0920-5691. DOI: 10.1023/A:1007927317325.
- [Lar05] R.P. Larkin. “Radar techniques for wildlife biology”. In: *Techniques for wildlife investigations and management, 6th edition*. The Wildlife Society, Bethesda, Maryland, USA, 2005, pp. 448–464.

- [LK81] Bruce D. Lucas and Takeo Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, Volume 2*. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.
- [LL01] Ivan Laptev and Tony Lindeberg. “Tracking of Multi-state Hand Models Using Particle Filtering and a Hierarchy of Multi-scale Image Features”. In: *IEEE Workshop on Scale-Space and Morphology, Vancouver, Canada*. Ed. by Michael Kerckhove. Vol. 2106. Lecture Notes in Computer Science. Springer, 2001, pp. 63–74. ISBN: 3-540-42317-6.
- [LMB02] Gildas Lefaix, Éric Marchand, and Patrick Bouthemy. “Motion-Based Obstacle Detection and Tracking for Car Driving Assistance”. In: *International Conference on Pattern Recognition*. 2002, pp. 74–77.
- [Low99] D. G. Lowe. “Object recognition from local scale-invariant features”. In: *Seventh IEEE International Computer Vision Conference*. Vol. 2. 1999, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.
- [Lu+13] Wei-Lwun Lu et al. “Learning to Track and Identify Players from Broadcast Sports Videos”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (2013), pp. 1704–1716.
- [MLB07] Mark W. Maimone, P. Chris Leger, and Jeffrey J. Biesiadecki. “Overview of the Mars Exploration Rovers Autonomous Mobility and Vision Capabilities”. In: *IEEE International Conference on Robotics and Automation*. 2007.
- [MS85] Leonard A. McGee and Stanley F. Schmidt. *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*. Tech. rep. Naval Air Station Moffett Field, Mountain View, California, USA: National Aeronautics and Space Administration, 1985.
- [MU79] David Marr and Shimon Ullman. *Directional Selectivity and its Use in Early Visual Processing*. Tech. rep. 524. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1979.
- [Nag14] Hans Hellmut Nagel. *Image Sequence Server, Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe*. 2014. URL: http://i21www.ira.uka.de/image_sequences/.

- [Nag90] Hans Hellmut Nagel. “Extending the ‘Oriented Smoothness Constraint’ into the Temporal Domain and the Estimation of Derivatives of Optical Flow”. In: *European Conference on Computer Vision*. Ed. by Olivier D. Faugeras. Vol. 427. Lecture Notes in Computer Science. Springer, 1990, pp. 139–148. ISBN: 3-540-52522-X.
- [ON92] G.A. Orban and Hans Hellmut Nagel. *Artificial and biological vision systems*. EUR (Luxembourg). Springer-Verlag, 1992. ISBN: 9783540560128.
- [Pap+06] Nils Papenberg et al. “Highly Accurate Optic Flow Computation with Theoretically Justified Warping”. In: *International Journal of Computer Vision* 67.2 (2006), pp. 141–158.
- [Pap12] White Paper. *Curiosity Rover Will Use Space-Proven Sensors to Safely Navigate Surface of Mars*. Tech. rep. Waterloo, ON, Canada: Teledyne DALSA Corporation, 2012.
- [Pit07] Robert I. Pitts. *Computer Science Department, Boston University, USA*. 2007. URL: <http://www.bu.edu/cs>.
- [Rap+11] Daniel H. Rapoport et al. “A Novel Validation Algorithm Allows for Automated Cell Tracking and the Extraction of Biologically Meaningful Parameters”. In: *PLoS ONE* 6.11 (Nov. 2011), e27315. DOI: 10.1371/journal.pone.0027315.
- [RPP12] Matina Kalcounis Rueppell, Thomas Parrish, and Sebastian Pauli. *Application of Object Tracking in Video Recordings to the Observation of Mice in the Wild*. Tech. rep. Department of Biology, Department of Mathematics, and Statistics, University of North Carolina, Greensboro, USA, 2012.
- [Ruh+05] P. Ruhnau et al. “Variational Optical Flow Estimation for Particle Image Velocimetry”. In: *Experiments in Fluids* 38 (2005), pp. 21–32.
- [Sch13] Christian Schmaltz. *Advanced Image Analysis Lecture*. 2013. URL: <http://www.mia.uni-saarland.de/Teaching/aia13.shtml>.
- [SH89] D. Shulman and J.-Y. Herve. “Regularization of discontinuous flow fields”. In: *Proceedings of the Workshop on Visual Motion*. 1989, pp. 81–86. DOI: 10.1109/WVM.1989.47097.
- [She+09] Huan Shen et al. “Vision Based Navigation System of Intelligent Vehicles: A Robust Object Tracking Approach”. In: *Hybrid Intelligent Systems*. IEEE Computer Society, 2009, pp. 359–364. ISBN: 978-0-7695-3745-0.

- [Sin90] A. Singh. “An estimation-theoretic framework for image-flow computation”. In: *International Conference on Computer Vision*. Dec. 1990, pp. 168–177.
- [SK11] Muhammad Syah Houari Sabirin and Munchurl Kim. “Object Tracking”. In: ed. by Hanna Goszczynska. InTech, 2011. Chap. 8.
- [Sma+08] Ihor Smal et al. “Particle Filtering for Multiple Object Tracking in Dynamic Fluorescence Microscopy Images: Application to Microtubule Growth Analysis”. In: *IEEE Transactions on Medical Imaging* 27.6 (2008), pp. 789–804.
- [SS06] Agustín Salgado and Javier Sánchez. “A Temporal Regularizer for Large Optical Flow Estimation”. In: *International Conference on Image Processing*. IEEE, 2006, pp. 1233–1236.
- [Thr+06] Sebastian Thrun et al. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of Field Robotics* 23.9 (2006), pp. 661–692. DOI: 10.1002/rob.20147.
- [WAB06] Josh Wills, Sameer Agarwal, and Serge Belongie. “A Feature-based Approach for Dense Segmentation and Estimation of Large Disparity Motion”. In: *International Journal of Computer Vision* 68.2 (2006), pp. 125–143.
- [Wed+11] Andreas Wedel et al. “Stereoscopic Scene Flow Computation for 3D Motion Understanding”. In: *International Journal of Computer Vision* 95.1 (2011), pp. 29–51.
- [Wir07] Oliver Wirjadi. *Survey of 3D image segmentation methods*. Tech. rep. 123. Fraunhofer Institut für Techno- und Wirtschaftsmathematik, 2007. URL: <http://kluedo.ub.uni-kl.de/volltexte/2008/2213/pdf/bericht123.pdf>.
- [WKC94] Dan S. Wallach, Sharma Kunapalli, and Michael F. Cohen. “Accelerated MPEG compression of dynamic polygonal scenes”. In: *ACM SIGGRAPH*. ACM, 1994, pp. 193–196. ISBN: 0-89791-667-0.
- [WS01a] Joachim Weickert and Christoph Schnörr. “A Theoretical Framework for Convex Regularizers in PDE-Based Computation of Image Motion”. In: *International Journal of Computer Vision* 45.3 (2001), pp. 245–264.
- [WS01b] Joachim Weickert and Christoph Schnörr. “Variational Optic Flow Computation with a Spatio-Temporal Smoothness Constraint”. In: *Journal of Mathematical Imaging and Vision* 14.3 (2001), pp. 245–255.

- [WWB88] A.M. Waxman, J. Wu, and Fredrik Bergholm. “Convected activation profiles and the measurement of visual motion”. In: *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*. June 1988, pp. 717–723. DOI: 10.1109/CVPR.1988.196313.
- [Yao+10] Angela Yao et al. “Tracking People in Broadcast Sports”. In: *Symposium - Deutsche Arbeitsgemeinschaft für Mustererkennung*. Ed. by Michael Goesele et al. Vol. 6376. Lecture Notes in Computer Science. Springer, 2010, pp. 151–161. ISBN: 978-3-642-15985-5.
- [ZBW11] Henning Zimmer, Andrés Bruhn, and Joachim Weickert. “Optic Flow in Harmony”. In: *International Journal of Computer Vision* 93.3 (2011), pp. 368–388.
- [ZR98] Dusan S. Zrnic and Alexander V. Ryzhkov. “Observations of insects and birds with a polarimetric radar”. In: *IEEE Transactions on Geoscience and Remote Sensing* 36.2 (1998), pp. 661–668.

Additional Sources and Further Reading

- [14] *Wikipedia Category: Computer Vision*. 2014. URL: http://en.wikipedia.org/wiki/Category:Computer_vision.
- [AOM06] Yilmaz Alper, Javed Omar, and Shah Mubarak. “Object Tracking: A Survey”. In: *ACM Computing Surveys* 38.4 (July 2006).
- [Dem14] Oliver Demetz. *Correspondence Problems in Computer Vision Lecture*. 2014. URL: <http://www.mia.uni-saarland.de/Teaching/COPCV14/copcv14.shtml>.
- [JKS95] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine Vision*. 0-07-032018-7. McGraw-Hill Inc., 1995.
- [LS06] Hongyu Li and I-Fan Shen. “Similarity Measure for Vector Field Learning”. In: *ISNN (1)*. Ed. by Jun Wang et al. Vol. 3971. Lecture Notes in Computer Science. Springer, 2006, pp. 436–441. ISBN: 3-540-34439-X.
- [Wei13] Joachim Weickert. *Image Processing and Computer Vision Lecture*. 2013. URL: <http://www.mia.uni-saarland.de/Teaching/ipcv13.shtml>.