Faculty of Informatics
Masaryk University

# Fast Anisotropic Filtering and Performance Evaluation Tool for Optical Flow in Biomedical Image Analysis

Ph.D. Thesis

**Vladimír Ulman**

Brno, 2011

I hereby declare that this thesis is my own original work that I have written singly. To the best of my knowledge and belief, it contains no material previously published or written by another person, except where due acknowledgement is made in the text. All the sources and literature that I have used I cite properly and provide full link to its source.

Vladimír Ulman, January 21, 2011

Supervisor: Dr. Michal Kozubek

# Abstract

The thesis is focused on the analysis of time-lapse images acquired using a fluorescence light microscope. In particular, for the purpose of automated evaluation of motion of stained cell structures, e.g., proteins or cell nuclei, perceived over a time period, we aim towards an object tracking based on an optical flow field. An optical flow method estimates a flow field in which a vector is assigned to every pixel in an image. The vector represents the difference in position of the same pixel content between two images. To track the given position it is then enough to simply follow flow vectors provided good flow estimates are available.

The thesis reviews the process from acquiring image data to methods for computing optical flow. The description starts with the limits of the imaging technology and characterization of the obtained image data. The survey part reviews and discusses methods that allow for conducting object tracking. Optical flow methods based on filtering are then inspected more closely as well as the representation of motion in spatio-temporal images.

Emphasis is put on efficient and accurate image filtering, which is an essential part of the filtering-based optical flow methods. The Gaussian and Gabor filters are studied. Firstly, recursive 1D filtering is analyzed to show it is very fast, efficient and accurate. On the other hand, handling of boundary conditions is somewhat complicated but it is demonstrated to be feasible. Secondly, separability of Gaussian and Gabor filter is investigated resulting in a framework which utilizes many recursive 1D image filtering tasks along generally oriented axes. The framework allows for filtering with general anisotropic Gaussian and Gabor filters. The anisotropy manifests itself with elliptical kernel shape with distinguished main axis. Important achieved result is that this axis can be arbitrarily oriented. The framework is more accurate but slightly less efficient compared to an optimal solution available. Nonetheless, for the target case of Gabor bank filtering a scheme is presented which is shown to give an almost-optimal efficiency.

The fast and more motion sensitive Gabor bank filtering was tested on the original Heeger's optical flow method. The method utilizes bank of Gabor filters and an error function which controls the estimation of flow vector from the collection of filter responses. A preliminary result is given which uses new bank parameters as well as a new error function.

A generator of synthetic sequences of test images with associated ground-truth flow fields was developed and is described in the thesis. The generator works with one global motion layer to move the whole cell and several independent local motion layers to additionally move selected interior cell structures. Movements are described using flow fields which are function of time. Altogether, the generator allows for the synthesis of datasets simulating time-lapse acquisition of complex processes in live cells. Such synthetic sequences are an indispensable tool for the verification of the algorithms that estimate flow field.

# Acknowledgements

I sincerely thank my supervisor Michal Kozubek for leading, encouraging, discussing, helping and supporting me throughout the years. I also wish to thank my colleagues and friends Jan Hubený, David Svoboda, Pavel Matula and Petr Matula for discussions, suggestions and comments and all folks in the Centre for Biomedial Image Analysis for a great atmosphere for conducting research and developement. I would also like to thank my wife, Jana, for patience and understanding during the course of the study — especially, when deadlines were approaching. Last but not least, I thank the Muse for coming...
...and not letting go.

The thesis has been supported by the Ministry of Education of the Czech Republic (Grants No. MSM0021622419, LC535 and 2B06052).

# List of publications

The thesis is based on the following publications. These will be cited as [P*] in the text.


[P1] V. Ulman, "Arbitrarily-oriented anisotropic 3D Gaussian filtering computed with 1D convolutions without interpolation," in *Proceedings of 8th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision*, (Athens), pp. 56–62, 2008. ISSN 1792-4618.

[P2] V. Ulman, "Filtering with anisotropic 3D Gabor filter bank efficiently computed with 1D convolutions without interpolation," in *Proceedings of the Seventh IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, (Calgary), pp. 33–42, 2010.

[P3] V. Ulman, "Boundary treatment for Young–van Vliet recursive zero-mean Gabor filtering," *EURASIP Journal on Advances in Signal Processing*, 2011. Submitted, under 2nd round of review.

[P4] V. Ulman, "Improving accuracy of optical flow of Heeger's original method on biomedical images," in *Proceedings of the 7th International Conference on Image Analysis and Recognition, ICIAR 2010*, pp. 263–273, 2010. LNCS 6111.

[P5] V. Ulman and J. Hubený, "On generating ground-truth time-lapse image sequences and flow fields," in *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, (Angers), pp. 234–239, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007.

[P6] V. Ulman and J. Hubený, "Pseudo-real image sequence generator for optical flow computations," in *Proceedings of 15th Scandinavian Conference on Image Analysis*, (Heidelberg), pp. 976–985, 2007. LNCS 4522.

[P7] J. Hubený, V. Ulman, and P. Matula, "Estimating large local motion in live-cell imaging using variational optical flow," in *VISAPP: Proc. of the Second International Conference on Computer Vision Theory and Applications*, pp. 542–548, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007.

# Contents

# Chapter 1

# Introduction

## 1.1 Live cell studies

Computer science has found its way into many established scientific fields. A cell biology has not became an exception. And similarly to other fields, computers, as the visible acting tool of the computer science, have settled in most, if not all, aspects of the field. Quite often, it is not a typical desktop computer, which we are used to, but it may take a form of a small embedded microchip which we would certainly find in almost any electric device in the biological laboratory. But computer science is not only used to design a specific piece of programme to drive some instrument, even if the programme is allowed to make decisions and act differently as a result of different values read from instrument's sensors. Computers, so to speak, store the state-of-the-art knowledge. They help to navigate within it, help to find relevant information, help to discover functional dependencies, help to discover repeating patterns (consider, for instance, the human genome which is known to consist of roughly 20.000 genes), etc. They even aid in the planning of biological experiment by performing simulations and suggesting what probes to use, for instance. These remarks are examples of what is called data processing and data analysis.

In the cell biology, as the name suggests, we deal with cells. Unfortunately, cells are naturally invisible to the human naked eye. Technical reason for that is that the size of a cell is simply too below an eye resolution, i.e., the size of the smallest distinguishable patch that one can typically see is greater than any cell. A workaround has emerged when a microscope was invented. A microscope, as a magnifier, enabled us to *see* cells. Various modifications to the microscope and various specimen preparation techniques have been developed in order to enable observation of different cells, different parts of them and even different processes in the cells. Anyway, the information acquired from the microscope is always a visual one. Hence, *image* is the primary representation of information in the cell biology when microscopes are used. This has paved the way for the fields of image processing and image analysis to have became significant representatives of the utilization of the computer science in the cell biology.

We have mentioned the topic of various types of microscopes and various imaging techniques. Different modifications of the microscope have gradually appeared over the years [1] until we have arrived to an optical microscope equipped with a motorized stage, motorized revolver with optical filters, digital camera and a confocal unit, eventuelly. Each of its components is interconnected with a computer. This shall allow for unsupervised

automated acquisition of the specimen: long-term (over-night) study with many periodically taken images by the digital camera. Images of individual cells from different portions of a specimen may be obtained using motorized stage (lateral movement) or automatic autofocusing may be implemented as well if the stage allows also for a movement along the optical axis (axial movement). These are only a few examples of possibilities of modern acquisition in the cell biology. The selection of installed hardware is closely bound with the imaging technique that is going to be used [2, 3]. Problem in the cell biology is that the cell itself is transparent [4, 3]. It is merely like a transparent sticker attached to the glass. The sticker is very difficult to observe if it is not enhanced in some way, e.g., coloured or stained. We just simply see through it otherwise. Therefore, cells or mostly only their interior parts, nuclei or chromosomes for instance, are often stained. Staining with a fluorescent dye is a popular approach to enhance certain cell material [5, 4, 6]. The principle is that the incoming light from a microscope light source excites the fluorescent dye, the dye then emits some light which is collected and directed to the digital camera [7]. The use of optical filters, namely the excitation filter to reduce frequency band of the incoming light and emission filter to reduce frequency band of the outcomming light are key factors to achieve a good performance of the technique [5]. In addition, reasonable combination of the filters and dyes allows to stain the specimen with two or more dyes, each of which will enhance different parts of a cell. By switching the filters in an optical path one may capture independent images of the same portion of the specimen, each time with different parts visible. The images are artificially composed together in the computer afterwards so that a mutual relation of cell parts can be studied. Acquisition of cell images using this particular type of microscopes with the technique of fluorescence staining is denoted as *fluorescence light microscopy*.

The most important property of the fluorescence imaging is that it shows physical spatial structure of a cell [8, 5]. This is especially well suited for applications such as studies of the spatial structure of chromatin as well as function-structure relationship in human genome, changes of this structure and/or function during cell cycle, differentiation and between healthy and cancer cells, studies of the function, dynamics and interactions of selected proteins in live cells or finding new biomarkers which make difference between normal healthy cell and cancer cell. Among these we focus on the live cell studies in which one typically deals with a sequence of fluorescent images that are acquired periodically over the time, the time-lapse sequence. As a consequence, the time dimension is introduced into the image data enabling new views of explored cells. For example, we may measure the speed of movement, directly observe important pathways of fluorescence proteins or indirectly observe pathways of a material to which the fluorescence proteins are attached, compare the spread of chromatin during the cell division or just observe the growth of cells. On the other hand, the possibility of observing a cell over a certain, even small, period of time is traded for a few new drawbacks which do not occur in fixed fluorescence cell studies, e.g., the FISH or immunofluorescence techniques. The most apparent one is that the staining and then observation of cell material must not lead to an immediate death of the cell or to abrupt changes in the cell behaviour [3]. This was rather limiting earlier but nowadays it is mostly overcome by labelling (or tagging) live cells with fluorescent proteins [9, 4]. The acquisition of stained live cells is also called the time-lapse fluorescence light microscopy.

Once the biological process is captured by means of a sequence of images, we would

like to analyze it. We may think of two fundamental types of problems (or tasks) [10]. The first one is related with the analysis of single still image. It contains image processing tasks [11, 12, 6], such as to improve the quality of an image (e.g., noise removal, edge enhancement) or as to extract objects in the image (e.g., segmentation of whatever is demanded, be it the cell itself, nuclei, mitochondria or proteins, to name a few), and image analysis tasks, such as to extract features (e.g., measure volume of a cell, density of chromatin, counts of telomeres) or analyze mutual relations (e.g., detect co-occurence of proteins suggesting they interact). The second type is related with the sequence of images and with a semantic link between them such as associations or correspondence. The main task here is to track structures in images which gives us the ability to say a certain object in one image corresponds to a certain object in some other image in the sequence [13, 14]. Or, this can also be important, to be able to say the object that appeared previously is no more present in the following images. Both types of problems are equally important. Consider, what good would it be to extract features from all single images in a time-lapse sequence while not being able to show their development over the time (because of the links missing)? On the other hand, how can we establish a link between two images when we have no augmented regions to link between (because of the segmentation lacking)? This is the chicken-egg problem. This is perhaps the reason why solutions from any of the two sorts are often half-way inbetween them when the analysis of time-lapse images is considered. That is, one has to tackle both segmentation and tracking in the solution as in [15, 11, 16, 14, 17, 10], to give only a few references.

In this thesis we shall consider a particular method of optical flow computation, which belongs to the second type entirely [18]. This is because the method, like any other optical flow computation method, provides results that are only preliminary in terms of a tracking application. The method results must be further processed in order to make the tracking complete but, due to the chicken-egg, that would involve some segmentation method to extract regions in images to track.

## 1.2 Topics of the thesis

### Optical flow for tracking and other applications

We will focus on the computation of an optical flow. An *optical flow* method assigns a vector to every pixel in an image. Note that the term pixel is used to denote the smallest picture element. The vector represents the difference in position of the same pixel content between two images, it is called the *flow vector*. In the image sequence, the content seems to flow or drift along the assigned flow vectors. Altogether they constitute a flow field for a given pair of images. Since the flow computation works directly at the level of image pixels, it more or less tells motions of anonymous masses in the image sequence [19, 11]. The flow field describes an optical (or visual or percieved) flow of displayed content, or mass or simply brightness patterns, from one image to the other. There is no explicit reference stating what the mass represents (e.g., nuclei, groups of proteins, single mitochondria), that would be a task for image segmentation. There is also no explicit reference relating masses, or already recognized structures, in the consecutive images, that would be a task for tracking. The flow field, if it is computed correctly, really only expresses what we percieve at the first sight when we inspect a time-lapse sequence.

In this work we aim to adapt the optical flow computation to the images from time-

lapse fluorescence light microscopy. The aim is to provide flow fields of such a quality that enables to design a simple yet reliable tracking application based on flow fields. The tracker should be given initially a set of points (positions) in the image. Possibly, each point would represent an object of biological interest. It should then move the points (adjust the positions) using the intermediate pre-computed flow fields automatically so that, eventually, the points would follow objects that they are supposed to represent. We won't deal with the methods to automatically select points from given image, it is beyond the scope of this work.

There is also a bunch of other interesting applications in image processing and analysis that would benefit from correct flow fields, i.e., from the information of what has happened between the two consecutive images [20]. Some couldn't be done without such information. For example, a computation of average velocities or acceleration of certain cell material after the cell was subjected to some infection, for instance. This really requires to know the change of coordinates to assess the distance travelled during the given amount of time. For some applications, using the flow field represents an alternative solution, which is another motivation to the effort of computing a flow field. For example, we may fit the flow field to some model of global motion (e.g., translation, rotation, cell contraction or expansion) to correct for global motion, i.e., subtract the modeled global motion flow field from the computed one, to study remained local motions within the cell easier. According to the flow field we may warp a result of segmentation, which is typically represented as a mask image, to prepare a good initialization for the segmentation of the consecutive image, etc. Indeed, optical flow is often used for 2D motion-based segmentation methods [21]. But we won't demonstrate such applications of optical flow in this thesis. Here, we regard the optical flow field computation just as a tool to support methods that link, either manually or automatically, augmented (e.g., segmented) content between images in a time-lapse sequence.

### Intensive Gabor filtering and the energy-based optical flow method

Among the optical flow computation methods we will put emphasis, for reasons to be explained later in Chapter 3, on methods based on an intensive use of image filtering. Namely, we will revisit the method originally proposed by David J. Heeger. For the moment let us just state that the methods based on image filtering represent one branch of optical flow computation. The underlying theory is inspired by some early stages in the processes in the human vision [22, 23, 24, 25]. Basically, the current belief is that there are several receptors in the early stage of the human vision. Different receptors are sensitive to different orientations of patterns and different velocities of patterns. The perception is based on the responses. It is expected to form in some middle stages [26, 27, 28, 29]. Anyway, in the flow computations we simulate the function of receptors with image filtering with Gabor banks. Traditionally, researches used only a few filters. We think the reasons were twofold: 1) the limiting memory resources in the computers, since filtering with every single filter stores its result in a new copy of the input image, and 2) the unavailability of efficient filtering schemes for specialized Gabor filters, since filtering is often time consuming except for a few simple cases. The first barrier is no longer an issue as the amount of installed memory in a personal computer regularily increases and got already several times above the size of images acquired with current digital cameras used in the fluorescence microscopy. The second barrier was broken just recently when algorithms

for efficient Gaussian and Gabor image filtering were reported [30, 31, 32, 33]. The thesis includes one such algorithm for filtering with Gabor banks with increased sensitivity. As a direct consequence, optical flow based on filtering and consequent image analysis tasks can be conducted easily and immediately on a common personal computer found in every office or laboratory.

**Generator for evaluation of optical flow**

Last but not least, we have developed a unique generator of artificial time-lapse image sequences accompanied with flow fields for the purpose of evaluation of optical flow methods. The flow fields serve the purpose of a correct data, often termed as a *ground-truth data*. It allows for comparison and quantitative evaluation of computed flow field with respect to a correct flow field, so that one can immediatelly see how good an algorithm is performing on the given test data. This is done predominantly by computing an angular error between a computed flow vector and a ground-truth vector, it is then averaged over all vectors in the image. Such benchmarking was intensively used in the results section of the thesis.

## 1.3 Outline of the thesis

The format of the thesis is such that the text is extended with full reprints of the original publications (co-authored or solely authored by the author of this thesis). The publications constitute significant part of the contribution of this thesis. The text itself aims to provide a comprehensive introduction to the these by providing broader context to the problem in question, more elaborated introduction, motivation and related work, additional reasoning and justification to the published methods, etc. We advice the reader to approach each topic by first reading the introductory part of the respective chapter, then the original publications at the end of it and then continue with the rest of the chapter. Summaries to all publications are given in Sections 4.2 and 5.3.

The content of the following chapters is as follows. We start, in Chapter 2, with giving more details on the process of acquistion to make description of the nature of input time-lapse images easier, which is important to understand some arguments in the next chapter. Chapter 3 discusses ways to achieve tracking and arrives to the conclusion that it is worth trying to approach it via the optical flow. An overview of techniques of optical flow computations is then presented. Reasons for the selection as well as necessary underlying theory of the particular method are given. This chapter represents the survey part of the thesis whereas the following two chapters represent the practical part and the two main contributions of the thesis. In Chapter 4 we propose a method to separate spatial Gaussian filtering, extend it to complex spatial Gabor filtering and, finally, propose a new scheme for efficient spatial filtering with specialized Gabor banks. We continue with the Heeger's optical flow method, for which the Gabor bank filtering plays a central role both in terms of time consumption and accuracy. We propose two main modifications to it. Chapter 5 overviews means to optical flow accuracy measurements and surveys tools to achieve it. It then describes design, control and future development directions of our in-house developed generator of artificial time-lapse microscopy images with ground-truth flow fields. The text of the thesis concludes with summary. The thesis concludes with reprints of the original publications.

# Chapter 2

# Time-lapse fluorescence microscopy

So far, we have outlined possibilities of live cell studies: we tried to give principle of the imaging technology based on the fluorescence phenomenon, we tried to give examples of what sort of biological experiments we can observe with it and, also, we tried to show what sorts of computer science problems may arise when implementing a fully automated analysis system.

In this chapter we would like to illustrate typical properties of input data that we deal with. We would like to do it by exploring the imaging technology to the necessary level of detail and pinpointing its aspects and limits.

## 2.1 Acquisition using an optical microscope

### 2.1.1 Specimen preparation

In order to make use of the fluorescence phenomenon in live cell studies, cells must be prepared in advance. A special biological marker is inserted into a cell by means of inserting a special gene sequence to a cell DNA to force it to produce the marker during its lifetime and, sometimes, in its next generations. The marker is known to attach only to certain intracellular part, quiet often to some protein, which we then observe in an experiment. We then choose which part to observe by choosing the right marker [4, 34]. Each marker is visually different or unique to fullfil its role: to outline or designate the selected part of cell in order to ease its observation in a microscope. In our case, we make use of fluorophores or fluorochromes, which are terms commonly used for markers with the ability to fluoresce. They enable us, under appropriate circumstances that we shall discuss in the next section, to obtain images in which markers are displayed with substantially stronger intensities than the rest of a cell. In other words, bright patches in obtained images are showing accumulations of markers. And since the distance between marker and target molecules is smaller then resolution of an optical microscope, we assume that exactly the same bright patches in obtained images are, in fact, showing the selected intracellular parts as well [35]. This is how flourochromes visually enhance parts they attach to.

In addition to the requirements given above, the inclusion and presence of a marker must be harmless to cells and must not influence their behaviour [3]. Although it seems

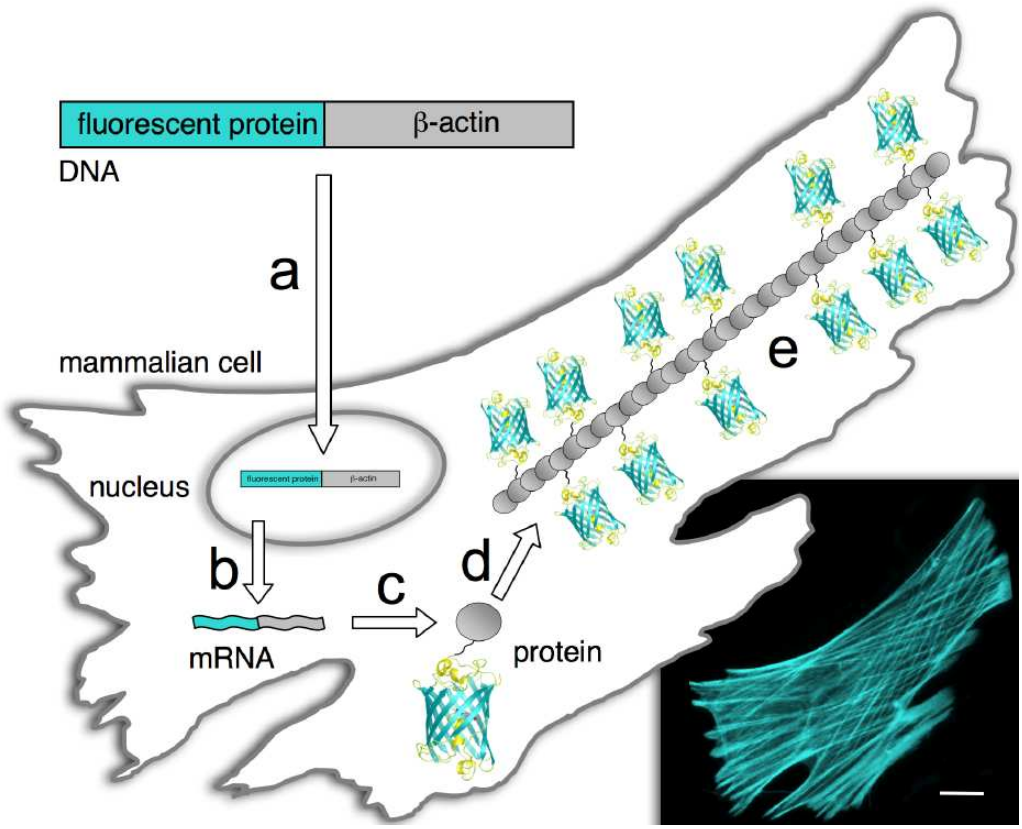


Figure 2.1: A schematic representation of how a monomeric fluorescent protein is employed for imaging of $\beta$-actin. A mammalian cell is transfected (a) with a cDNA chimera composed of a fusion of the genes encoding the fluorescent protein and $\beta$-actin. The gene is transcribed (b) to produce mRNA that is then translated (c) to form the chimeric protein. The trafficking and localization (d) of the protein is dictated by the protein-of-interest, the fluorescent protein, ideally, does not interfere. In the case of $\beta$-actin, the chimera is incorporated into actin filaments (e) along with the endogenous protein. Shown in the inset is a fluorescence image of a gray fox lung fibroblast (FoLu) cell that has been transfected with mTFP1-$\beta$-actin [36]. Reprinted with permission from the Scholarpedia [37].

as a natural requirement on the marker properties, it is common only in the live cell studies. In the following text, we will come across another, rather more implicit, additional requirements on the fluorochromes that will emanate from the used technology. For example, we would like to use fluorochromes with narrow excitation and emission bands or with increased resistance to photobleaching (light emitted from a fluorochrome fades proportionally to the time the fluorochrome is being exposed to the excitation light).

A foundation to markers, that fulfil the most of the requirements listed above, was set in early 1960s when Shimomura *et al.* [38] managed to isolate a naturally fluorescent protein, the now-famous green fluorescent protein GFP, that was previously observed as a green fluorescent substance in jellyfish [39]. It took another 30 years until researchers managed to clone GFP [40]. Finally in 1994, Chalfie *et al.* were the first who used GFP in an experiment with live cells [41] and started the "green revolution" [9]. The marker gene sequence is appended to the sequence of target protein and inserted into a living

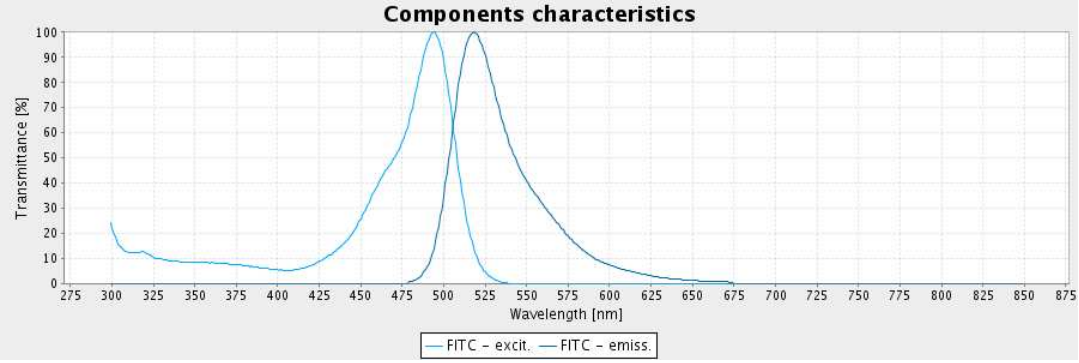


Figure 2.2: Example of excitation/emission spectra of the FITC fluorescent protein. Note that emission peak is positioned at longer wavelengths. Figure was generated with the CBIA's "Optic: Modelling and optimization of the light throughput of an optical system" at http://cbia.fi.muni.cz/optic.

cell. The cell then produces original protein joined together with the fluorescent one, see Fig. 2.1. Since 1994, in a relative short period of time, wealth of fluorescent markers, also called fluorescent probes, has become available for live cell studies [4, 34]. Different markers allowed biologists to tag nearly any protein they may desire. They also differ in their excitation and emission frequency bands allowing to use more markers at the same time. For example, there exists BFP (blue) or CFP (cyan) fluorescent proteins. Refer to Rizzo *et al.* [34] for a recent overview of fluorescent proteins available.

### 2.1.2 Microscope components

The selection and constitution of components in a fluorescence microscope is determined by the fluorescence phenomenon: a material absorps irradiating light (often termed the excitation light) and immediately emits its own light. But to make use of the fluorescence, it is vital that emitted light has different wavelength than excitation light (that the particular fluorescence material is sensitive to) in order to be able to distinguish between the two types of lights. In fact, George G. Stokes was the first to find out that emission is always at longer wavelenghts. This is reffered to as the Stokes shift, see Fig. 2.2 for an example of excitation/emission spectra of the FITC fluorochrome. For detailed explanation of the phenomenon of fluorescence we refer to Wolf [8].

#### Main components of a fluorescence microscope

The main components of a fluorescence *wide-field* optical microscope are depicted in Fig. 2.3, refer to Stephens and Allan [42] or Kozubek [2] for detailed explanation of modern fluorescence microscope setups. There may exist other variants of fluorescence wide-field microscopes, the figure shows a reflected microscope for instance, but the acquistion principle is always the same. The light passes from the light source through excitation filter into a condenser. The excitation filter narrows the frequency spectra of the light source by suppressing some frequency bands. In the condenser, the incoming relatively spread beam of light is condensed into a narrower and hence more intensive beam. The light then approaches specimen. Some photons from the incoming light pass through the spec-

imen. Some are reflected into various directions, eventuelly into an objective or even back into the condenser. Some are absorbed by fluorescent molecules which, in turn, emit new photons (with longer wavelengths) again into all directions. Portion of the reflected irradiating light and emitted light is collected by the objective and continues through an emission filter into a detector. The emission filter (sometimes also reffered to as barrier) blocks some frequency bands preventing them to reach the detector. For the detector, a photomultiplier tube (PMT) or a CCD camera are the two most often used devices [1, 43, 42, 4, 3]. In the following and the rest of this thesis, we will always consider only the use of a CCD camera — even for the case of confocal microscopy. This follows the concept introduced by Kozubek *et al.* [44] and later evolved by the same authors [45, 46].

Depending on the position of the objective we distinguish between transmitted and reflected microscopes. In the former one, the light that passes through the specimen is collected what makes the condenser and objective appear on opposite sides of the specimen. In the latter one, the light that is reflected towards the condenser is collected. This is a preferred setup because the fluorescence microscope objective can serve also as a well-corrected condenser. In fact, they are a single component and, as such, a reflected microscope easily has the objective/condenser always in perfect alignment. On the other hand, the light from specimen now travels the same path towards the light source. To allow it to reach the detector, a beam-splitter in the form of dichroic mirror (also termed dichromatic mirror) is inserted between the excitation and emission filters and objective, just like in Fig. 2.3. The dichroic mirror reflects some wavelengths while other wavelengths are transmitted. Ideally, the reflected wavelengths match excitation intervals of often used fluorochromes and, in the same fashion, the passed-through wavelenghts match their emission intervals. Since reflected microscopes place condenser/objective on one side of a specimen, it may just be below or above a specimen allowing to look at a specimen from bottom or from top. The placement below a specimen is, perhaps, the most common in live cell studies because we may easily cover a specimen with a special chamber with special athmosphere in order to provide living cells with more suitable environment from their point of view. It is referred to this setup as to the inverted microscope.

The combination of excitation and emission filters acts as selector for what is going to be imaged when, for instance, a specimen is stained with more dyes. The selection is realized, at first, by letting pass a light only of proper (excitation) wavelengths such that only certain fluorochromes may fluoresce and, at second, by blocking unwanted light heading towards the detector. This is also useful to suppress specimen autofluorescence. Autofluorescence is an effect when some cell material naturally fluoresce, without our additional intervention, when irradiated with light of proper wavelengths. If the light source would include such wavelengths, the specimen would autofluoresce and possibly interfere with light emitted from inserted fluorochromes. The filters are especially important in microscopes equipped only with source of white light, i.e., light comprising of wide range of wavelengths. The dichroic mirror may additionally "post-narrow" the spectra of irradiating light as well as it may additionally "pre-narrow" the spectra of light travelling towards the detector. This is due to the reflectance/transmittance characteristics of the mirror.

In some rare cases, one or both filters may be omitted from the optical path. For example, excitation filter may be omitted if a single-wavelength laser source is used. Similarily, if a detector sensitive only to certain interval of wavelengths is employed, the emission

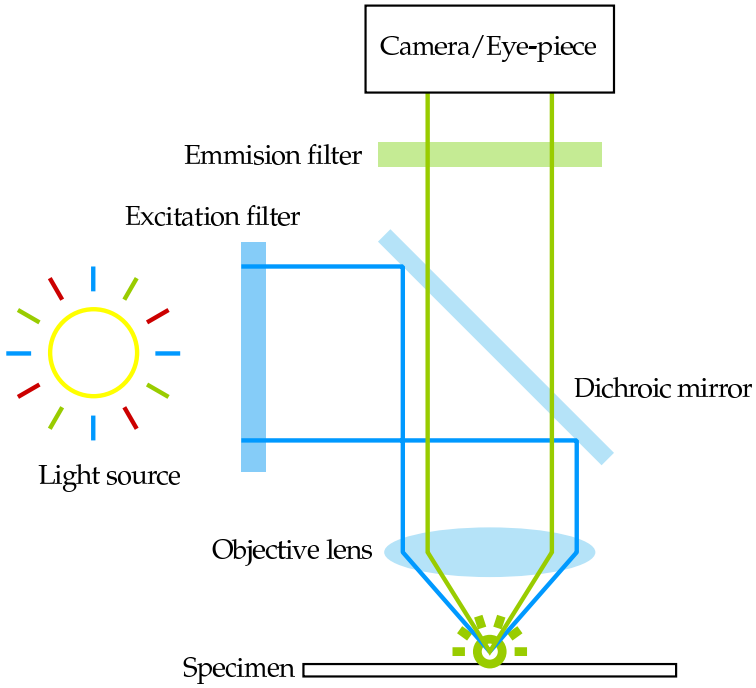


Figure 2.3: Typical components and illustration of their typical arrangement on the optical path in a fluorescence wide-field reflected microscope. The irradiating light from source passes through the excitation filter. After it is reflected from the beam splitter, typically a dichroic mirror, it travels into condenser/objective and through a specimen where it causes emission of new emitted light. The light is then transmitted through the beam splitter, through the emission filter and reaches the detector. The objective works also as a condenser in this setup. Reprinted with permission from Jan Hubený's dissertation [47].

filter may be omitted as the detector already filters the other wavelengths by ignoring them. However, modern fluorescence microscopes are usually equipped with a small number of excitation and emission filters in the body (or chassis) of a microscope by means of, possibly motorized, filter-cube revolvers or filter wheels [1]. The selection of filters is usually driven by excitation/emission spectra of fluorescent dyes one is expecting to use in her biological experiments. Fluorochromes employed in the dyes should have reasonably separated excitation/emission spectra. They also should respect reflect/pass-through properties of the dichroic mirror, or vice versa.

### Confocal unit

Owing to the transparent nature of cells, we may improve an optical microscope with a, so called, *confocal unit*. It is a piece of hardware that enables microscope to acquire thin optical sections from a specimen by inserting an opaque plate with a tiny hole in it into the optical path. The purpose of the pin-hole is to let all light from the focal plane pass while blocking any light emerging from other depths in a specimen (depths are understood in the direction of the optical axis, the $z$ axis), see Fig. 2.4 for illustration of the principle. The focal plane is a plane perpendicular to the optical axis with the property that objects from this plane appear the sharpest in a detector compared to images of objects from other depths. Clearly, the smaller the pin-hole diameter is, the narrower the optical section is because less light from planes further to the focal one reaches the detector. Of course, the pin-hole must be carefully positioned such that it enables passing of light right from the focal plane and not from any other. Regarding the terms often used in microscopy with respect to the coordinate system, we call the *axial direction* the one which is parallel to the optical axis and we call the *lateral direction* or lateral plane the one which is parallel to the focal plane. This imaging mode is called the *confocal microscopy* while the traditional one (without the confocal unit) is called the wide-field microscopy. A consequence for the wide-field optical microscopes is that the acquired image in the detector is formed with contribution of photons emerging also from other planes than the focal one. And since images of the other planes are slightly blurred in the detector, the acquired image is somewhat blurred as well. Again, the necessary condition for confocal microscopy is that the specimen must be transparent. If it were not transparent, the incoming light would be reflected or absorped by specimen surface. The light wouldn't be able to penetrate into a specimen. In fact, the confocal effect may be used, not necessarily only, whenever the fluorescence staining is used because of the same prerequisities.

Problem with the use of confocal effect is that the pin-hole blocks most of the irradiating light. This is because the pin-hole diameter is very small, usually on the order of tens of micrometers. Moreover, single pin-hole enables to acquire small patch of a specimen at certain lateral (and axial) position. In order to assess the whole 2D image of specimen at certain (axial) depth, one has to move the pin-hole within the lateral plane, typically by means of meander scan, so that the whole visible area of specimen is covered. An elegant solution was invented by Nipkow in 1884 [48] and adapted for optical microscopy by Petráň in 1968 *et al.* [49]. Nipkow originated a rotating opaque disc that nowadays has thousands of pin-holes in it spread over the disc such that any two pin-holes are sufficiently far away from each other not to disturb the blocking effects and such that they altogether cover the whole visible area of specimen during revelation of the disc, see the

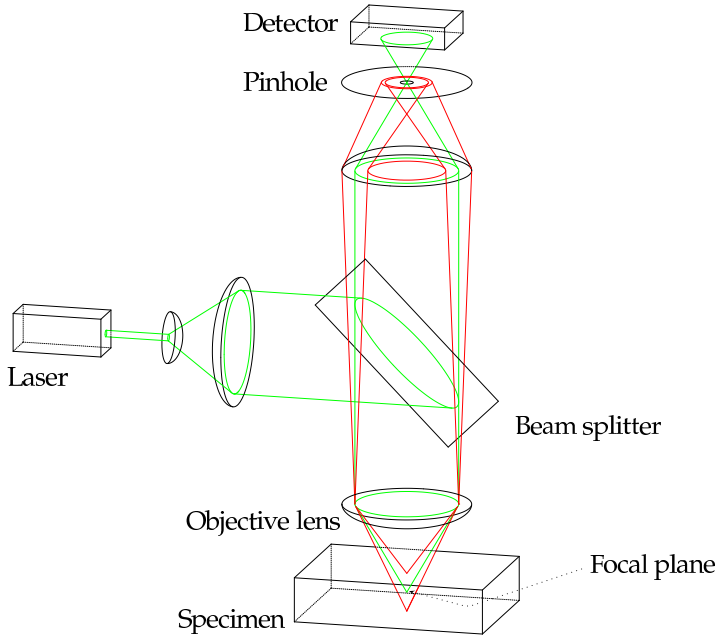


Figure 2.4: The principle of confocality. The light from source irradiates a specimen. The returning light carries images of whole range of thin optical sections, i.e., thin planes within a specimen perpendicular to the optical axis. Photons from other than some selected plane hit the opaque surface of the plate with pin-hole. Only a thin optical section is imaged in the detector in this way. Reprinted with permission from Jan Hubený's dissertation [47].

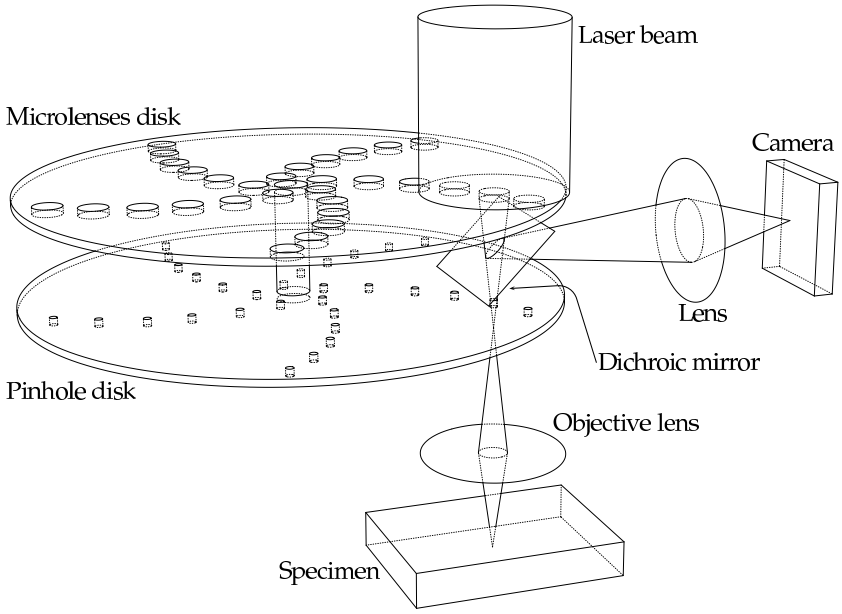


Figure 2.5: The principle of microlens Nipkow rotating disc. The light from source is focused with microlenses in order to increase its intensity. It then passes through a dichroic mirror into corresponding pin-hole. The arrangement of microlenses and pin-holes on both discs is identical, only diameter of microlenses is several times greater than diameter of pin-holes. Both discs are mounted on a common shaft keeping their rotation synchronized. The light then travels into condenser/objective and through a specimen where it is reflected and travels exactly the same path back. Only the reflected light from focal plane (and small lateral and axial proximities) squeezes through some pin-hole in the disc. It is then reflected by the dichroic mirror and reaches a detector, typically a CCD camera. Reprinted with permission from Jan Hubený's dissertation [47].

Pinhole disk in Fig 2.5. The more pin-holes are available, the greater part of specimen is imaged in a detector simultaneously. The meander movement of pin-hole is replaced by revelation of the disc. This makes the whole acquistion faster but requires a 2D detector to be used, e.g., a CCD camera. Originally without the microlenses disk, only about 5% of the irradiating light reached a specimen [45], what means that the sum of areas of all pin-holes represents about 5% of the visible area in the microscope. As the image of the specimen is captured in portions of 5%, the exposure time is, therefore, 20 times longer to capture the whole image compared to the wide-field mode. With the advent [50] of, so called, microlens Nipkow disc the light throughput had increased up to 40–60% [46], see Fig. 2.5. The total area of pin-holes, however, remained approximately the same. Only the intensity of light was locally increased, Fig. 2.6, allowing to, possibly, shorten the exposure time in order to obtain the same amount of excited photons from a specimen. As the area of pin-holes was kept but the exposure time was shortened, the overall acquisition time is shorter compared to the classical Nipkow disc but still longer compared to the wide-field mode.

Note that even when the confocal unit is used, acquired images are always two dimensional (2D). A 3D image of specimen may be formed by acquiring 2D images at con-

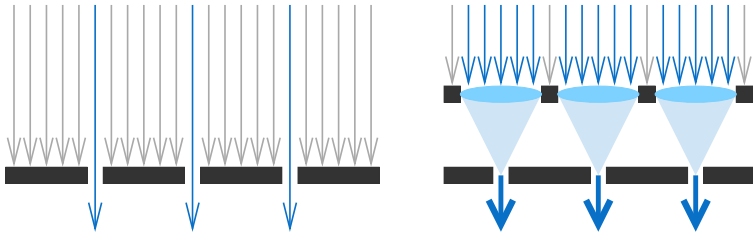

Figure 2.6: The comparison of light throughput in confocal units with Nipkow rotating discs with or without the use of microlenses. The drawing in the left-hand-side shows classical configuration only with the Nipkow disc. Most of the incoming light simply hits the opaque surface of the disc. The drawing in the right-hand-side shows two discs: the upper one with microlenses of diameter several times greater than diameter of pin-holes that are in the lower disc. Significantly greater portion of incoming light is focused to pin-holes resulting in a lot more light reaching a specimen. Reprinted with permission from Jan Hubený's dissertation [47].

secutively changing depths and then stacking them one above the other, just like plates are stacked in a kitchen cabinet. A microscope must be equipped with a moving stage or moving objectives in the axial direction to be capable of acquiring optical sections positioned at different depths. For further reading on the topic of time-lapse confocal (3D) microscopy general requirements, limitations and directions in image analysis and visualization we refer to the overview publication by Gerlich *et al.* [43].

### 2.1.3 The acquisition

We have observed that though the principle of visually enhancing some biological material with the use of fluorescence is fairly simple, the selection for correct and functional microscope setup to make use of it is rather difficult. The greatest deal of guilt in this situation is, perhaps, attributed to the "analog" nature of fluorescent dyes and, partially, of the optical components as well, see Fig. 2.7. In this figure, we observe that there is relatively significant, and not unusuall, intersection in the excitation and emission wavelenghts of the FITC fluorescence dye. The intersection reveals very narrow band from which, in the ideal case, the excitation filter should start blocking wavelengths present in an irradiating light and/or the emission filter should stop blocking wavelengths present in the reflected light — similarily to the way the S630_60 emission filter does in the figure. All microscope components' characteristics should rather precisely correspond with characteristics of fluorescent dyes biologists would like to use [3]. On the other hand, biologists must plan their experiment with respect to available dyes that respects available light source, filters and also dichroic mirror's properties and, of course, that binds proteins of interest in a cell. Sometimes in studies where two or more different intracellular objects are to be observed, a fluorescent dye must be used to stain one object even though a more appropriate dye exists only to meet the above requirements and to avoid interferrence with a dye used to stain the second object, or vice versa. Thus, the acquisition just has to start with a compromising setup sometimes.

Unfortunatelly, this is not where all difficulties end. Even during the acquisition itself, there exist two dominant obstacles closely connected with the fluorescence microscopy: the *photobleaching* and the *phototoxicity* [43, 3]. The former is attributed to the fact that

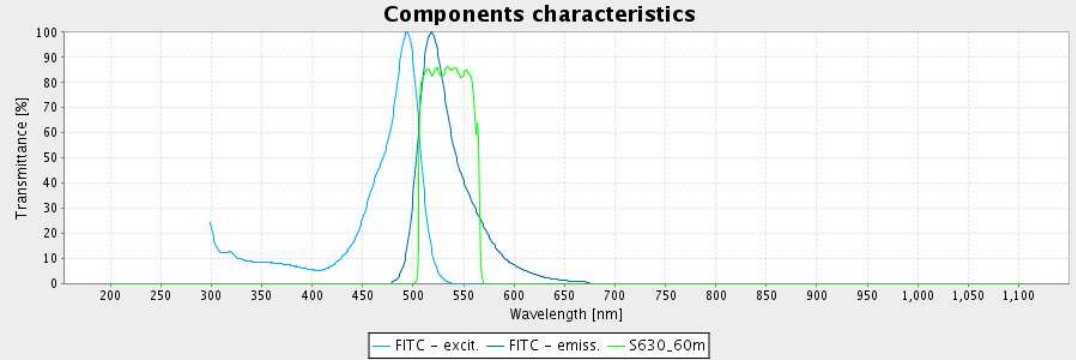


Figure 2.7: Example of excitation/emission spectra of the PubSpectra FITC fluorescent protein and pass-through characteristics of the PubSpectra S630_60m emission filter. Note that the filter is tunned for this fluorescent protein. Figure generated with the CBIA's "Optic: Modelling and optimization of the light throughput of an optical system" at http://cbia.fi.muni.cz/optic.

fluorescence dye can be understood as a container of light photons that the dye emitts as long as it is exposed to the excitation light until the container is empty [8]. The latter refers to the fact that cells are normally not used to be exposed to intensive light and so they may change their behaviour when irradiated. As a result, the exposure time should be as short as possible in order not to waste all photons from the container, i.e., keep some for a next shot, and not to change cell behaviour significantly. On the other hand, short exposure time forces less photons to be emitted from the container resulting in faint images in the detector. A partial solution is to increase intensity of the irradiating light since it increases intensity of the emitted light but this works only to some extent, i.e., there is always an upper bound on the amount of released photons per unit of time [8].

While the problem of appropriately designing biological experiment with respect to imaging capabilities is common both to the wide-field and confocal microscopy, the photobleaching and phototoxicity are subjects of serious concern mostly only in the confocal microscopy [43, 3]. This is because the confocality is achieved only by blocking light comming from other axial distances. In other words, the whole specimen is irradiated and bleaches while only a 2D image of one optical section of specimen is being formed in a detector. To acquire the whole volume of optical sections, a specimen is irradiated several times longer — more than it would be in the wide-field mode. Hence, in the setting for live cell imaging the exposure time is merely a compromise between quality, e.g., signal-to-noise ratio, of every single acquired 2D image and the number of such images possible to obtain in total. Note that the total number of such 2D images in the whole time series is a multiple of the number of slices (the 2D images of optical sections) captured at single time instant and the number of time instants required. The first multiplicand controls the distance between neighboring optical sections (axial resolution, in fact) while the second multiplicand controls the delay between two consecutive time instants when total duration of the process to be "recorded" is known. The latter one is crucial for this thesis as it controls the temporal sampling rate. Generally, since acquisition of single 2D image in the confocal mode lasts longer than in the wide-field mode and, on top of it, several such images are required to form the stack of images (the volume) at certain time instant, the wide-field mode is often preferred to the confocal [43, 42]. Especially, it is so

when observed process is relatively dynamic, e.g., fast movements or rapid division occur, etc. In such cases the confocal mode may be simply too slow. In case when an observed structure is rather flat (in the lateral plane) without significant deformations along the $z$ axis (in the axial direction), we may afford obtaining only 2D images over time but all of improved quality.

## 2.2 Characterization of acquired/input image data

### 2.2.1 Properties of image data

Images we typically deal with owe most of their nature to the fact that they were acquired at the bleeding edge of microscope capabilities. We tried to give an insight what are an optical fluorescence microscope components and what are their mutual relations in terms of components' optical parameters. We saw that each component is indispensable. Unfortunatelly, each introduces some small attenuation [8] and some kind of noise or error into the process of formation of acquired image [42, 1].

The optical components, e.g., lenses, objective or filters, typically introduce aberrations in obtained images. The most important are the non-ideal point spread function (PSF) representing monochromatic aberrations and chromatic aberrations [1]. The former one describes how a point in a specimen is imaged with the optical system. In general, it follows from this function that images of points are a bit blurred in the lateral direction and more blurred in the axial direction [2, 6]. Correspondingly, microscopes have worse axial resolution than lateral resolution. Clearly, axial resolution is relevant only to 3D images, i.e., when confocal unit is used. The chromatic aberration exists due to the fact that light when transmitted throughout lens is refracted differently depending on the light wavelength. The same object when stained with two different dyes is possibly rendered at slightly different positions in a detector because different staining involves use of probes with different emission wavelengths. This is important when one is investigating mutual positions of stained objects, so called co-localizations studies. These require that cell is acquired twice, each time with microscope set up for acquisition of the given fluorescent probe, each time it is stored in a different colour channel of the acquired image. Note, however, that in this thesis we are focused on analysis of movement in the single channel. Hence, we don't have to pay attention to such inter-channel errors. Similarily, we will not focus on the process of reverting the errors due to the point spread function. This is a job of a process called deconvolution and it is beyond scope of this work.

The detector, we assume a CCD chip, typically introduces great deal of noise because the less light reaches the detector, the more apparent the noise is. And since, as we have seen in the text above, the amount of light is greatly decreased by short exposure times and by the properties of the optical system, the presence of noise introduced by the CCD chip may be considerable [35]. The CCDs give rise to the following three most evident noise types [1, 4]: the readout noise, the dark charge noise and the photon shot noise. The readout noise occurs during the "counting" of photons in the detector, i.e., during the process when "amount" of incoming light is translated into an electric signal before it is digitized. This is an intrinsic parameter of every CCD chip and there is nothing we can do about it. The dark charge noise, also often termed the dark current noise, is a noise produced by thermally generated charge. It is especially evident in tests with

shutter remaining closed during image integration in the CCD, the CCD will read out some values despite there was no incomming light. That is why this noise has adjective *dark*. It is manifested predominantly in regions in images where no fluorescence appears and the background values are still not zero (zero pixel intensities represent no signal). An intensive cooling of the CCD, e.g., to -70°C, suppresses this source to some extent. Last but not least, the photon shot noise is a detectable statistical fluctuation in the incoming photon flux. This one is especially a result of small intensity of light approaching the CCD chip because otherwise the fluctuation would be negligible.

Hence, the data considered in this thesis will be time-lapse sequences of monochromatic grayscale, i.e., single channel, 2D or 3D images. We will call the single 2D or 3D image a frame, sequences will be consisting of consecutive frames. The displayed structures in each frame will be a bit unsharp, due to the point spread function, and low contrast, due to the limited amount of light during the acquisition process. There will be often present certain amount of, so called, non-specific staining or cross-talks. This is a weak signal emanating from regions in a cell where either no fluorescent dye should be present or another dye with partly overlapping excitation/emission spectra is present. Technically speaking, images suffer from low SNR (signal-to-noise ratio) and low contrast, they are rather faint typically with absence of strong edges. Typical dimension will not exceed 1000px neither in the $x$ axis nor in the $y$ axis. In the case of 3D images, the dimension in the $z$ axis will not exceed 100px. The number of time instants, frames, is mostly less than 10. Example of two HL60 nuclei are presented in Fig. 2.8. A nice summary on challenges to automated tracking of fluorescence microscopy images can be also found in the recent work of Dzyubachyk *et al.* [51].

### 2.2.2 Properties of observed movements

Cells are mostly floating, waggling or rotating in their medium on the microscope slide, see Fig. 2.9. Our experience shows that it happens predominantly in the lateral direction. Movement, in the sense of floating up or down, in the axial direction seems to be very unlikely during the observation, they only decline sometimes. We believe that this is because of the medium, which is liquid and in which cells are, sort of, nailed down on the slide by the gravity. Another reason is the small thickness of the specimen, i.e., the distance between the slide and the cover glass, which doesn't permit cells to float or rotate significantly in the axial direction as well. Hence, they typically don't occlude. On the other hand, they often touch one another. Cells also do not change their size dramatically. If such phenomenon is expected to occur, the sampling period between acquisitions is usually adjusted so that there is no great change between two consecutive images in the series. All of this, to a great extent, is applicable to subcellular components as well, see Fig. 2.10.

For instance, Matula *et al.* [16] designed an alignment technique to suppress global movement of live cells in a time-lapse confocal (3D) images. The authors reported no significant scale changes even between several time-consecutive images in their test data. They also reported no considerable rotation around $x$ or $y$ axis. Their technique is based on pairing of detected intracellular objects. In this case, images of telomeres and images of HP1 domains were used. They succeeded with an extension of the translation, rotation and scale invariant 2D registration method of Chang *et al.* [52] into 3D space even when

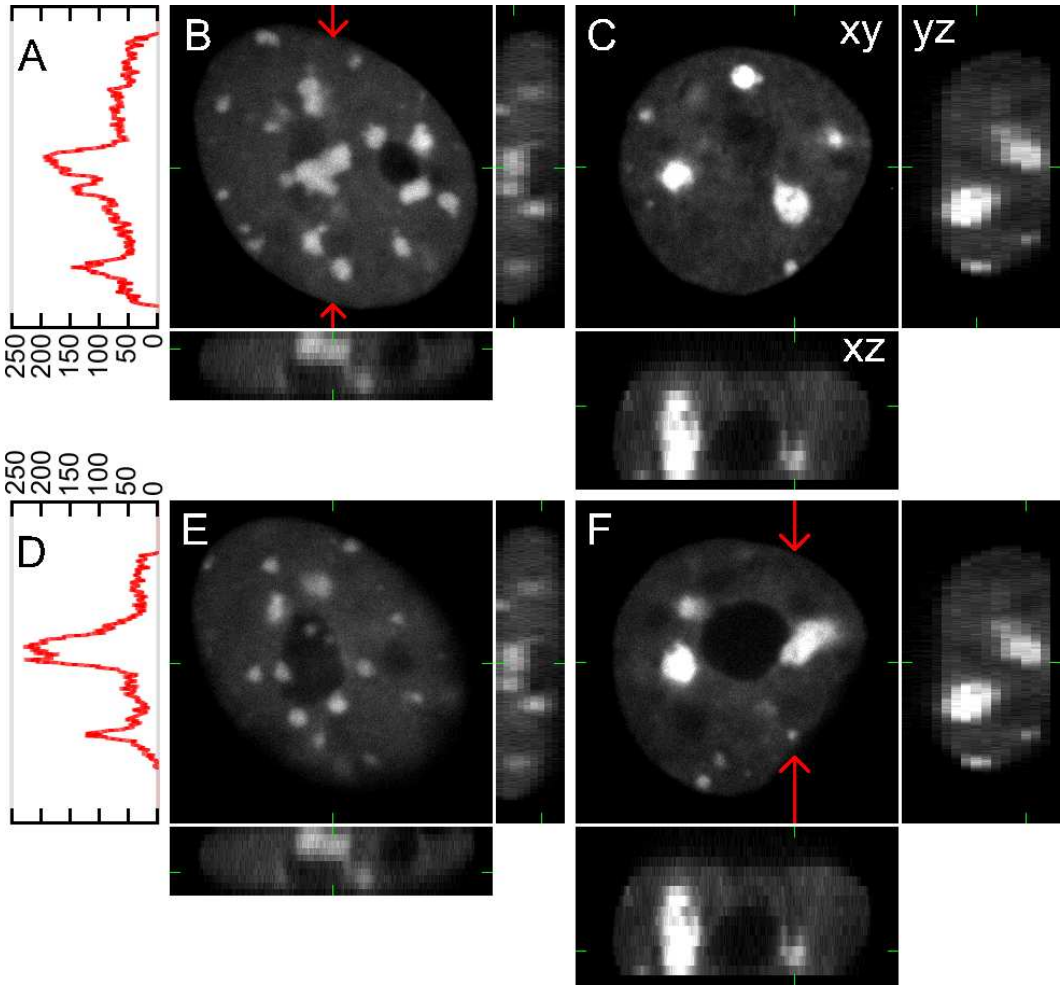


Figure 2.8: Example of two 3D frames of HL60 nuclei with stained HP1 domains. Only a digest in the form of two xy, yz and xz orthogonal cross-sections for each cell can be printed here. The first cell is displayed in B and E: it spans a volume of $300 \times 300 \times 10$ pixels which corresponds to a volume of $23.4 \times 23.4 \times 5.0\mu$m in this case (results in z-step of size $0.5\mu$m), two lateral sections at z=2 in B and z=6 in E are shown, the yz section is shown for x=150 and the xz section is shown for y=150. The second cell is displayed in C and F: it spans a volume of $300 \times 300 \times 16$ pixels which corresponds to a volume of $19.5 \times 19.5 \times 9.6\mu$m in this case (results in z-step of size $0.6\mu$m), two lateral sections at z=7 in C and z=12 in F are shown, the yz section is shown for x=203 and the xz section is shown for y=150. Notice that to keep the aspect ratio, the yz and xz sections of the second cell are rendered with "greater pixels". Also notice that the resolution in the axial direction ($z$ axis) is worse, the texture is more jagged. The example illustrates the sort of spatial arrangements of stained structures inside these cells as well as the intensity proportions we have to deal with when analysing this type of images. The latter is demonstrated with two intensity profiles shown in A and D. The A (or D) profile sweeps along imaginary line in the xy section of B (or F) as designated with red arrows in B (or F). The more a red curve in A or D is to the left, the greater pixel value it represents. All images were enhanced for printing purposes. Courtesy of Vladan Ondřej.

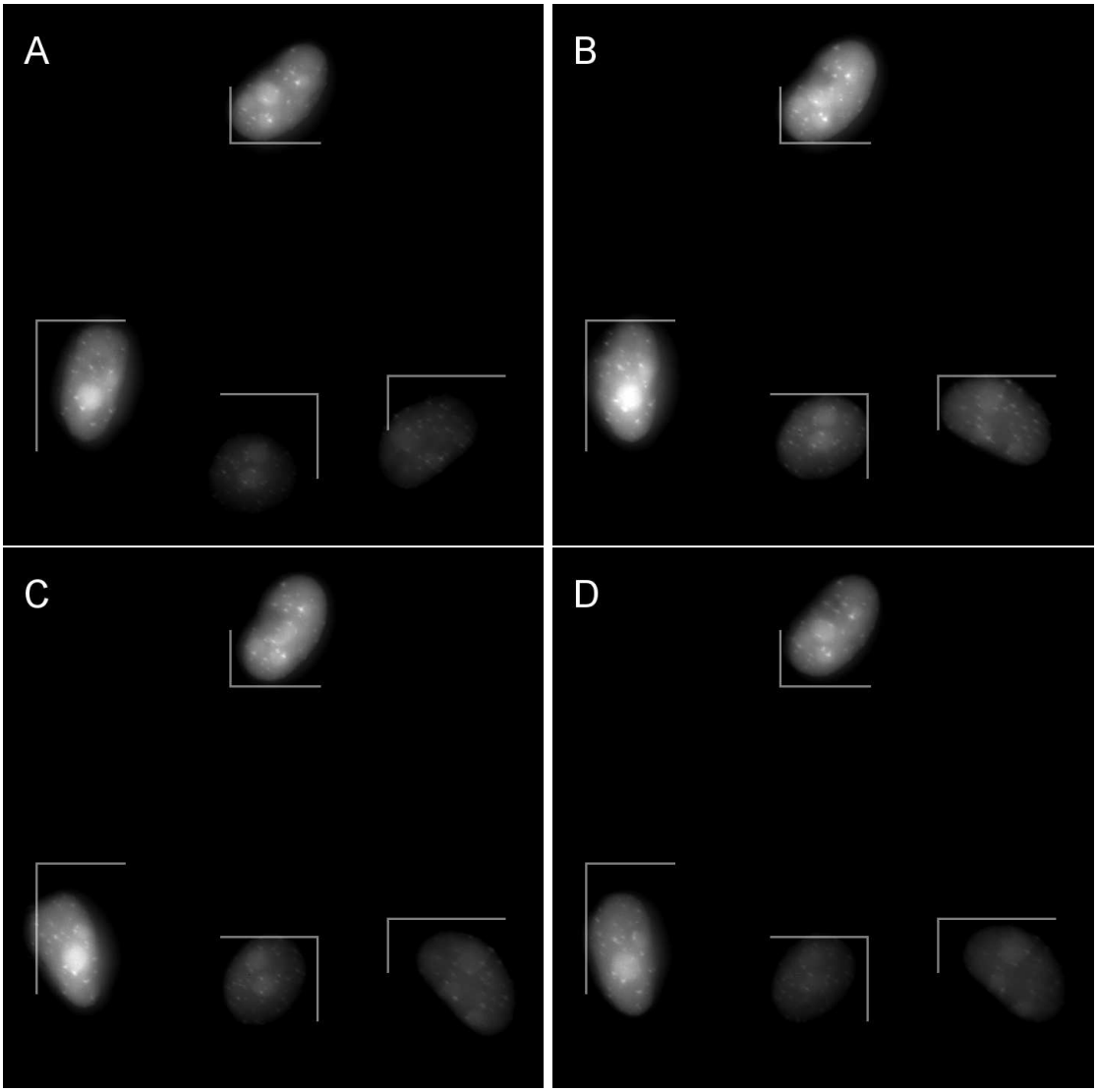


Figure 2.9: Illustration of a typical motility of isolated whole cells. Four consecutive 2D frames (time instants) are shown in A,B,C and D. The artificial line-marks are occuping the same locations (they are registered) in the image sequence. They were inserted only to ease observation of cell movements. Using these, one may notice that the bottom-left cell is waggling while, for example, the bottom-right cell is translating and rotating at the same time. All images were enhanced for printing purposes.

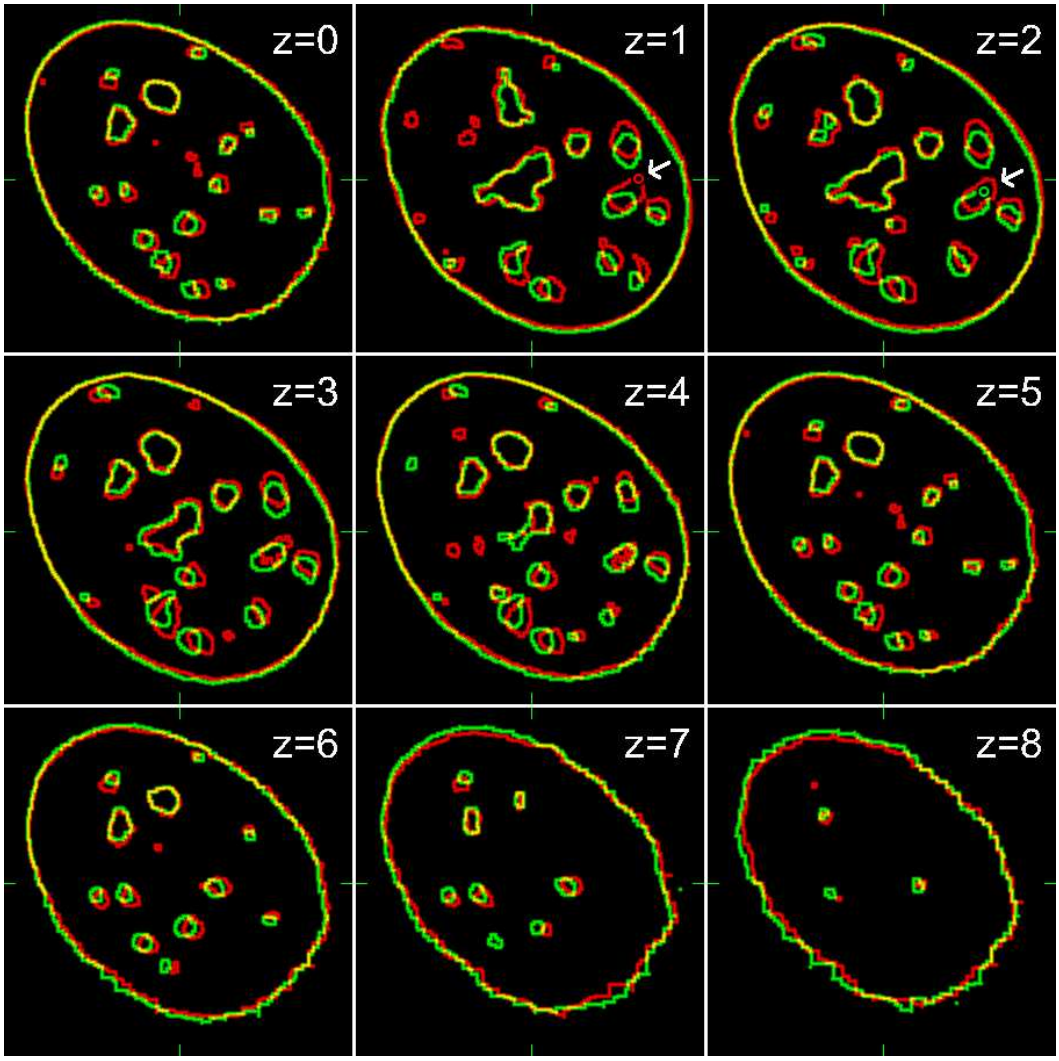


Figure 2.10: Illustration of movements of intracellular structures (HP1 domains in a HL60 nucleus). Nine slices of the two consecutive frames mapped to red (darker in B&W print) and to green (brighter in B&W print) are shown here. Quite intentionally, this 5th frame was also shown in Fig. 2.8B with raw pixel intensities. In order to give an example of how cell changes and moves between two consecutive time instants, we have opted to show only silhouettes of cell and HP1 domains. In the display, we observe mosty translational and/or rotational motion of domains of various magnitudes: from stationary ones inside the cell, though the cell itself is a bit clockwise rotating, to domains travelling as far as 12.5 pixels between these two frames, e.g., pixels in the red (earlier) circle are translated to the green (latter) circle by the vector $(-5, 11, 1)$. However, this magnitude of movement also includes a local magnitude of global movement of cell as such. This movement must be "subtracted" from the measured one in order to assess a true movement of the domain within the cell, which is usually the information desired by biologists. Notice that we also observe a bottom of the cell in slice z=8 as the cell contour is smaller. In fact, the slice at z=9 is empty in the original data because it is imaging optical section from the specimen just bellow this cell. Courtesy of Vladan Ondřej.

they kept the support for rotations only around the $z$ axis, that is 2D lateral rotations. From their results it suffices to describe a global movement of the particular cells by means of a 3D translation vector and rotation angle around the $z$ axis. This was also evidenced by Bornfleth *et al.* [15]. Dufour *et al.* [53] proposed a segmentation and tracking method that uses volume conservation constraint to improve outlining of cell boundaries. They tested their method on two types of cells, namely the human parasite *Entamoeba histolytica* and proliferating MDCK cells. Sample images in their publication evidence a flat lateral distribution of cells in the acquired 3D volumes as well.

We are often working with images where only a single cell is shown. Quite often, these images are a result of an automatic or semi-automatic extraction that simply makes a copy of the relevant rectangular portion from the original image. In this way, it aims to split images with several cells into more images each with a single cell and its small neigborhood. Clearly, once the region (in 2D) or volume (in 3D) of interest is determined around a given cell, it is fixed and kept constant over the whole image sequence so that the movement of cell is apparent even in the new sequence of extracted (and smaller) images. A region/volume of interest is given by its offset within the original image and by its size. For instance, this was the case of images of cell in Fig. 2.8.

# Chapter 3

# Related work and theory

We aim to *be able* to track *some objects* in time-lapse image sequences of live cell studies. The goal is not to design an ultimate tracker, i.e., a program that would follow and draw a line along some user-selected *object* in all images in the time-lapse sequence. We wish not to head towards an approach with explicit segmentation involved. This is predominantly because we don't want to restrict ourselves to tracking of any *particular* objects. We aim only to provide just enough information, in whatever form, that would enable anyone to link any objects in the sequence once they are segmented, either beforehand or afterwards. The optical flow technique can achieve this.

The purpose of this chapter is to review approaches to tracking followed by introduction to optical flow computation. Both topics should be dealt with mainly in the context of biomedical images. In the second half, we take a close look on the representation of motion both in spatio-temporal images as well as in the Fourier domain. We will also discuss some of its aspects with respect to the human visual system as well as with respect to motion estimation based on Gabor filtering.

## 3.1 Tracking in biomedical images

### 3.1.1 Overview of approaches to tracking

#### The two main concepts

According to the literature [20, 54, 11, 13, 55, 17, 14, 53, 18, 56] and, we believe, also as a consequence of the chicken-egg dilema explained in the introduction, we have only two fundamental options how to approach the tracking. Either we preprocess the input images and obtain some characterizing feature vectors in the first step among which we establish correspondences in the second step; or we estimate motion of some anonymous masses (patterns) either moving or still and use the estimate in following stages where objects are identified and tracking is completed. The feature vector carries information such as coordinate of centre of mass, mean intensity value, size, roundness, local dominant orientation or even some intensity pattern, etc. [13, 46, 57]. In the rest of this section we shall see that the two typical representants of the two concepts are the techniques of image registration and optical flow, although counter-examples exist.

The image registration, in the context of this thesis, can be regarded as the process

of overlaying two images such that they become aligned. The images should show the same scene taken at different times. A *parameterized* transformation model transforms one image to the other. The registration process seeks proper parameters of this model for given image pair such that images align well in terms of some similarity criterion. Clearly, the model should be general enough to allow for correct alignment. This is useful, for example, for suppresing global movement of a cell [16, 10]. The image registration can be also conducted at the level of objects, i.e., aligning views (cropped subimages) or matching feature vectors of the same object in two or more images. This is useful for tracking several independently moving objects within images in the sequence [58]. A necessary precondition in this case is that the linked objects must be segmented beforehand.

The optical flow techniques produce flow fields. A flow field describes an optical (visual or percieved) flow of displayed content (mass or simply small brightness patterns) from the earlier frame to the later one by means of a collection of, so called, flow vectors. There is one flow vector associated with every pixel. There exists one flow field for every pair of consecutive frames in the time-lapse sequence, eventually. In order to track certain structure in the sequence, it is simplest to adjust the structure's position according to the flow fields. However, to understand the type of detected motion, e.g., how much a cell is rotating, further analysis of the flow fields must be conducted as well as segmentation of the cell.

#### Tracking as motion estimation

Despite establishing inter-frame correspondence of still objects is also a goal of any tracking method, most researchers expect that the subjects to tracking are in motion. This is probably why methods typical for the general field of motion estimation, a representant of the computer vision field, are often encountered in the time-lapse microscopy image processing and analysis. The optical flow is nice example of such an application. Some concepts are, however, called differently.

For instance, Cédras and Shah [20] in their survey on motion-based recognition used the term "motion correspondence" for the sort of methods that, we cite: "deals with extracting interesting points, characteristic features in an image, that can be tracked in time." As a counter-approach they used the optical flow. Konrad [54] considered, we cite: "two essential models in motion estimation: a *motion model*, i.e., how to represent motion in an image sequence, and a model relating motion parameters to image intensities, called an *observation model*." The former model tries to describe what happens with pixel intensities in the image sequence if they display object in movement. We aim to find its driving parameters such that the model fits to the given image sequence. It should encompass at least an image formation model, motion model and surface model [54]. But in the end, it is only a transformation, whose parameters we seek, within images, i.e., an image registration technique. The latter model, according to Konrad [54], deals directly with image intensities respecting the constant brightness assumption (eq. (3.8), explained in Section 3.1.3), i.e., variants of optical flow. Dubuisson06 [56] identifies the two sources of information for tracking: a model of appearance of the moving object and a model of the dynamic behaviour of the object. The former source, in fact, enables us to seek difference in positions over time of some fits of the *a priori* given model. The model can be arbitrary, e.g., colour distribution, intensity patterns or edges. Dubuisson notes that

deformable models, snakes and appearance models are the most popular techniques for tracking. The second main source of information, according to the author [56], is the model of dynamic behaviour, which imposes no constraints on the shape of the object. It rather deals directly only with sets of observations, which we may understand as pixel intensities. Methods based on this source of information then estimate evolution of such observations from which the motion can be estimated. Finally, the recent (computer vision) survey publication on general object tracking by Yilmaz *et al.* [59] identifies four features to track: the colour, edges, optical flow and texture (in a form of precomputed descriptors). The colour information can't be exploited in microscopy unless we stain the same tracked structures with more dyes, which is fairly unusual. The edges refer to the deformable models and the texture to the image registration. The publication often notes that the optical flow, besides tracking based solely on it, is often used as an auxiliary source of information, especially for methods based on active countours [60, 61, 62]. Occasionally, it even initiates a particular method [63].

To sum it up, the techniques of image registration and optical flow seem to be the two most often used for correspondence finding. Thus, we shall examine each closer.

### 3.1.2 Image registration

#### Classifications of image registration techniques

We have already mentioned one aspect according to which the registration techniques can be classified. This was the level or (spatial) scope at which the registration shall operate. In particular, we may want to align a whole given image to a reference one. We may also want to align just subimages containing objects extracted from both given and reference images prior to the registration. While in the former case we align one image to the other one, in the latter case we align $n$ to $m$ objects *simultaneously*. Note that $n$ does not have to be the same as $m$ due to extra or missing detected objects in, possibly, either images.

Another classification is possible according to type of a *transformation model*. Say that the transformation model has vector $\vec{p_N}$ of $N$ model parameters. When all $N$ parameters are supplied, we obtain a particular instance of the model, the *image transformation*. The transformation can be, in the general case, defined with function $\vec{y} = T(\vec{p}_N, \vec{x})$ that "transforms" input image coordinate $\vec{x}$ (a column vector) into its corresponding coordinate $\vec{y}$ in the transformed image. To transform a source image means then to forward its pixel value at $\vec{x}$ to a new coordinate $T(\vec{p}_N, \vec{x})$ in the transformed image. If good model parameters $\vec{p}_N$ are used, the transformed image should be similar to the reference image, in which case we say that the transformed image *is registered*.

#### Transformation models

The most frequently used transformation models in the time-lapse microscopy are the rigid, affine and nonrigid transformations [10]. The rigid transformation preserves shape and content of the transformed pattern, it can only translate and/or rotate it but it can't deform it. The affine transformation, in addition to what the rigid one is capable of, can scale and shear. Hence, it only preserves topology. The nonrigid transformation can be

arbitrary. For example, the model of affine transformation for 2D image is given as

$$\vec{y} = T(\vec{p}_6, \vec{x}) = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \vec{x} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{3.1}$$

Here, $N = 6$ and $\vec{p}_6 = (a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}, b_1, b_2)$. The parameter $N$ represents the degree of freedom of the transformation model. The higher the number is, the more complex and also the more general the model is. The affine transformation degree of freedom is up to $N \leq n^2 + n$ for a sequence of $n$D images.

The selection of a model is based on *apriori* knowledge of processed data [64]. When tracking several cells in an image sequence, the transformation model should be quite general, e.g., it should allow for local deformations, cell divisions, cell entering and/or leaving the frames [13, 51]. When aligning single cell, the transformation model should also account for intracellular deformations such as in Fig. 2.10 where different cell structures change their shapes and/or positions differently. Elastic transformation models are often used when nonrigid transforms are required [10].

It is important to realize that in order to fully determine parameters, the vector $\vec{p}_N$, of the assumed transformation model, *we* should be able to find at least $N$ applications of it in the processed pair of images *at the same time*. In other words, we should be able to find a *mapping* in the form of $j$ pairs $(\vec{x}_i, \vec{y}_i)$, $i = 1, \ldots, j$ with $j \geq N$. The mapping yields a collection of constraints $\vec{y}_i = T(\vec{p}_N, \vec{x}_i)$, which we solve for $\vec{p}_N$. However, such system has rarely a single correct solution (ideal fit of the model). As a result, minimization of a residual error $E_{\vec{p}_N}$ is often sought. The residual error is supposed to indicate quality of $\vec{p}_N$ under the examined constraints. For example, when group of points move over time, the residual error may take the form:

$$E_{\vec{p}_N} = \sum_{i=1}^{j} (\vec{y}_i - T(\vec{p}_N, \vec{x}_i))^2. \tag{3.2}$$

The goal of image registration technique is to find mapping as well as $\vec{p}_N$ that together minimize $E_{\vec{p}_N}$. In words, it seeks optimal mapping whose optimality is (quantitatively) supported with the most consistent instance of the transformation model. Note that the mapping represents the correspondence. Unfortunately, it often leads to an iterative cost-minimizing process [65, 66], which may be time demanding. When some very flexible model is employed ($N$ is high), we may not be able to have $N$ constraints available to complete the registration, e.g., when, returning to the example, only $j < N$ points is available.

### Principle of voxel-based and feature-based techniques

The image registration techniques can be also classified as voxel-based or feature-based [64, 10]. Basically, this classification distinguishes between sources of information supplied into the registration routine.

The *voxel-based techniques* work directly with image data. They typically perform no feature extraction. The mapping pairs $(\vec{x}_i, \vec{y}_i)$ are given with the image transformation $\vec{y}_i = T(\vec{p}_N, \vec{x}_i)$. These techniques evaluate blocks of image data with correlation-like measures [64, 10]. During the search for appropriate instance, the vector $\vec{p}_N$ of the transformation

model, each instance is examined against very large constraint system with $j \gg N$. Note that $j$ can be as large as the number of pixel coordinates in the overlap of a transformation of the given image, $I^G$, and the reference image, $I_R$. The residual error is usually of the form:

$$E_{\vec{p}_N} = \sum_{i=1}^{j} IVSM\left(I^G(\vec{x}_i), I^R(T(\vec{p}_N, \vec{x}_i))\right) \tag{3.3}$$

where $IVSM\left(I^G(\vec{x}), I^R(\vec{y})\right)$ is an Intensity Values Similarity Measure operating on pixel value at $\vec{x}$ in the given image and on pixel value at the mapped (destination) coordinate $\vec{y} = T(\vec{p}_N, \vec{x})$ in the reference image. The $IVSM$ returns with a real number. The higher it is, the less similar the intensities are. The $E_{\vec{p}_N}$ is often a sum of absolute/squared differences or other correlation-like similarity measure [10, 64]. If the right-hand-side of eq. (3.3) is modified, correlation or a more elaborate measure of, so called, correlation ratio [67] or mutual information can be obtained [68]. A common property of most voxel-based registration techniques is a great computational demand as, in the worst case, for every reasonable model parameter combination $\vec{p}_N$ the sum $E_{\vec{p}_N}$ must be evaluated, which, in turn, ranges nearly over the whole input image. Considerably faster variants based on the fast Fourier transform exist [69].

The *feature-based techniques*, or also alternatively the *point-based* techniques, extract feature vectors both from the given and reference images. There is typically one such vector associated with some salient loci (corner, landmark or contour, etc.) or otherwise interesting region such as some intracellular structure of interest (gene particle, chromosome territory, protein domain, etc.) or even a whole cell. A feature vector consists of several measured quantities whose selection depends on situation and on type of objects the feature vector is associated to. The purpose of the vector is to uniquely describe and identify the object it represents, at least object coordinate is always included in the vector. This is where the designation *point*-based has come from. Other examples of the measured quantities are mean intensity, local histogram, area/volume or local maximum curvature [65, 10]. In general, the selection of points as well as of the features should be invariant to the assumed transformation model [65, 64], e.g., assign feature vectors only to centres of circular blobs if rotation is expected to occur. It is these feature vectors that are dealt with in the registration routine. Say, $j$ vector pairs are examined. The residual error has the form:

$$E_{\vec{p}_N} = \sum_{i=1}^{j} FVSM(\vec{v}^G_{\vec{x}_i}, \vec{v}^R_{\vec{y}_i}) \tag{3.4}$$

where $FVSM(\vec{v}^G_{\vec{x}}, \vec{v}^R_{\vec{y}})$ is a Feature Vectors Similarity Measure operating on a pair of feature vectors associated to object at point $\vec{x}$ in the given image and to object at point $\vec{y}$ in the reference image provided the pair $(\vec{x}, \vec{y})$ is included in the examined mapping. The transform model function $T$ is usually incorporated into the $FVSM$ by assuming that motion of objects obeys some implicit geometric properties. Note that object coordinate is always included in the feature vector. The $FVSM$ returns with a real number. The higher it is, the less similar its input feature vectors are.

The point-based approaches are generally preferred in biomedical imaging [10] since for every $\vec{p}_N$ there are only a few easily computed feature vectors dealt with in comparison to the large number of examined pixels in the voxel-based approach, i.e., the sum in eq. (3.4)

aggregates over far less values of $i$ than the sum in eq. (3.3).

### 3.1.3 Optical flow

We are already familiar with the fact that any optical flow method produces flow field for a pair of images and that there is one flow vector associated to every pixel. The idea is that the flow vector should estimate relative change of coordinate that the *intensity* at the associated pixel undergoes between the two frames. This is formalized for a pair of 2D images as

$$I(x, y, t) = I(x + u, y + v, t + 1) \tag{3.5}$$

in which $I(x, y, t)$ is an intensity value at coordinate $(x, y)$ at some time instant $t$, the following time instant is represented as $t + 1$. A flow vector $(u, v)$ is associated to the pixel at $\vec{x} = (x, y)$. Note that we have increased the image dimensionality and replaced the sequence of 2D images with only a single *spatio-temporal* 3D image in which the consecutive original 2D frames are stacked along the third axis. The third axis is, therefore, often denoted the temporal axis, the $t$ axis.

The equation is usually reformulated using the Taylor series expansion [19, 70, 71]. The right-hand-side of eq. (3.5) is then changed to:

$$I(x + u, y + v, t + 1) = I(\vec{x}, t) + uI_x(\vec{x}, t) + vI_y(\vec{x}, t) + I_t(\vec{x}, t) + O(\vec{x}, t). \tag{3.6}$$

The error term $O(\vec{x}, t)$ encompasses the rest of the expansion, namely the 2nd and higher derivatives. The $I_x(\vec{x}, t)$, $I_y(\vec{x}, t)$ and $I_t(\vec{x}, t)$ are the partial derivatives of image intensities in the directions of the $x$, $y$ and $t$ axes, respectively. We assume that the error term is negligible, i.e., $O(\vec{x}, t) \approx 0$, and thus we drop it in the following equations. After substitution of eq. (3.6) to the original eq. (3.5) and subtracting the common term $I(\vec{x}, t)$, we obtain:

$$uI_x(\vec{x}, t) + vI_y(\vec{x}, t) + I_t(\vec{x}, t) = 0. \tag{3.7}$$

By letting $\nabla I(\vec{x}, t) = (I_x(\vec{x}, t), I_y(\vec{x}, t), I_t(\vec{x}, t))$ to be the image *gradient*, a row vector of partial derivatives, we obtain equivalent to eq. (3.7):

$$\nabla I(x, y, t) \cdot (u, v, 1) = 0. \tag{3.8}$$

These two equations are often called the *brightness constancy constraint* or the *brightness constancy assumption* [71]. Sometimes, it is even called the gradient constraint equation [72, 70]. Provided all of the above assumptions hold, this constraint relates image gradient at some coordinate with the optical flow velocity $(u, v)$. In particular, the vector $(u, v, 1)$ should be perpendicular to the image gradient $\nabla I(\vec{x}, t)$. It is remarkable how well this constraint performs in general [71] since the original eq. (3.5) is quite often violated in real images. There can be many reasons for it, e.g., different illumination conditions or noise. In time-lapse microscopy, the violation is mainly due to the noise and the effect of photo-bleching, which both differ with every acquisition.

Another way to arrive to these constraints is to track points of constant brightness in the constructed spatio-temporal 3D image [19, 72, 71]. That is to follow a spatial trajectory $\vec{x}(t) = (x(t), y(t))$ in the course of time $t$ such that $I(\vec{x}(t), t) = c$, where $c$ is some constant pixel intensity. Taking a temporal derivative of it,

$$\frac{dI(\vec{x}(t), t)}{dt} = 0, \tag{3.9}$$

using the chain rule we obtain,

$$\frac{dI(\vec{x}(t),t)}{dt} = \frac{\partial I}{\partial u}\frac{du(t)}{dt} + \frac{\partial I}{\partial v}\frac{dv(t)}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = 0, \tag{3.10}$$

which we further develop to eq. (3.7). In order to achieve that, it is helpful to realize that derivative of trajectory in time is a velocity, e.g., $\frac{du(t)}{dt} = u$, and that velocity components $u(t)$ and $v(t)$ are in the directions of axes $x$ and $y$, respectively. The partial derivatives, for example, $\frac{\partial I}{\partial u}$ then becomes $I_x$. We will later realize that this derivation is more in the view of Section 3.3.1.

With the 30 years of existence[1] of the now-classical constraint on brightness constancy, eq (3.8), there also co-exist a collateral classical problem and classical solution to it. The problem is that for any pixel we have only a single constraint (be it eq. (3.5) or eq. (3.8)) with two unknowns. There has to be at least one constraint more added to estimate flow vector $(u, v)$ for the given pixel. In fact, any number of constraints may be added and turn the estimation of optimal flow vectors into a minimization problem. Horn and Schunck [19] proposed to use the smoothness constraint that forces the flow vectors $(u, v)$ to be locally smooth. This can be expressed with Laplacians of the components of the flow vectors [19] or alternatively as a sum of squared 1st derivatives of the flow field [71]. For this moment let us conclude that there exist a few variants of both the brightness constancy and smoothness constraints, see [70, 71, 73] for overview. We will return to this topic later in Section 3.2 when comparing different optical flow computation methods.

A particular limit of the approach is that the estimated velocities $(u, v)$ should be small, i.e., $|(u, v)| < 2$px per frame [70, 73]. It can be best seen in eq. (3.6) in which *truncated* Taylor expansion is pulled into the constraint. In order to keep the error term $O(\vec{x}, t)$ small the terms $u$ and $v$ should be rather small. Computing optical flow for sequences where faster velocities occur is typically achieved in a coarse-to-fine manner [74, 75, 70]. It uses pyramidal representations [76, 77] of both images between which the flow fields is to be estimated. The pyramid comprises of several copies of an input image at iteratively decreased spatial resolutions with the original image in the bottom. Velocities are first estimated in higher levels, they are then propagated towards the bottom by warping one image according to the current estimates. The warped image is expected to become sufficiently close to the second image and the process is repeated.

Note that despite we have presented optical flow for 2D image sequences, many constraints can be readily extended to general dimension $n$D, $n \geq 3$. However, providing a working implementation of such a method may be far from the decleared readiness. Implementations of some established (differential) methods for optical flow computation on 3D image sequences has been published just recently [78, 79] and [P7].

### True motion, optical, component and normal flows

Let us make a comment on the nature of optical flow fields. It is not true that the flow field must describe exactly what is *happening* in the displayed scene. It really should only describe what we *see* in some low level sense, i.e., without emploing our prior knowledge of the world. In other words, the estabished definition, due to Horn and Schunck [19], is that optical flow is "the distribution of *apparent* velocities of movement of *brightness patterns*

[1] Indeed, the influential publication by Horn and Schunck [19] has been published in 1981.

in an image." This is especially the case of real world images where *perspective* projection from inherently 3D real world is applied to form a flat 2D image and where specular effects, light reflections, shadows, occlusion and/or surface change happen [80, 81, 82]. For example, consider a car moving on a straight road towards us and the road is exactly paralel to the normal of image plane. Thus, the road in the image is pictured as it is going from the top to the bottom. The car then appears to move a bit downwards and gets bigger with each image in the sequence. Hence, the optical flow vectors would point a bit downwards and a bit away from a common centre located somewhere in the car what would suggests that in reality the car is perhaps crashing and drowning into the road while it is getting physically bigger because the pixel resolution is constant within the sequence. But the car just goes on the road. This is an illustration of the difference between the optical flow field and the true *motion flow* field. The true motion flow field should describe our understanding of the scene. In this case the motion flow vectors assigned to the car would have to contain additional element to express the depth. This element would also allow to make all vectors originating from the car to be exactly the same because the car is expected to be rigid, i.e., every piece of the car (bodywork, roof, windows, bumbers, lights, etc.) should move with the same velocity and in the same direction.

We can regard the optical flow field equal to the true motion flow field in cell microscopy. The reasons are explained later in Section 5.2 in Chapter 5. We will briefly summarize them here, for convenience: Cells are floating predominantly in their lateral direction in their physiologic liquid and, owing to the gravity, they "sit" on the slide, which we observe either from the bottom or from the top. The axial distance of cells from objective can be, therefore, regarded constant in both wide-field and confocal modes.

Even though there is a reduction of information when moving from the true motion flow field towards the optical flow field in the general case, the latter is sometimes still impossible to retrieve. It is very often due to the effect of aperture, Fig. 3.1. Following the figure, there may be many acceptable flow vectors and no clue to select which one is correct. Most of the optical flow computation methods in this case tend to select the flow vector in the direction of image gradient, the one given by eq. (3.8). Such vector is often termed the *component flow* vector as it is only partially correct, only one component of the correct optical flow vector is found. In fact, some methods start directly with the component flow which they then re-adjust according to some additional constraints to finally obtain a flow field [83, 84, 85]. The component flow is often also called the normal flow.

### 3.1.4 Current trends in time-lapse microscopy

For time-lapse microscopy, the review by Eils and Athale [13] suggests to use either specialized single particle tracking, parameterized image registration or optical flow. This is confirmed also by Gerlich *et al.* [11] and others [55, 17]. However, for the particle tracker Eils and Athale [13] note: "The basic principle of single particle tracking is to find for each object in a given time frame its corresponding object in the next frame. The correspondence is based on object features, nearest neighbor information, or other inter-object relationships. Object features can be dynamic criteria such as displacement and acceleration of an object as well as area/volume or mean gray value of the object." In our opinion, this essentially describes the image registration techniques as outlined above.

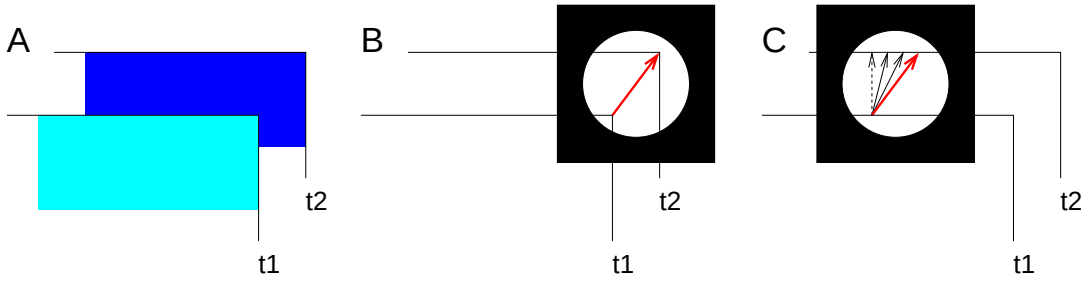


Figure 3.1: The aperture effect. In A, suppose a constant intensity "pattern" translating top-right and shown at two consecutive time instants t1 and t2, t2>t1. It occupies the lighter area and moves into the darker one in the course of time. In B and C, suppose we observe the situation with a spatially fixed apertures. Owing to the corner we can easily tell what is the correct direction of movement in B. But if there is only limited amount of information, as in C, we can't be sure which of the suggested vectors is the correct one. The dotted one is the component flow vector, i.e., vector in the direction of image gradient. The red thick one is the correct vector.

Hence, we regard both particle tracking and parameterized image registration to be only two differently focused image registration techniques. The techniques of image registation and optical flow, thus, dominate.

### Typical applications of image registration to biomedical images

Kozubek *et al.* [46] suggests to stay only with the image registration for tracking. Meijering *et al.* [17, 57] notes that the basic concept underlying the vast majority of published methods for tracking particles or other cell interior structures involves detecting individual particles in every frame and then linking them. In general, such concept leads to some variant of image registration technique as we see [17, 57] that, basically, a feature vector is computed, distance measure is established and correspondence finding algorithm is used. According to the recent literature [11, 13, 17, 10], two different tasks in live cell studies are solved with image registration techniques.

The first task is the suppression of global motion when only a *single* cell is imaged. The images of a cell shall be registered after it such that any changes inside the cell are clearly apparent. In this case the registration of a whole image is considered, usually the rigid transformation model is used [10] and appropriate feature points are extracted. The feature points can be either associated to some well detectable cell structures that are static or fixed within the cell [86] or Matula *et al.* [16] showed that the feature points can also represent intracellular structures of interest provided that majority of them move mainly or solely due to the movement of the cell. In both cases, the feature vectors may contain only coordinates of the points they represent because the assumption on rigid cell movement preserves the spatial relative arrangements between the points. As a result, eq. (3.2) can be used directly. Note that we must extract at least $N$ feature points to constrain the model properly. Alternatively, Rieger *et al.* [87] showed an approach based on computation of inertia tensor directly from pixel intensities, thus avoiding the need for segmentation prior the registration. Their approach offers a successful example of application of voxel-based image registration technique.

The second task is simply to track given, usually already segmented, objects. These

objects may take a form of many cells moving individually throughout an image sequence or several intracellular structures of interest moving within a single cell. Unlike in the first task, the tracking should only provide the mapping (the inter-frame correspondence) between appearances of the object. Typically, a point-based representation with feature vector is utilized [16, 58, 15, 17, 57]. In the case of tracking whole cells, usually only a centre of mass is computed for each and the correspondences between frames are established in the nearest neighbor fashion [88, 58]. This should work perfectly until the inter-frame translations of cells are greater than are cell radii. A very popular approach is to track cells with deformable models [57, 10, 53], which is somewhat a hybrid scheme according to the taxonomy we have presented in this work. The approach conducts segmentation simultaneously with tracking by combining both feature vectors and direct intensity investigation. Sometimes, ideas typical for computation of optical flow are incorporated [89]. In the case of tracking cell particles, for which, interestingly, it does hold that their inter-frame translations are several times greater than are their radii, additional constraints are often introduced. These are, for example, more containing feature vectors (e.g., shape characteristics or intensity patterns) to certify match, assumption on motion smoothness or probabilistic approaches, see [18, 17, 57, 10] and references therein.

**Applications of optical flow to biomedical images**

Optical flow is a good candidate for global motion suppression in live cell studies [90, 73, 91, 10]. With the optical flow, we can easily correct for global motion even in sequences showing several isolated (not touching) cells provided there is always a reasonable (spatial) distance between the cells. Modified optical flow for global motion suppression was also used by Kim *et al.* [91]. A general registration, an alignment, of two frames based on optical flow was presented earlier by Bouguet [90]. The review by Miura [18] on tracking movement in cell biology notices that optical flow has been mostly overlooked and gives an example analysis of protein movement using it. He notes that optical flow is a viable option when objects change their shape or when there are many object with overlapping tracks. It suffices to compute flow fields directly without any prior assumptions and consequently track either user or automatically selected/segmented regions [47, 92]. The granularity of this methodology depends on what is demanded in the study and what is possible from results of the optical flow computation. Recently, we showed that modern optical flow algorithms provide good accuracy on fluorescence microscopy images [P7] so that it can be successfully applied for global motion suppression as well as for tracking cells and also intracellular structures [47]. The tracking was achieved by marking objects of interest and following those marks. Basically, it amounts to making a list of coordinates which is updated by the flow fields for every next frame. The authors did not test their method on tracking of particles. Quelhas *et al.* [86] used the optical flow for fluorescence images of a plant root. They detected cell divison in already registered frames by detecting local peaks in magnitudes of the flow field. Note that in both publications [P7] and [86] the modern optical flow computation methods [93, 94] were utilized.

The methods based on the principles of active contours, level sets and deformable models are notably popular for the task of tracking cells as a whole [14, 53, 56, 10, 51], in contrast to tracking particles or other cell interior structures. Note that this is a concept in which the segmentation and tracking is inseperably interconnected. Zimmer *et al.* [14] also

admit that such methods typically require tuning of many parameters, especially weights or different energy terms and initial contour. They suggest to make use of additional sources of information from the image, such as the motion itself obtained with optical flow [89], to improve performance of the deformable cell tracking models.

### Conclusion on motion estimation in biomedical images

It is difficult, if not impossible, to state which approach is in general better. The complete tracking solution will always need to extract data from a given frame (includes segmentation to allow for image analysis) and link it between the frames (to allow for time-resolved evaluation of some features, be it a volume or acceleration). The extraction and linking is sometimes referred to as to the spatial and temporal aspects of tracking [57], respectively. Using this parallel, the two aspects will always be present in tracking because it deals with spatio-temporal data.

To some extent, it is a question of one's preference which approach is favoured as all should lead to the same result, ideally. We have decided to split the task and focus predominantly on motion estimation as such, postponing the segmentation step to some other stage in the analysis. We have opted to use the optical flow for the motion estimation.

The optical flow computation is based on a few simple assumptions, namely the tracked pixel should look similar over time, small local region of pixels should exhibit rather similar motion and temporal sampling should be frequent enough such that inter-frame correspondences can be reliably determined. These are, however, general requirements for motion estimation. On the other hand, the segmentation, especially on the type of images from cell microscopy, need to utilize some form of prior knowledge about the extracted objects. For example, to segment protein molecules it is usually to seek for round intensity patches, small in diameter with pixel intensities (at least slightly) above their surround. But segmentation of cell nuclei is incomparably more complex process requiring, for example, the use of level set methods [95]. We think it is better to conduct segmentation with the aid of motion estimation (optical flow), rather than conducting motion estimation with the aid of segmentation (object-level registration techniques). We think it is better not to restrict to estimation of motion of only some objects.

A flow field can be regarded as general and flexible image transform as it directs motion of every single pixel. Actually, the resolution is slightly worse as we don't have enough constraints for the determination of flow field and so, typically, pixels in close vicinity cooperate. Still, it is possible to use the same optical flow computation routine to estimate motion of several cells shown at the same time or just to focus on a single cell and estimate motion inside it (provided cell is captured at sufficient resolution).

In addition, we are also inspired by the human visual system, which is clearly a good system to model in terms of its performance. In particular, it seems that we, the human beings, can make statements on motion of any object even if we have never seen it before, i.e., we had no prior knowledge or experience. This motivation will be strongly apparent especially in the sections to come.

## 3.2 Optical flow computation techniques

In this section we would like to identify an optical flow computation method that would perform the best on *our* type of data that we have described in Section 2.2.1. Unfortunately, none of the methods is universal and the choice strongly depends on the type of image data. Recently, a clue was given by Miura [18] for measuring movement of proteins, vesicles and cells. He attended to the differential approach based on eq. (3.8) with increased temporal support in form of either the temporal local smoothness constraint or spatial local smoothness together with eigenvalue analysis of spatio-temporal structural tensor. In addition to his results, we have decided to also rely on earlier studies, namely on the influential study by Barron *et al.* [72] followed by Galvin *et al.* [81]. Similar attempts have been made by McCane *et al.* [96] and by Baker *et al.* [97] but they have limited scope of tested methods (they have focused predominantly on the differential approaches). Since there are currently many various methods to choose from, we have opted to focus only on major approaches rather than on particular method. Basically, fundamental difference between approaches is in the selection of constraints they employ.

Following the taxonomy of Barron [72], these are the following major approaches:

**Differential:** It computes velocity from spatio-temporal intensity derivatives (1st or 2nd order) often accompanied with some smoothness constraint. Velocities are estimated locally or globally over all pixels simultaneously by means of minimizing sum of outcomes of penalizer functions (often squares only). This approach is often called the gradient approach as well.

**Region matching:** Images are divided into many reasonably small blocks. Velocity is defined as the shift vector that maximizes some similarity measure (cross-corelation, or minimizes sum of squared difference) between two image blocks at different times.

**Energy-based:** The input image is decomposed with a collection of bandpass velocity-tuned filters into several energy images (computed pixel-wise as magnitude of the complex result of the filtering). Velocity is computed in weighted least squares sense. For every pixel, the residuals are differences between the measured energy outputs and the expected filtering responses. The expected responses are functions of velocity. Estimated velocity is the one realizing the least squares minimum.

**Phase-based:** The input image is decomposed with a collection of bandpass velocity-tuned filters into several phase images (computed pixel-wise from the complex result of the filtering). Each phase image provides one suggestion of *component* velocity, which is estimated in the direction of phase gradient. A velocity is estimated by fitting a motion model in the least squares sense to the component velocities aggregated over a small spatial neighborhood.

The reader is kindly referred to the survey publications [72, 81, 96, 18, 73, 97] for references on particular methods. Nevertheless, we have already explained the idea of the differential approach. The region matching resembles the voxel-based registration technique since both approaches measure similarity of regions directly with pixel intensities. In the optical flow, however, only shift of blocks is expected in contrast to voxel-based registration where arbitrary image transform may be employed. The last two approaches

are sometimes treated together under a common name of *filtering-based* optical flow. They will be explained later in Section 3.3.4 in detail.

As of 2002 [98], it was assumed [72, 98] that the generally good performers were the method by Lucas and Kanade [99], which is the representant of differential approach with local spatial smoothness operator, and the method by Fleet and Jepson [83], which represents the phase-based approach. The other surveys [81, 96, 97] haven't tested energy-based nor phase-based approaches. But they have agreed on the method of Lucas and Kanade.

Considerable improvement came with the nowadays highly-regarded methods by Brox *et al.* [94] in 2004, Bruhn *et al.* [100] and Papenberg [93], followed with the TV-L1 method by Zach *et al.* [101] in 2007. These are all representants of the (new era of) differential approach. Recently, even modifications to some of these for 3D image sequences were reported [P7] and [102]. Refer to the recent survey on data and smoothness constraints by Weickert *et al.* [73]. The novelty of these methods was in the use of robust penalty functions, the use of coarse-to-fine scheme for optimization and the incorporation of stronger local constraints on the motion [71]. And it was exactly this "technological" advance that enabled the progress. As Sun *et al.* [103] notices the formulation has changed little since Horn and Schunck [19]. For instance, Brox *et al.* and Zach *et al.* suggested in their methods the use of intensity preservation constraint, eq. (3.5). Sun and colleagues also show that applying the new computational concepts into the the method of Horn and Schunck, the performance of this method has improved as well [103].

However, even these modern differential methods have their limits. They are, like any other differential methods, strongly dependent on sampling density of input images both in the spatial and temporal dimensions. This affects the computation of derivatives, especially the higher order ones. A common approach to "stabilize" estimation of derivatives is to smooth the image sequence prior the numerical differentiation. Fleet and Weiss [71] noted that we may actually conduct the smoothing and differentiation in one step with some smoothing derivative filter. This idea has been adopted by Weber and Malik [104] and later by Bruno and Pellerin [98]. The authors in fact, used several bandpass derivative filters. Since their "prefiltering" resembled filtering with Gabor filter, we will, according to Heeger [105], show later in Section 3.3.4 that the "prefiltering" approach is very close to the energy-based approach adopted by Heeger [106].

It is only our impression that all four approaches are based on more or less similar or even the same assumptions in terms of intensity preservation, smoothness of the recovered motion and well sampled image data. Hence, we would expect to obtain rather equal performance. Since the phase-based method had been competetive with the differential until 2004, what have happened that now it seems to lack behind? Notably that we have discussed that the modern differential methods have not changed their paradigm, but "only" the underlying computation machinery. In the similar fashion, we have identified a weak point in both energy- and phase-based methods: the filtering. Indeed, most researchers and even today [107] aim towards fast implementation of filtering, which was traditionally the greatest performance bottleneck of the approach in terms of time demand, but they do not focus on optimizing the filters for optical flow computation. We believe that it was predominantly because of the fact that in order to compute the filtering fast, the filters could take only limited number of some simple shapes.

We shall investigate in this thesis that if we manage to overcome this barrier and allow

for fast filtering with filters of, possibly, any shape, if that would allow for the filtering-based methods to improve as well. This, the results of such methods in earlier studies and the fact that human visual system was evidenced to conduct intensive spatio-temporal filtering, were the main motivating factors in our decision to consider the filtering-based optical flow in the rest of the thesis. Moreover, modern differential methods have provided rather good accuracy on live cell images [P7] (about 9° for sequences with combined global and local motions) but it still leaves a room for improvement.

## 3.3 Motion estimation based on filtering

### 3.3.1 Representation of motion

#### Motion in the space-time domain

Motion detection is extraction of spatio-temporal orientation [24]. This encompasses two concepts. Firstly, we recall that a velocity is distance travelled over time. This basic definition inevitably connects two dimensions: the space and time. Indeed, we can't tell if something is in motion or is still from a single image, and this is important to stress, without employing our prior knowledge of the world, e.g., car shown on road is most probably moving. We really need to see at least two images and compare them. We may generalize this idea and collect all frames we have in the image sequence and stack them one above the other while preserving the order in which they appear in the sequence. If the sequence consisted of 2D images/frames, the stack can be considered as a single 3D image, quite often denoted as 2D+t image, also called the space-time cube. The time (or temporal) dimension is simply added to the original dimensionality of the frames, which is commonly termed as the space (or spatial) dimension. A spatio-temporal image is constructed in this way.

The second concept encompassed with the spatio-temporal representation is the notion of deemed orientation of structures created from patterns in motion. This follows from the fact that if a pattern is translating on a regular basis between the frames, it leaves a spatio-temporal trace in the stack, see Fig. 3.2. If we fix a point within the translating pattern, we may obtain a trajectory of pattern's movement. Since the trace is a representation of spatial positions over a time, it allows for velocity extraction. Considering the left drawing, the 1D-over-time example, in Fig. 3.2, we see that velocity corresponds to the tilt of the thick line or, equally, of the tilt of a pattern's edge.

The given representation of motion has two shortcommings. As it is directly targeted at simple translational movement, it doesn't cope with complex movements and with deformations of patterns over time. However, both is a matter of resolution that we use. What is a complex motion anyway? Is it a rotation, for example? Can't it be characterized at more finer level, i.e., with higher resolution, with piece-wise translations? For instance, the rotational movement of some object can be represented as many translational movements of individual pieces of the original object when the pieces are sufficiently small. This example will be recalled again in Chapter 5 on the generator of ground-truth datasets for optical flow (Fig. 5.1C on page 99). The situation is similar with deformations of an object. On the other hand, isn't it the difference in velocities assigned to different parts of the object that we use to realize that object is actually being deformed? We will not give

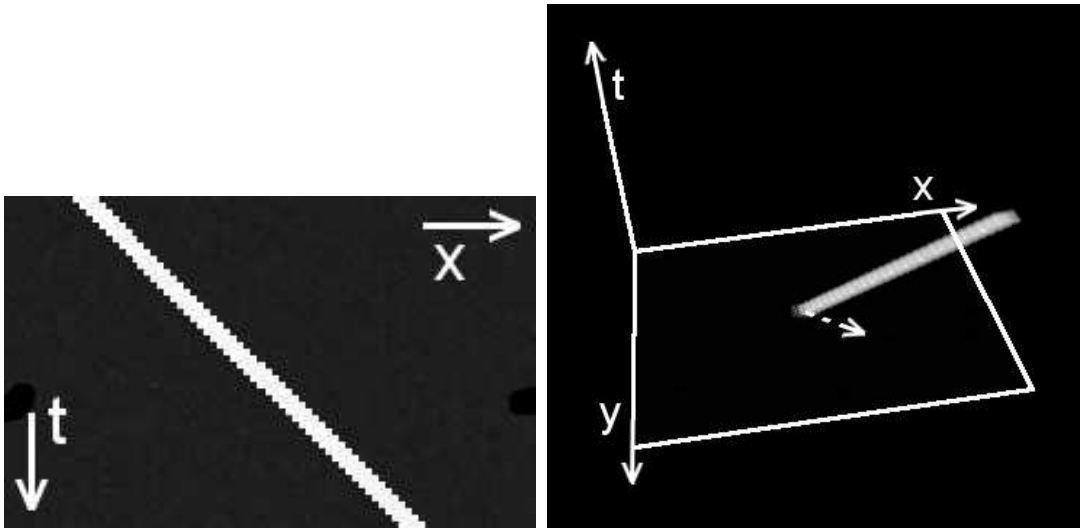


Figure 3.2: Two examples of traces left after translating narrow patches. In the left, the patch translates rightward with velocity 1px per frame in a 1D+t spatio-temporal image. In the right, the patch translates along the dashed vector (1,1) between frames, i.e., with velocity of $\sqrt{2}$ pixels per frame, in the x-y plane in the 2D+t spatio-temporal image.

an ultimate answer on this matter. We rather wanted to bring it to the reader's attention. In the end, we are willing to compute an optical flow for a given image sequence. As the optical flow is represented with a flow field, a collection of *straight* vectors assigned to pixels, everything related is commited to looking for simple straight, i.e., translational, movements of pixels. If rotational movements are to be discovered, one has to further process computed flow fields or take some completely different approach.

On the other hand, the given representation of motion is advantageous whenever the captured scene is rich in texture (to avoid the aperture effect). Consider the example with a 2D+t space-time cube in which the initial image has ideally random texture. We (logically) factor the image into a convex foreground region and a background. The background is static, not moving and not changing over the time. The foreground translates but its content is not changing. As the foreground moves into a new location in the next image, it reveals a portion of previously unseen background that we ideally randomly fill again. As a result, the foreground is indistinguishable from the background in every frame of the sequence. But its motion is apparent in the presented representation because both foreground and background are not chaning over the time. Both give rise to intensity isolines which have different slope, e.g., the background related isolines are parallel to the time axis in the space-time representation. The representation can show a movement of otherwise "invisible" pattern in this way [54]. This is something that the "segment-and-track" approach can never handle. A similar example from real life exists. Consider a fish that can perfectly mimic its environment such that we can't see the fish when it is still. But we see it, actually its outline and with difficulties, whenever it is in movement owing to the revealed portions of the environment, which produce spurious artifacts, that we manage to detect, into otherwise perfect fish cover. Another advantage, provided the texture is rich enough, is that the moving pattern can be of whatever size and shape.

**Motion in the Fourier domain**

We can also view the image motion in the frequency domain. Again, since in the motion analysis the two dimensions, space and time, are always treated together, we are going to find the Fourier transform of the spatio-temporal representation above. We will stay with representation of a translating 2D image. Let us use the notation $I_0(x, y)$ for values of the image at position $(x, y)$. The spatio-temporal image $I$ will have one coordinate more, namely the $t$. Suppose the original image $I_0$ is translating by $(u, v)$ between frames. Hence, its trace in the spatio-temporal image will be in the direction $(u, v, 1)$ and its content will be defined as

$$I(x, y, t) = I_0(x - ut, y - vt). \tag{3.11}$$

Note that $\forall x, y : I(x, y, 0) = I_0(x, y)$.

The Fourier transform of $I(x, y, t)$, which we are going to denote $\mathcal{F}_I(\omega_x, \omega_y, \omega_t)$, is given as

$$\mathcal{F}_I(\omega_x, \omega_y, \omega_t) = \iiint_{-\infty}^{\infty} I(x, y, t)\ e^{-i2\pi(\omega_x x + \omega_y y + \omega_t t)} dxdydt, \tag{3.12}$$

into which we apply eq. (3.11) and rearrange to obtain

$$\mathcal{F}_I(\omega_x, \omega_y, \omega_t) = \int_{-\infty}^{\infty} \left[ \iint_{-\infty}^{\infty} I_0(x - ut, y - vt)\ e^{-i2\pi(\omega_x x + \omega_y y)} dxdy \right] e^{-i2\pi\omega_t t} dt. \tag{3.13}$$

The $i$ is the complex unit, $i^2 = -1$. The equation can be simplified if we introduce notation $\mathcal{F}_{I_0}(\omega_x, \omega_y)$ to be the Fourier transform of the original translating 2D image $I_0(x, y)$. Using the Fourier shift property,

$$\iint_{-\infty}^{\infty} I_0(x - ut, y - vt)\ e^{-i2\pi(\omega_x x + \omega_y y)} dxdy = \mathcal{F}_{I_0}(\omega_x, \omega_y)\ e^{-i2\pi(\omega_x u + \omega_y v)t}, \tag{3.14}$$

we replace the inner square brackets with it and shift the transform out of the integral,

$$\mathcal{F}_I(\omega_x, \omega_y, \omega_t) = \mathcal{F}_{I_0}(\omega_x, \omega_y) \int_{-\infty}^{\infty} e^{-i2\pi(\omega_x u + \omega_y v)t}\ e^{-i2\pi\omega_t t} dt. \tag{3.15}$$

Finally, we make use of the fact that the Fourier transform at $\omega$ of the complex exponential $e^{i2\pi Ut}$, function of $t$ with a parameter $U$, is the Dirac delta function $\delta(\omega - U)$. And since the integral above is the Fourier transform of such complex exponential, we arrive to the important relation between the 3D Fourier transform of the spatio-temporal image and 2D Fourier transform of the original translating 2D image:

$$\mathcal{F}_I(\omega_x, \omega_y, \omega_t) = \mathcal{F}_{I_0}(\omega_x, \omega_y)\ \delta(\omega_x u + \omega_y v + \omega_t). \tag{3.16}$$

The equation basically formalizes the fact that in the Fourier domain a translating pattern gives rise to the, so called, motion plane which is perpendicular to the spatio-temporal direction of motion [22, 23, 105], see also an example in Fig. 3.3. Why is that? The key is in the Dirac delta function which basically states that a value of the left-hand-side transform is possibly non-zero only at coordinates for which it holds $\omega_x u + \omega_y v + \omega_t = 0$, i.e.,

$$(\omega_x, \omega_y, \omega_t) \cdot (u, v, 1) = 0. \tag{3.17}$$

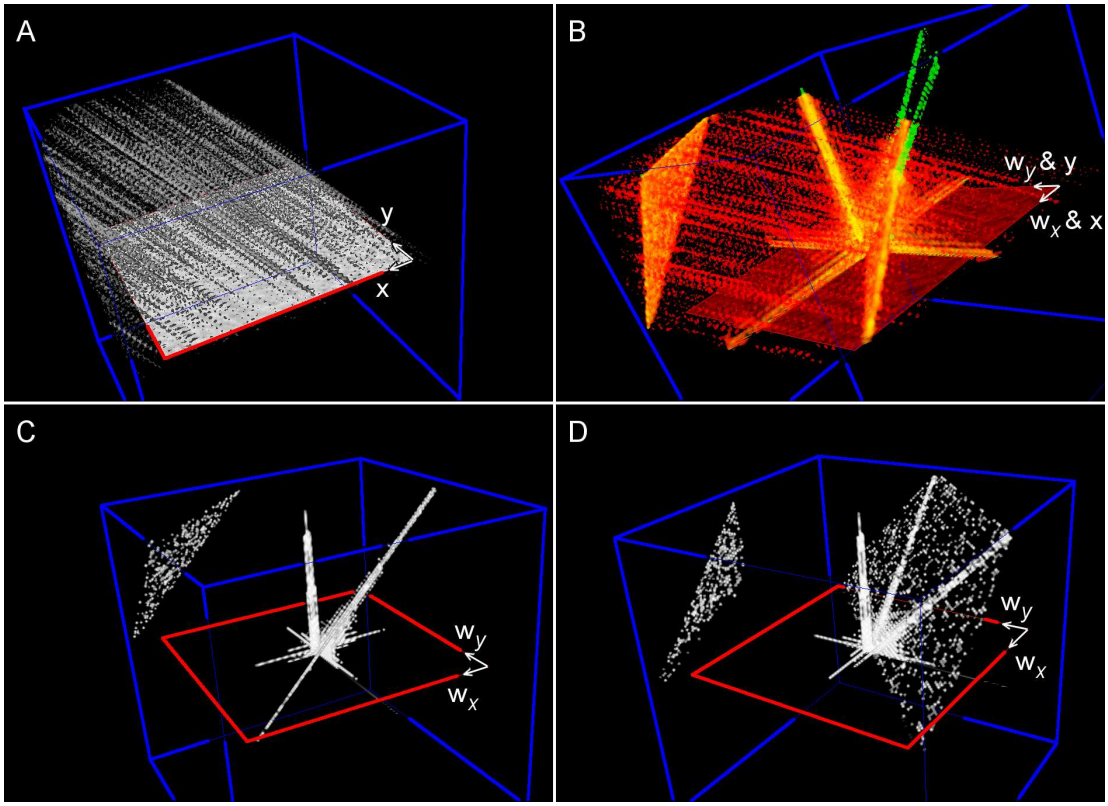


Figure 3.3: Example of analysis of a translating random 2D image in the Fourier domain. In A, the spatio-temporal image is illustrated. For the purpose of visualization only the $I_0(x, y)$ together with trajectories of some brighter patches are displayed. The image translates with velocity $(1, 1)$ pixels per frame. In B, the spatio-temporal image (in red) is overlaid with its Fourier transform (in green) to illustrate the perpendicularity of the motion plane. The plane is given with the two V-style lines, best displayed in D. In C and D, two views of only the Fourier transform are given with a demonstration of motion plane thickness in C. Notice the replica of the motion plane, which is due to temporal alias, in the "left" corners.

The Fourier coordinates must be simply aligned on a plane perpendicular to the velocity vector $(u, v, 1)$. Notice that the motion plane is actually a sheared, not rotated, version of the $\mathcal{F}_{I_0}(\omega_x, \omega_y)$ as only a third coordinate $\omega_t$ is added to obtain the $\mathcal{F}_I(\omega_x, \omega_y, \omega_t)$.

There may be many variations of this expression. For instance, the one in eq. (3.16) is particularly important because it easily completes the description of the motion plane: it shows that the origin of the Fourier domain, i.e., $\omega_x = \omega_y = \omega_t = 0$, always lies in the motion plane whatever velocity the plane represents. The other variant of eq. (3.16) tells the mutual relation between the spatial frequencies $\omega_x$ and $\omega_y$, the temporal frequency $\omega_t$ and the velocity of motion given from the velocity vector $(u, v, 1)$ in terms of their magnitudes:

$$|(u, v)| = \frac{-\omega_t}{|(\omega_x, \omega_y)| \cos \phi} \tag{3.18}$$

where $|(u, v)| = \sqrt{u^2 + v^2}$ and

$$\cos \phi = \frac{(u, v) \cdot (\omega_x, \omega_y)}{|(u, v)| \ |(\omega_x, \omega_y)|}. \tag{3.19}$$

The $\cos \phi$ in both equations is the angle between the spatial frequencies and projection $(u, v)$ of the velocity vector $(u, v, 1)$ into the Fourier domain plane $\omega_t = 0$. The denominator in eq. (3.18) is then a projection of the spatial frequencies onto the axis given by $(u, v)$. Note that this axis is perpendicular to the intersection of the motion plane with the plane $\omega_t = 0$, both lie in the latter plane. The idea we would like to comment on here is, probably, more obvious in the degraded case for 1D, where the eq. (3.18) changes into

$$|u| = \frac{-\omega_t}{|\omega_x| \cos \phi}. \tag{3.20}$$

As the $\cos \phi$ can be either 1, or -1, when both $u$ and $\omega_x$ have, or don't have, the same sign, respectively, we may rewrite it into simplier $u = -\omega_t / \omega_x$. The point of this relation is twofold. Firstly, it shows that the slope of a motion line, in the case of 1D, in the Fourier domain is proportional to the velocity. Secondly, it shows that the ratio between temporal and spatial frequencies is constant. Notably, smaller spatial frequencies imply smaller temporal frequencies and vice versa. In the time-lapse sequences of 2D images, we can read the relation in the similar fashion with the specificity that distance of the projected $(wx, wy)$ from the origin is considered and related with the $w_t$. Such ratio preservation will be worth considering during the design of filter banks.

Let us make one more comment on the derivation of eq. (3.16). We were assuming improper integrals and images of infinite sizes. This is correct from the pure mathematics point of view. But in the reality of computer programs we see that images are always discrete and limited in size. Assuming image of infinite size enabled us to overcome certain issues with boundary conditions, where we would have to define what values to use for $I(x, y, t)$ for which, say, $x - ut < 0$. This is a classical image processing problem with boundaries. We offer an alternative attitude on the derivation and apprehension of the subsequently derived properties. The Fourier transform is a global operation while we aim at computing optical flow, i.e., at computing vectors of *local* apparent motions. We shall see in the next section that filtering the image may be actually regarded as computing a *local* Fourier transforms. In this context, the word "local" means computing

within spatio-temporally bounded volume, which constitues only a small fraction of the volume of the entire space-time image. If the transform is taken far from boundaries, the derivation above will be valid.

**Extracting motion: A parallel with the human visual system**

Humans have been given an incredible gift to immediately measure, or at least guess, a velocity of a translating pattern. Not only that we, humans, can tell if the motion is rightward or leftward, we can rate a distance travelled by comparing some two frames in an, sort of, autonomous automatical manner. A judgement on rapidity and some preliminary velocity estimation is made immediately [23]. Clearly, machines are not that gifted. After the early works [19, 99, 108, 24, 106] on motion estimation have been published in the beginning of 1980s, researches in the machine vision have quite soon identified a processing framework common to the approaches [109]. Even different branches of optical flow computation were found to be somewhat equal or equally powerful [109, 110, 105]. For the computations based on spatio-temporal image representation, the framework has taken the form of the following processing pipeline.

When processing given point in the space-time representation, we really first try to extract orientation of local structure and then combine this information in some small spatio-temporally bounded vicinity to decide on what is the velocity at the given point [111, 106, 27, 112]. The extraction of local structure is done passively by means of applying various orientation selective filters [24, 110]. The spatio-temporal stack is processed with each filter and its responses are stored in a new copy of the stack, this is often called a channel [74, 22, 24, 109]. The idea of filters is that they sweep the stack producing intensive (strong) responses in regions where they detect a structure of interest, e.g., an orientation, to which the filter is tuned. The term filtering really fits here as the stack is processed really only to *catch* [113, 114] regions with structure of interest. Unfortunatelly, responses are not only a function of processed structure but in addition a function of intensity contrast present in the image data [110]. Responses from more channels then must be further processed in order to arrive to a common consensus whether certain orientation is really present in the data. Examples of such processing will be given later in Section 3.3.4.

The identification of a common computational framework doesn't seem to be a coincidence. Most of the approaches, especially the filtering-based ones, admit their inspiration by the human visual system [24, 106, 115] or in the visual system of macaque or cats. Without a doubt, this is a good system to model — we experience it everyday, although we are able to fool it, sometimes. With great simplifications made, the accepted concept is that the processing of observed motion in human visual system happens in three stages: in the early filtering and parallel pathways, in the primary visual cortex and in the middle temporal area [26, 28, 29]. In the early stages, the image is, sort of, sampled with photoreceptive cells at the retina in the eye. Their responses are collected with the ganglion cells from areas that are overlapping. The receptive fields of ganglion cells resemble simple lowpass and bandpass filtering tuned to different frequencies in the spatial and/or temporal domain. Depending on type, cells filter with different filters. All cells work in parallel. Ganglion cell synapse into the LGN (*lateral geniculate nucleus*), which, basically, carry on with filtering with yet another various filtering parameters. Different

LGN further process information from different retina cells enabling to emphasize different aspects of the same visual stimuli (the input image). The cells here discard information about spatial luminance and rather emphasize changes across space and time [28]. Though a common qualitative characteristics of receptive fields of cells at this stage are known, there is about $10^6$ such cells, each with specific slightly differing parameters. They seem to form functional subgroups. As a result lots of locally preprocessed channels are transmitted further to the primary visual cortex, the V1. The V1, which is located in the rear part of human brain, consists of simple and complex cells, which in some sense reflects computational scheme they represent. The simple cells can be modelled with Gabor filters [116, 117] while the complex cells can be modelled as a summation of *quadrature pairs*[2] of simple cells. It is the primary visual cortex where the first orientation selective cells appear. The primary cortex can be regarded as a first place where simple motion patterns are extracted from the observed image. Moreover, cells can be thought of as banks of spatio-temporal filters that decompose the visual world along the spatial and temporal dimensions and, in doing so, determine the envelope of information to which we have access [28]. The processed stimuli as well as many pathways bypassing the V1 continue to subsequent cortical areas V2, V3 and so on until they reach the middle temporal area, the MT or often denoted V5 as well. It is assumed that the understanding of complex motions, relations between moving stimuli, anticipation of motion and similar tasks are happening at this stage.

There is a certain amount of striking resemblance of the typical approach we take in machine vision when compared to the simple description of the model of human visual system. Apart from "digitizing" the world with photoreceptors, the first interesting parallel emerges right in the early stages. It seems that behaviour of the ganglian cells can be approximated with the convolution operation during which a window slides over the convolved image. Secondly, the filtering is applied in a serial fashion, the LGN after the ganglian ones, with the property that LGNs may interact with variously spatially-arranged ganglian cells. This can be simulated with multi-scale approaches. Together it seems that human vision preprocesses input image with some bandpass filtering prior to computation of Gabor quadrature banks, in V1, to estimate reliably spatio-temporal orientation. Finally, some reasoning or evaluation, maybe enhanced with our knowledge and experience converted into anticipation of particular motion, happens in the MT to yield our perception of what we see. This corresponds with what we have said at the beginning: we first try to extract orientations by means of bandpass filtering (recall the representation of orientation in the Fourier domain) and then combine the results to estimate velocities.

Note that there is also one striking difference between the two processing pipelines. It is the fact that human visual system doesn't seem to work "on demand". For instance, instead of detecting a magnitude of observed movement and appropriatelly modifing receptive fields of cells at any level of the model, human brains simply seem to provide an impressive amount of variously tuned cells containing also the at-the-moment-appropriate ones. All the responses are then processed, often filtered again, in many different patterns and many levels until a perception is build. The difference is in that we, in contrast to our vision, currently can not compute and further process such an amount of filtering operations in parallel and in real time.

[2] According to Freeman and Adelson [124]: "A pair of filters is said to be in quadrature if they have the same frequency response but differ in phase by 90° (i.e., are Hilbert transform of each other [138])."

### 3.3.2 Filters to detect motion

#### Gabor filters in the space-time domain

Following the parallel with the human visual system, we will focus on simulating the filtering done in the V1 area. We are seeking predominantly a filter with parameters close to the obtained receptive fields data of the simple and complex cortex cells. The filter should be linear to achieve fast computation, eventuelly. Secondly, it should have good parameters in the Fourier domain because the motion of arbitrary patterns can be, probably, best detected via its (local) Fourier transform. When designing a filter bank, a great deal of attention is given to how the bank samples the Fourier domain [118, 119, 98]. From the analysis of the human visual system [28] and the analysis of motion representation in the Fourier domain [24] at the same time, it follows that the filter should be able to create a quadrature pair.

The Gabor filter seems to be a good adept [110, 116, 118, 111]. The 1D Gabor filter is given as

$$\mathcal{G}b(x,\sigma,w) = \frac{1}{(2\pi)^{1/2}\sigma}\ e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\ e^{i2\pi wx}. \tag{3.21}$$

The filter, originally defined by Gabor [120], encompasses the carrier $e^{i2\pi wx}$ of frequency $w\mathrm{px}^{-1}$ within the Gaussian envelope $e^{-x^2/2\sigma^2}$, see Fig. 3.4. Its generalized form, however, is probably due to Daugman [116] who was seeking a spatial linear filter with optimal conjoint limits on resolution of orientation, spatial frequency and two-dimensional spatial position. Citing from his work [116]: "Each such filter occupies an irreducible quantal volume (corresponding to an independent datum) in a four-dimensional information hyperspace whose axes are interpretable as 2D visual space, orientation, and spatial frequency, and thus such a filter set could subserve an optimally efficient sampling of these variables." In this light, the human visual system appears to enjoy optimal, in accordance with the information theory, encoding of the visual stimuli for subsequent processing [121]. It matches well the receptive fields of simple cells [116, 117]. Its real and imaginary parts form together a quadrature pair. It is also bandpass limited in the Fourier domain. For derivation of further descriptive parameters, such as bandwidth and peak response, of given Gabor filter, we would direct the reader to the nice introductory publication by Movellan [122].

The Gabor filter is easily generalized to $n$D:

$$\mathcal{G}b(\vec{x},C,W) = \frac{1}{(2\pi)^{n/2}|C|^{1/2}}\ e^{-\frac{1}{2}\vec{x}^T C^{-1}\vec{x}}\ e^{iW\vec{x}} \tag{3.22}$$

where $C$ is a $n \times n$ Gaussian (symmetric and positive definite) covariance matrix, $W$ is a $1 \times n$ row matrix with frequency tuning and $\vec{x}$ is a coordinate column vector in the $n$D space. The matrix $W$ often takes the form $[2\pi w_1, \ldots, 2\pi w_n]$. The superscript $^T$ denotes a transposition.

When dealing with time-lapse sequences of 2D images, we are using 3D Gabor filters, technically. It is then advantageous [P2] to define tuning of a Gabor filter with parameters directly related to parameters of the motion for which the filter is supposed to be the most sensitive. We propose to use two angles $\alpha$ and $\beta$, as shown in Fig. 3.5, to steer the direction of the main axis of the filter envelope. It is well-known that any Gaussian

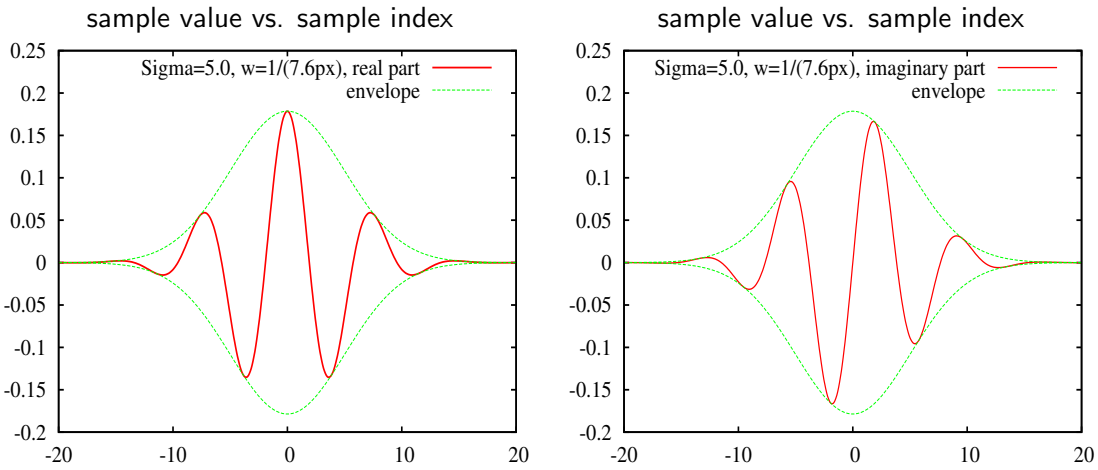


Figure 3.4: Example of the complex 1D Gabor filter with $\sigma = 5.0$ and frequency $w = 1/7.6\text{px}^{-1}$. The filter consists only of the red curves (darker in B&W print). The thinner green lines (lighter in B&W print) outline the Gaussian part of the Gabor filter. Note that they wrap the filter. Hence, it is called the Gaussian *envelope*.

is always separable[3] along its major axes. We define one of them to be the main axis, the referential axis in other words. Notice, in Fig. 3.5, that the direction along which the complex exponential propagates is perpendicular to the main axis. Details on constructing such a steered filter, i.e., details on obtaining $C$ and $W$ for eq. (3.22), are given in Section 2 of our original publication [P2].

This concept is not limiting [32]. Suppose any arbitrary $n$D Gabor filter is given. In particular, any matrix $C$ is given to define arbitrary Gaussian envelope. The Gaussian restricts it to always be a positive definite symmetric matrix. As such, there always exists decomposition $C = R^T E R$ where $R$ is a rotation matrix and $E$ is diagonal [123]. The decomposition is achieved with the Singular Value Decomposition technique, the SVD. The rotation matrix $R$ has column vectors orthogonal to each other and each is of length 1 [32]. The column vectors form an orthonormal basis. This has interesting consequences. It holds[4] $R^T R = I$, with $I$ being the square identity matrix, thus allowing for $R^T = R^{-1}$. The inverse $C^{-1}$ then takes the form of $R^T E^{-1} R$. If we think of $R$ as of a transition matrix, $\vec{u} = R\vec{x}$, from the (orthogonal) coordinate system of an image, with coordinate in $\vec{x}$, to a new (orthogonal) coordinate system given by the column vectors of $R$, with coordinate in $\vec{u}$, the inverse $C^{-1}$ in the Gaussian's exponential can be regarded as

$$e^{-\frac{1}{2}\vec{x}^T C^{-1}\vec{x}} = e^{-\frac{1}{2}\vec{x}^T (R^T E^{-1} R)\vec{x}} = e^{-\frac{1}{2}(R\vec{x})^T E^{-1} R\vec{x}} = e^{-\frac{1}{2}\vec{u}^T E^{-1}\vec{u}}. \tag{3.23}$$

Since $E$ was assumed to be diagonal, say $E = diag(\sigma_1^2, \ldots, \sigma_n^2)$, the shape of Gaussian

[3] A filter is said to be separable if the same convolution result can be achieved after at least $n$ 1D convolutions conducted in a serial fashion, i.e., next convolution is applied on a result of a previous one. Typically, it is expected that filter can be separated into exactly $n$ convolutions that are computed along the coordinate system axes. The order is not important.

[4] Due to the transposition, the $i$th column of $R$ becomes the $i$th row of $R^T$. Hence, values on the diagonal of the multiple $R^T R$ are squared lengths of vectors, which we know is 1 due to the assumed constraint on length. Values apart from the diagonal are dot products of different vectors, which we know is 0 due to the assumed orthogonality.

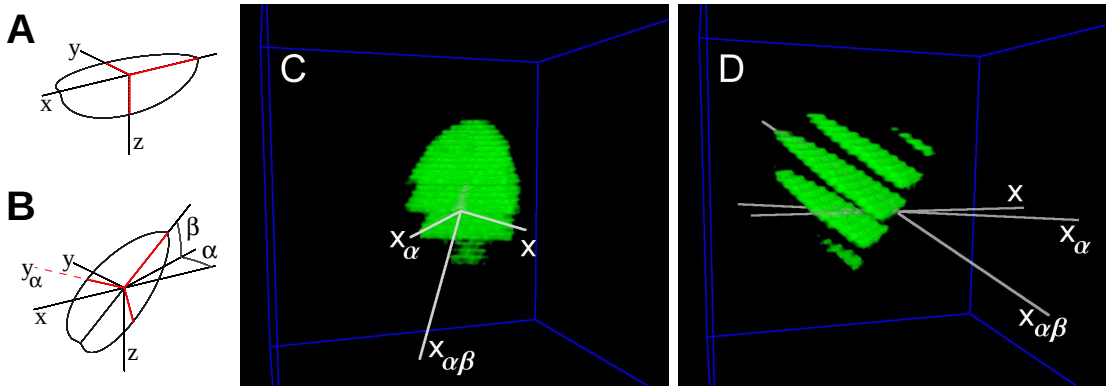


Figure 3.5: Steering an anisotropic Gabor filter. At first, consider the filter envelope centred at the origin of the (spatio-temporal) coordinate system and with its main axis aligned with the $x$ axis, in A. Desired filter tuning is obtained by rotating the envelope by angle $\alpha$ around the $z$ axis (around the origin). The coordinate system is never rotated but suppose for a moment that $y$ axis was rotated by $\alpha$ to obtain new axis $y_\alpha$. We rotate the filter envelope by angle $\beta$ around this new axis, shown in B. We refer to this second rotation as to the tilting of the filter. In C and D, there are two views on the same 3D Gabor filter. The filter main axis is initially aligned with the $x$ axis. After the rotations, it aligns with the $x_\alpha$ and $x_{\alpha\beta}$ axes, respectively. For the visualization, only positive lobes and only half of the filter is shown. When the filter is finally tuned, its main axis aligns with the axis denoted $x_{\alpha\beta}$ in C and D.

envelope is now clearly apparent by means of $\sigma_i$ along its major axes, which are identical to the new coordinate system axes. Any arbitrary Gaussian covariance matrix $C$ can be regarded to be, in fact, only a composition of simple obvious diagonal matrix extended with an implicit incorporated coordinate system transform.

### Gabor filters in the Fourier domain

The 1D Gabor filter, eq. (3.21), is a multiplication of two terms: the normalized Gaussian envelope with zero mean and the carrier, which is represented with complex exponential function. The Fourier transform of the multiple, $\mathcal{F}_{\mathcal{G}b}(\omega, \sigma, w)$, can be regarded as convolutions of Fourier transforms of individual terms. Further advancing the idea, the transform of Gaussian is Gaussian again with modified normalization constant and inversed sigma. The transform of a complex exponential is a single point[5] in the Fourier domain positioned at the frequency $w$ given in the exponential. In fact, it is the Dirac delta function $\delta(\omega - w)$. Finally, convolution on a single point produces a copy of the mirrored convolution kernel centred at that point. In fact, the impulse response is computed, one may refer to eq. (4.2) on page 61 for the formula of convolution with Gaussian kernel, for convenience. To sum it up, the Fourier transform of *complex* 1D Gabor filter is [25, 122]

$$\mathcal{F}_{\mathcal{G}b}(\omega, \sigma, w) = e^{-\frac{1}{2}(\omega - w)^2 \sigma^2}. \tag{3.24}$$

The Gabor filter's bandpass is exactly the same as the bandpass of its enveloping Gaussian except for that the Gaussian is always centred at the zero (origin of the Fourier domain)

[5] This can be derived from the Fourier transforms of $\cos(2\pi\omega x)$ and $\sin(2\pi\omega x)$, from the Euler equation $e^{ix} = \cos(x) + i\sin(x)$ and from the fact that the Fourier transform is linear.

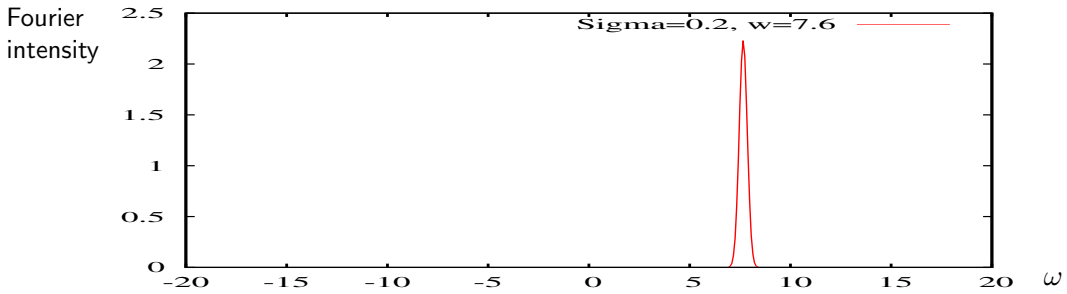


Figure 3.6: Example of the Fourier transform of the complex 1D Gabor filter with $\sigma = 5.0$ and frequency $w = 1/7.6\text{px}^{-1}$. The curve is (real) Gaussian with $\sigma = 0.2$ with its centre offset at $w = 7.6\text{px}^{-1}$.

while the Gabor filter's bandpass is positioned at some offset given by the frequency tuning of the filter, Fig 3.6. This is also supported by the fact that the complex filter is transformed into a pure real function.

The transform of a complex Gabor filter is in the general $n$D case of the form:

$$\mathcal{F}_{\mathcal{G}b}(\vec{\omega}, C, W) = e^{-\frac{1}{2}(\vec{\omega}-\vec{w}^W)^T D^{-1}(\vec{\omega}-\vec{w}^W)}, \tag{3.25}$$

simply "some" Gaussian centred at the Fourier domain coordinate $\vec{w}^W$. We had to adopt the notation $\vec{w}^W$ for a *column vector* to represent content of $W$ because we have defined $W$ to be a *row matrix*: $(\vec{w}^W)^T = (w_1, \ldots, w_n) \iff W = [w_1, \ldots, w_n]$. Note that $\vec{\omega}$ is a column vector representing frequency coordinate. As the transform is a Gaussian, its covariance matrix $D$ can be decomposed as explained in the previous section, eq. (3.23). In addition, we know that major axes are kept and only sigmas are inversed in the Fourier transform of a Gaussian. Hence, we write $D = R^T E^{-1} R$ while we assume $C = R^T E R$. Substituting result from the latter, $E^{-1} = RC^{-1}R^{-1}$, into the former, we obtain $D = R^T RC^{-1}RR^{-1} = C^{-1}$. The transform of a complex Gabor filter, eq. (3.25), is in the general $n$D case of the form:

$$\mathcal{F}_{\mathcal{G}b}(\vec{\omega}, C, W) = e^{-\frac{1}{2}(\vec{\omega}-\vec{w}^W)^T C(\vec{\omega}-\vec{w}^W)}, \tag{3.26}$$

A more elaborated example of the Fourier transforms of some Gabor banks will be given later on page 81 in Fig 4.11.

### Alternatives to Gabor filter

A result of convolution with complex Gabor filtering is a complex valued image. But the complex result is typically not used directly. Instead, it is dealt either with the magnitude (energy) or with the phase of it. This is where the division of filtering-based optical flow methods to energy-based and phase-based have come from [72]. However, both approaches have in common the struggle with computational demands of convolution with Gabor filter. We will show later in the text, in Section 4.1.3, that convolution with Gabor filter can be very time demanding if the filter is not of some special form, e.g., isotropic Gaussian envelope or carrier propagating along some of coordinate axes. Moreover, the computation may require many frames to process, i.e., the filter has large temporal support

and so requires many frames of the sequence to be kept in the memory at the same time. Researchers have tranditionally attacked this fact by using Gabor filters in some "basic" simple forms and/or by replacing the true Gabor filter with some, possibly recursive, approximations. We will review now some of these alternative approaches.

First of all, an interesting concept of, so called, steerable filters was introduced by Freeman and Adelson [124]. They proposed a framework that allowed for synthesizing output of a filter tuned to arbitrary orientation from outputs of a small set of basis filters by means of linearly combining them. Clearly, the concept has certain requirements. Unfortunately, any Gabor filter does not meet them [125]. On the other hand, a tuned 2nd derivative of Gaussian as well as a filter with whom a quadrature pair it forms can be computed using the concept. The former filter resembles a real part of complex Gabor filter while the latter resembles the imaginary part. In the 2D case, the arbitrarily oriented 2nd derivative Gaussian can be computed with only 3 separable spatial filters. The other filter can be computed with 4 separable filters.

The derivatives of Gaussian were utilized also in the method by Weber and Malik [104]. The advantage of Gaussian filter and its derivatives is that they are separable and, consequently, convolution with them can be computed rather fast, we'll explain that later in Section 4.1.3. Spinei *et al.* [119] and later Bruno and Pellerin [98] used the separability of Gaussian as well as the fact that convolution with Gabor filter can be also computed as a convolution with Gaussian filter, we will cover that again in Section 4.1.3. This is an exact replacement, no approximation. They further replaced the Gaussian filtering with recursive Gaussian filtering. But the recursive filtering *is* an approximation of the true Gaussian filtering. As the approximation is not the exact filter though it is rather accurate [126], the whole filtering pipeline does not compute an exact Gabor filtering. Since this is such a subtle change in the filtering, we may argue whether it is still Gabor filtering or an autonomous alternative. For instance, Bruno and Pellerin denoted it as a Gabor-like filtering, Spinei *et al.* did the opposite.

A solution, pioneered by Watson and Ahumada [23], made use of space-time separable Gabor-like filtering, i.e., the spatio-temporal 3D filter is broken down into a cascade of two 1D spatial filters and one 1D temporal filter, in which they used some *causal* temporal 1D filter. A causal filter requires only current and previous input values to compute its response. An anti-causal filter, on the other hand, requires in addition also future input values and so a delay is introduced before the response can be computed. The true 1D temporal Gabor filter is an anti-causal one. Adelson and Bergen [24] followed with the second and third order derivatives of Gaussian, spatial filters were anti-causal while temporal one was causal again. They showed a simple scheme to construct orientation selective Gabor-like quadrature pairs. Fleet and Langley [127] replaced the 1D component with recursive filter. In particular, when processing a 2D time-lapse image sequence, a complex spatial 2D Gabor filter is applied on every image in the sequence. Afterwards, the result is processed along the temporal axis. The advantage of recursive filtering in this application is predominantly in that it has very short temporal support, i.e., only a few frames must be kept in the memory, and that it is computed faster [112]. Furthermore, Clifford *et al.* [128] adaptively modified the temporal component tuning to optimise for the assumed local motion in the space-time image according to a previously measured velocity. Gautama and van Hulle [129] required to compute phase derivatives from result of a complex Gabor bank filtering. They conducted only the 2D spatial filtering, from which

the spatial phase gradient was obtained, completely leaving out the 1D temporal filtering. They instead, similar to the recursive filtering, aggregated phase difference based on which they estimated current temporal phase gradient. It is noteworthy that this approach has been recently re-implemented to work on GPU [107]. The authors claim that it achieved real-time performance and accuracy near the established differential methods such as the TV-L1 [101] or the one by Papenberg *et al.* [93].

Austvoll [84] used for his phase-based method complex directional filters with envelope approximated with Kaiser windows [130]. The advantage of this windowing function is that it has limited support, unlike Gaussian which falls down to zero at infinity and is, therefore, in practice always truncated. Austvoll and Nayar [131] have, however, observed that their optical flow method gives twice worse results when an IIR variant of filters is used. Furthermore, the directionality of filtering is achieved by rotating a space-time image around the temporal axis, i.e., technically around the $z$ axis.

Another alternative to complex Gabor-like filters is their extension, the monogenic signal [132]. While a Gabor filter is localized in the Fourier transform in the form of a convex bandpass blob, the monogenic signal forms a radial bandpass torus [133]. The bandpass consists of frequencies of a certain range in absolute value but with arbitrary direction. Instead of convolving with a quadrature pair of filters, a spherical quadrature triple is used. The approach should overcome the limit of a Gabor filter, which is that local phase can be estimated from its response partly successfully since the local orientation has to be known in advance to steer the filter. The monogenic signal can estimate the local orientation and the local phase [133]. The resemblance with Gabor filter is in the profile it has as it encompasses the Gabor filter at all phases. In particular, even and odd Gabor filters are "simulated" as well. While this looks like an interesting concept, we do not see the concept becoming wide spread in the literature at the time of writing.

Let us note that all of these *variants are* actually *trying* to approximate the Gabor filter — in contrast to Gabor filter approximating some of the variants above. Moreover, the Gabor filter has pleasant mathematical properties, e.g., the Fourier transform is given explicitly with closed-form formula with pleasant properties itself, and theoretical properties, e.g., it reaches the lower limit for the joint entropy for linear spatial filters [116]. It is also "biologically" motivated. This is why we have decided to focus directly on complex Gabor bank filtering in this thesis as a local orientation estimator for filtering-based optical flow computation. If we manage to achieve fast and correct implementation of the Gabor filtering, the motivation for the use of any of the above approximating variants will be void.

### 3.3.3 Applying filters to estimate motion

In this section, we would like to show how to combine Gabor bank filtering with the spatio-temporal representation of motion. We will show it in the context of the energy-based approach because, in our opinion, it is more illustrative compared to the phase-based approach (filter response is presented with its energy) and it appears to directly target at the orientation patterns in the space-time image.

### The principle of the energy-based approach

The energy-based approach to motion estimation is a way a complex Gabor filter is applied to instantaneously and quantitatively judge on the presence of local orientation in the spatio-temporal representation. We aim to show here only the principle by demonstrating it on different filter banks without any ambition to draw conclusions on their appropriacy. An example, where filtering responses are compared, will be given later in Fig 4.10 on page 80.

We begin with the example of spatio-temporal representations of 1D translating patterns, consider Fig. 3.7. The representation, the space-time image, is convolved several times, each time with different filters. It is to be understood that the filtering is done in a parallel fashion, where copies of input image are filtered exactly once, rather than in a serial fashion, where single copy is iteratively convolved each time with a different filter. Many (parallel) copies of the convolved input image, the channels, are created in this way. The filters may be rather of diverse shapes. But they always should be tuned to respond to some interval of local orientations. To understand this interval, it is advantageous to consider the whole situation in the Fourier domain.

Moving bars give rise to motion lines in the Fourier domain in the 1D+t case, just like moving 2D patterns give rise to motion planes in the 2D+t case [23, 111, 83]. The spatio-temporal and Fourier representations of the same motion are fortunatelly interconnected with the important invariant property of "mutual perpendicularity" [22], eq. (3.17). Thus, as bars moving with different velocities leave traces of different slopes in the space-time image, they also induce motion lines of different slopes with respect to the $\omega_x$ axis, Fig 3.7B, in the Fourier domain. Our aim, basically, is to detect the presence of such line and then estimate its slope by, sort of, sampling the Fourier domain with the filter bank, Fig. 3.7C,D. In other words, we are inspecting the Fourier domain through a few (weighted) windows, i.e., through the green (bright) blobs in the figure. If a blob is a product of Fourier transform of a Gabor filter, the blob is always convex. Moreover, as every motion line passes through the origin ($\omega_x = \omega_t = 0$), there exists certain range of slopes that define lines that pierce this blob. This is the interval of local orientations detected by the examined filter.

Unfortunatelly, real Gabor-like orientation selective filters are phase sensitive [24]. Since such filter includes the oscillating carrier, its response to a moving pattern depends on how the pattern happens to line up with the carrier at each moment. The response can be positive, negative or even zero, e.g., when the pattern would form the same carrier delayed by a quarter of carrier period. As a result, we can't judge on presence of local orientation directly from instantaneous responses. Instead, a pair of the same filters with the same envelopes and with frequency carriers out of phase by a quarter of the carrier period, i.e., by $\pi/2$ radian, is used. It is a quadrature filter pair [124]. If the carrier is based on basic trigonometric function, say it would be the sine function, the other in pair would be the cosine function. Instead of computing a single convolution with just the single real filter, we instead conduct two convolutions in parallel, once with the first and once with the second filter from the pair, square their responses and pixel-wise sum them. Such filtering result is called the energy, see Fig. 3.7K–P for example. Owing to the quadrature and the fact that $\sin^2 x + \cos^2 x = 1$, the instantaneous response, the energy, is stabilized and reflects the presence of local orientation proportionally. From the Euler

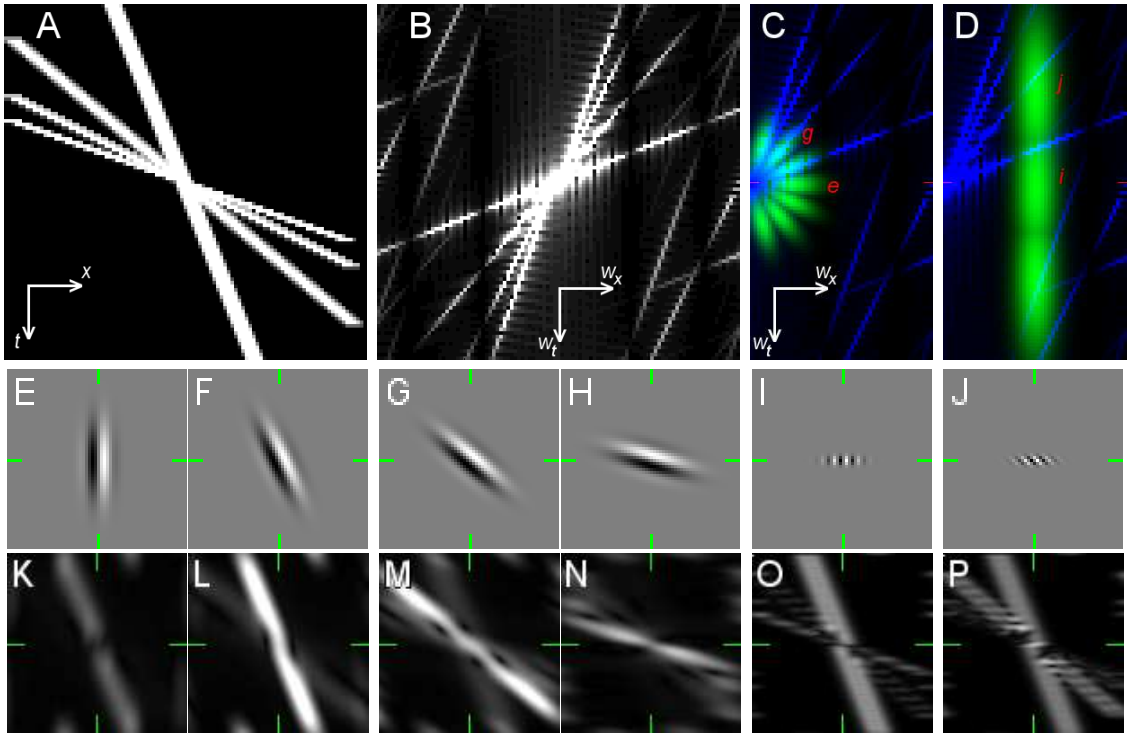


Figure 3.7: Illustration of application of filtering to motion estimation. The upper row shows several image overlays. In A, the spatio-temporal representations of four 1D rightward translating patterns are shown. From different slopes of the bars, we see that each pattern travelled with different velocity. Namely these were 0.4, 1.2, 2.0 and 2.8px/frame, respectively. The pattern was always of the same width. In B, image overlay of four respective Fourier transforms is shown. Notice, in A and B, that bars moving with velocities starting from 2px/frame tend to appear with similar orientations. In C and D in green colour (in bright intensities in B&W prints), two montages of right half-planes ($\omega_x \geq 0$) of Fourier transforms of two sample filter banks are shown. Every blob corresponds to one filter in a bank. In blue colour (in dark intensities) are the motion lines, from B, induced by the translating 1D bars. The middle row, in E–J, presents a collection of the space-time images of filters from the two banks while the lower row, in K–P, presents a collection of the energy responses of these filters. The coordinate systems here are the same as in A. Only the filters tuned to stationary and rightward motions are considered. Only the imaginary (sine/odd phase) parts from the filters are shown, in E–J. In C and D, one may notice lowercase ilatic letters nearby some four blobs that denote which spatio-temporal filter representation gave rise to the respective blob. All images were enhanced for printing purposes. The images K–N have intensities stretched by factor 1.3 while the images O and P have intensities stretched by factor 4.0. The brighter a pixel in the image K–P is, the stronger response on the input image A the respective filter gives.

formula, the real and imaginary parts of a complex Gabor filter form a quadrature pair.

Before we proceed further, we owe to explain a bit about the (in)compatibility matter present in the figure. It is due to the fact that the Fourier transform is a *global* operator while Gabor filtering, just like convolution with any other kernel, is a *local* operator. Hence, when investigating what is the prevailing motion at any given pixel of the space-time input image by means of inspecting its Fourier domain representation, we must consider Fourier transform taken only over a small vicinity of this pixel, a *local* Fourier transform. Otherwise, the responses in the Fourier domain may be dominated by pixels at spatio-temporally distant locations in the input space-time image (because the transform is computed over the whole image) and so the responses may be significantly different compared to the local Fourier transform. In this respect, the direct investigation of the Fourier transform of the space-time image, Fig. 3.7C and D, is correct only under the assumption that the image, Fig. 3.7A, represents only a small region around the investigated pixel. We will show, in Section 4.1.3 on page 71, that using a number of complex Gabor filters can be regarded as computing local Fourier transforms for the number of frequencies. Unfortunatelly, only large regions, such as the whole image, allow for nice and apparent motion lines. One should rather evaluate local Fourier transforms for several frequencies prior to designing an optimal filter bank.

**Some remarks on designing motion estimating filters**

When designing the filter bank, there are a few aspects worth considering. First of all, greater velocities produce motion lines of similar slopes in the Fourier domain. This is a consequence of eq. (3.20), from which it holds $u = -\omega_t/\omega_x$. For greater velocities $u$, a change in $\omega_t$ induces a smaller change in $\omega_x$, which, in turn, gives rise to rather vertical motion lines, Fig. 3.7B. Note that all lines pass through the origin in the Fourier domain. It is then somewhat harder to distinguish between them. They are more apparent in regions of high temporal and small spatial frequencies. Unfortunately, the high temporal frequencies face the barrier originating from the discrete nature of time-lapse image processing. The greatest frequency is $2\text{px}^{-1}$, which comes from the limit on the smallest practical wavelength of the period of 2px. Other limit, so typical for the temporal frequencies, stems from the rate of temporal sampling during acquistion of the time-lapse sequence, i.e., the frame rate and, consequently, the temporal resolution. An artificial increase in temporal resolution may be achieved by interpolating some missing frames. Filters tuned to greater temporal frequencies can be used then. As the velocity connects the space domain with the time domain, another work around may be to decrease the spatial resolution, e.g., by factor of 2, while keeping the temporal one. The velocity then appears slower in such down-sampled space-time images. This is a preferred solution because the spatial resolution is typically far better and the decimation by factor of 2, 4, or even 8 sometimes, still preserves the main features of the image data [109]. Moreover, instead of adding new full-size frames, we rather add new half-size ones. Since any of the two solutions do not add new information, the latter solution then appears more efficient. A scale space data representation, a pyramid, is built in this way [76, 74]. Furthermore, the same set of filter banks, specifically designed to estimate local orientations corresponding *only* to slower velocities, may be used at all levels. On the other hand, motion estimation at the reduced levels has also reduced accuracy due to the reduced resolution.

Owing to the utilization of the scale space approach, we are able to focus only on some type of motion estimating filters. In particular, we focus on the filters tuned to smaller velocities, let us say less than 1.2px/frame. Since smaller velocities produce motion lines that range from strictly horizontal slopes, when there is no motion present, up to $+/-50°$, when there is 1D rightward/leftward motion of up to 1.2px/frame, we may construct a filtering set similar to the one in Fig. 3.7I,J, or D, respectively. Evidence on human visual system, however, supports rather radially arranged ensemble of filters [116, 118], just like it is illustrated in Fig. 3.7E–H, or C, respectively. Following Daugman [116], and his result on Gabor filter's property of "irreducible quantal volume in the conjoint space-time and frequency domain hyperspace", and from eqs. (3.22) and (3.26), we observe that a filter is either small in the Fourier domain and large in the space-time, or vice versa. It can't be small in both domains simultaneously. This is a dilema. The radially arranged filters seem to be more orientation selective and less prone to detect aliases, thus better suited for motion estimation. Notice, in Fig. 3.7K–N, that such filter bank managed to extract different orientations well. But such filters, when tuned to small velocities as in insets E and F, require rather long temporal support in the space-time at the same time. Thus, it sets a requirement on minimum but still large number of frames present in the time-lapse sequence. In addition, longer temporal support has increased probability that the constant motion assumption will be violated. The constant motion assumption comes from the fact that the filters are designed to detect spatio-temporal orientation of straight, not curved, motion traces. Grzywacz and Yuille [115] suggest to prefer filters of short temporal and rather long spatial support. The filters of the sort shown in Fig. 3.7D have short temporal support, shown in insets I and J. Their Gaussian envelope was given with $\sigma_t = 1.0$, which constraints their minimum temporal support to 7 frames. The radially arranged group of filters has, on the other hand, support of up to 70 frames. By the way, the filter set shown in Fig. 3.7D is (1D+t)-version of the one used in the now-classical energy-based method by David Heeger [111, 106]. In the energy part of our example given in insets K–P, the Heeger's filter bank has rather balanced responses for the smallest velocity. The forward movement filter, inset J, also includes stronger response to velocity 1.2px/frame, inset P, while the "stationary" filter, inset I, also includes response to "false" motion, the motion alias of velocity 2.8px/frame, because the corresponding motion line's alias really pierces the middle green blob (nearly in the centre of the inset D). The Heeger's bank seems to be less selective to different local orientations but it is very economical in terms of its temporal support.

Notice that the discussion above applies even for the human visual system. For example, consider two cars moving on the street, one is going rather fast while the other is going very slow. Looking at the street, which car will we first realize as moving at all? And if we are further interested in details on the car, on which car the details will be easier to see? Details is a content of higher spatial frequencies. An extreme example is the noise, which itself can be regarded as very detailed textural information, though unwanted. It is well-known that noise can be suppressed with lowpass filters. Slowly moving cars tend to preserve intensities of pixels at given spatial coordinates over certain short period of time. Hence, slowly moving cars tend to occupy smaller temporal frequencies. In this respect, we may realize that humans see details (higher spatial frequencies) easier on slowly moving cars (small temporal frequencies) while details are, kind of, blurred (small spatial frequencies) on fast moving cars (high temporal frequencies). Clearly, the latter is

strongly dependent on the magnitude of car velocity. Note that it corresponds well with the motion lines in Fig, 3.7B, especially well with lines associated with faster motions. Citing from Bigün's work [118]: "The linear cells in the visual cortices of primates are very sensitive to i) gratings with high spatial frequencies moving slowly and ii) gratings with low spatial frequencies moving fast."

Moreover, if we turn our head with the movement of the fast car, we, sort of, introduce a "camera movement" to decrease the relative observed velocity. Despite that we aim to increase our ability to read details on the fast car, we often experience that we actually see better but still somewhat worse compared to the situation with the slowly moving car. This can be simulated with the scale space approach given above. Anandan [109] pointed out the following principle: "Large-scale image structures can be used to measure displacements over a large range with low accuracy and at a low sampling density, while small-scale image structures can be used to measure displacements over a short range with higher accuracy and at a higher sampling density." This is in accordance with our presented idea of the scale space approach. The first part of the sentence refers to a coarse representation of the space-time image, i.e., the copy of it at some higher level in the scale space, where we find half-sized frames with lower spatial resolution. In such a copy, only large-scale structures have survived the down-sampling and so they are dealt with here. Small displacement here is, owing to the low resolution, translated into larger physical distance than the same displacement in the original space-time image. This refers to the "over a large range with low accuracy." The latter is again due to the low resolution. As the down-sampled copy was created to contain only motions at slow velocities, their orientations in the spatio-temporal representation are almost vertical, i.e., along the temporal axis. Hence, we may resample in this axis to decrease the number of frames a bit while leaving the motion traces still apparent and continuous.

Finally, let us return to the first question on the delay we, humans, need for estimating velocity of motion as this can be regarded as the size of the temporal support of some cells in the V1. Clearly, we immediately recognize that one car is moving fast but we are not certain how fast it is. In contrast, we tend to inspect the motion of the slow car quite a while, compared with the other car. For instance, consider the pedestrian crossing: we immediatelly decide not to step into the road when a car is approaching fast while we tend to hesitate whether we make it or not when a car is approaching very slowly — probably because we need to accumulate the motion history to obtain an estimate on the velocity, which is later processed in the brain, which itself also significantly contributes to the delay before we eventuelly decide. Note that the discussion assumes regular temporal sampling of the human visual system.

### Conclusion on filtering to estimate motion

Anyway, the above discussed ideas about filter shapes and how to apply them have implicitly outlined certain, rather general [109], computational model on detecting local orientations in spatio-temporal representations of time-lapse sequences. The model suggests to decompose space-time images into several channels computed at a few spatial scales. Every channel is sensitive to some local orientation, whose presence is proportionally signified in every pixel with its intensity. Every channel can be characterized with its spatio-temporal bandpass properties in the Fourier domain as well. All channels taken

together over all scales are then expected to factor the Fourier domain well.

The discussion above arrived to the two main conclusions. It showed that it is worth considering the scale space approach in which the same set of filter banks may be used at all levels, the set should be tuned predominantly to slower velocities. And, it explained what type of spatial structures or patterns we should look for when we focus on detecting certain local orientation in the spatio-temporal representation, i.e., when we focus on certain velocity. As a consequence, it seems that it is worth designing complex Gabor filters of the "radial type" [116], tuned to motion planes of the slope between $\pm 50°$ and localized at higher spatial frequencies in the Fourier domain [118] and with short temporal support and reasonably long spatial support [115] in the space-time domain. Owing to the interconnection between velocity, slope of the motion plane, spatial and temporal frequencies [22], we can compute what should be an optimal temporal frequency, eq. (3.18) on page 39, for a filter once we decide what should be its spatial frequency and velocity to be tuned to. In the case of 2D+t, every filter is tuned to local orientation corresponding to some 2D spatial direction of movement with velocity below 1.2px/frame. Filters tuned to the same magnitude of movement form a filter bank, see Fig. 3.8 for an example of such a bank.

The filter bank is a must. Firstly, we have noted earlier that responses are also a function of intensity contrast present in the image data [110]. By comparing responses among filters we easier recognize whether a given response is predominantly due to the local orientation or not. Secondly, we have noted earlier that we are, sort of, inspecting the local Fourier domain through a few windows, through a few apertures in other words. Every channel represents one such aperture. Since there can be many motion planes that pierce given single aperture while producing (nearly) the same responses, we need to use more apertures in order to correctly and reliably estimate a motion plane. Clearly, we seek motion plane that is consistent with responses of all channels. The performance is greatly influenced by a distribution of the apertures in the Fourier domain. An example of how a motion plane intersects a filter bank is given in Fig. 3.8.

The suggested model together with the suggested filter setting seems viable for motion estimation. As it aims to mimic the human visual system we notice, from discussions in the previous sections, that it, at least, allows for the same features.

Note that the collection of instantaneous energy responses, which is the result of application of the complex Gabor bank filtering, must be further combined to decide on what is the velocity at every pixel of the space-time image. This follows the processing framework outlined in Section 3.3.1 on page 40. Examples of complete methods are subject of the next section.

### 3.3.4 Examples of filtering-based optical flow methods

Depending on how we treat the quadrature pair filtering results we recognize two types of filtering-based optical flow computation methods: the energy-based and the phase-based. The former works with magnitudes (energies) computed on the complex results while the latter uses their phase. The former is represented with the method by David Heeger [111, 106] while the latter is represented with the method by David Fleet and Allan Jepson [83]. Both methods have in common the intensive use of filtering of the space-time image. Thus, both methods split the space-time image into several channels. They essentially

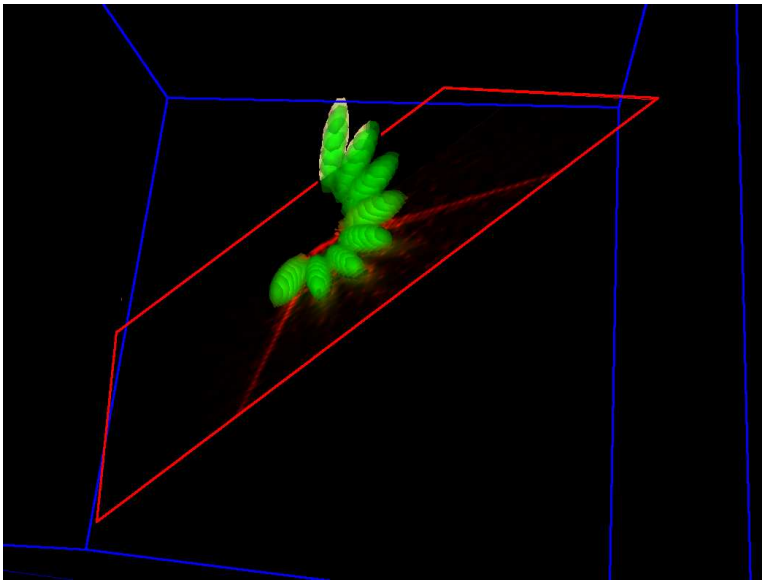

Figure 3.8: Example of 3D filter bank shown in the Fourier domain. All filters in the bank were tuned to the same magnitude of movement, i.e., to the same slope of the motion plane, but to different spatial directions, or headings, of the movement, i.e., to different projections of the normal of respective motion planes into the plane $\omega_y = 0$. The motion plane is represented with the red frame, it represents the movement shown in Fig. 3.2 in the right and in Fig. 3.3C,D.

differ in the next stage in the way results from the channels are combined to yield velocity estimates.

### The energy-based example

Basically, the energy-based approach is based on the "Fourier nature" of a motion as it tries to estimate slope of the motion plane from a few observations made via the filters/channels. Heeger [106] used the Parseval's theorem[6] to be able to directly compare the measured energies with theoretical responses of the channels to some given velocity. The theoretical responses were (pre)computed for every channel and for every reasonable velocity in the Fourier domain, basically, as a volume of intersection of the velocity induced motion plane and the blob representing the associated filter. The theoretical responses were functions of velocity. They represent what would be an ideal response of the given filter to a perfectly random plane translating with the examined velocity. So, for every pixel in the space-time image there is a collection of measured responses and many collections of ideal responses, each associated with certain velocity. The task is then to find the velocity whose collection best matches, in the least square sense, the measured collection. Clearly, the size of all collections is the same and it is exactly the number of the channels used. In other words, the method seeks optimal velocity by seeking a motion plane that best explains the measured responses. To allow for comparisons, any motion plane is "seen" only via the theoretical responses. The method seeks such responses that match the measured ones the closest.

[6]Parseval's theorem states that the integral of the squared values over the space-time image is proportional to the integral of squared components of its Fourier transform.

### The phase-based example

The phase-based approach, according to Fleet and Jepson [83], benefits from the observation that gradient of a local phase computed in every channel, i.e., in every complex filtering result, is in fact local instantaneous frequency present in the filtering result [134]. If the frequency is due to a pattern translating by $(u, v)$ pixels per frame, eq. (3.17) should hold and we may change it to obtain

$$(\phi'_x, \phi'_y, \phi'_t) \cdot (u, v, 1) = 0, \tag{3.27}$$

where $\phi'_x$ is derivative of phase in the direction of the $x$ axis, etc. Such equation is a variance on the constant brightness constraint given in eq. (3.8). Instead of following a contour of constant intensity in the space-time image, a contour of constant phase is followed in the (local) Fourier transform of the space-time image. This approach works with the motion-related property of "mutual perpendicularity" [22], eq. (3.17), and with the fact that amount of information is equally the same in the space-time image and in its Fourier transform. The phase is often more stable with respect to smooth constrast changes and near-identity affine deformations in the space-time image [83, 135, 136]. This was also accented in the comparison publications, e.g., in the one by Barron *et al.* [72], where the phase-based method was among the best performing methods in terms of accuracy.

Once the filtering is done, the method of Fleet and Jepson [83] computes for every pixel, i.e., for every spatio-temporal coordinate $(x, y, t)$, actually an estimate of a *component* velocity $\vec{v}_i$, rather than a final velocity $\vec{v} = (u, v)$. For the reasons explained earlier, a final velocity cannot be estimated directly from only a single channel. Hence, it is called component velocity as it estimates only a final velocity component in the direction of spatial phase gradient $\vec{n}_i$:

$$\vec{v}_i = \tilde{v}_i \vec{n}_i = \tilde{v}_i \frac{(\phi'_x, \phi'_y)}{|(\phi'_x, \phi'_y)|}. \tag{3.28}$$

Such component velocity is obtained for every pixel coordinate and every channel directly from the filtering result $R(x, y, t)$ using the equation $(\phi'_x, \phi'_y, \phi'_t) \cdot (\tilde{v}_i \vec{n}_i, 1) = 0$, a variant of eq. (3.27), and the following equation, as the authors suggest, for computation of the instantaneous local phase gradient:

$$(\phi'_x, \phi'_y, \phi'_t) = \frac{Im[R^*(x, y, t) \nabla R(x, y, t)]}{R^*(x, y, t) R(x, y, t)}. \tag{3.29}$$

The operation $Im[z]$ extracts imaginary part from a complex number $z$, the $R^*(x, y, t)$ is a complex conjugate of $R(x, y, t)$. Note that the numerator is actually a real vector of imaginary parts taken from elements of multiplication of a complex scalar, the filtering result, with a complex vector, the gradient of the filtering result. The denominator is a squared magnitude, a squared energy, of the filtering result.

The authors assume that the final velocity can be described as $(u, v) = (\alpha_0 + \alpha_1 x + \alpha_2 y, \beta_0 + \beta_1 x + \beta_2 y)$ where the six unknowns are gathered in a column vector $\boldsymbol{a} = (\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2)^T$ which is assumed to be constant within a small local region centred at coordinate $(x, y, t)$. For the projection of the final velocity to its component velocity it should hold

$$(u, v) \cdot \vec{v}_i = \tilde{v}_i. \tag{3.30}$$

Rewriting the left-hand-side of the equation and giving it the form of a multiplication of row and column six-element vectors, it is changed into a constraint for the final velocity:

$$\left[\frac{1}{|(\phi'_x, \phi'_y)|}(\phi'_x, \phi'_x x, \phi'_x y, \phi'_y, \phi'_y x, \phi'_y y)\right] \boldsymbol{a} = \tilde{v}_i. \tag{3.31}$$

Finally, for every pixel in the space-time image, the component estimates from all channels and tiny spatial vicinity of the examined pixel are further combined in an over-constrainted system of linear equations $\boldsymbol{Ra} = \boldsymbol{s}$ in six unknowns in $\boldsymbol{a}$, with rows of $\boldsymbol{R}$ given in eq. (3.31) and column vector $\boldsymbol{s}$ given with corresponding elements $\tilde{v}_i$. Least squares solution that minimizes $|\boldsymbol{Ra} - \boldsymbol{s}|^2$ defines the final velocity $(u, v) = (\alpha_0, \beta_0)$. Typically, not all computed component velocities are used to form the matrix $\boldsymbol{R}$. The authors employ a few sanity tests on the intermediate results to, possibly, discard some so that the final velocity is estimated only from reasonable and valid constraints. Sometimes, the tests do not make the computation of final velocity possible at all.

The approach by Fleet and Jepson [83] was influential. Not only they have shown that it was possible to achieve good results with the filtering-based approach to optical flow computation, they have also shown that other than intensity-based constraint can be used and that not all measured values must necessarily be always used. For instance, we will show in Section 4.3.4 that suppressing some intermediate results considerably improves accuracy of the method by Heeger.

### The other examples

It may be a coincidence, however, we see reflection of the above concepts in another popular approach by Joseph Weber and Jitendra Malik [104], published 5 years after Fleet and Jepson [83].

Weber and Malik used the differential approach, represented with eq. (3.8), to optical flow computation within the filtering-based framework. The authors used several, say $N$, differently bandpass filtered images $R_k(x, y, t)$, i.e., the channels, to obtain more independent constraints associated with every pixel so that its final velocity could be estimated. The authors also employed several additional constraints to reject some intermediate results before they would incorrectly influence the final velocity estimation. Hence, for every pixel individually only $N' \leq N$ constraints are used. The final velocity for every single pixel is estimated, again, from an over-constrainted linear system of the form:

$$\begin{pmatrix} R_{1x} & R_{1y} \\ R_{2x} & R_{2y} \\ \vdots & \vdots \\ R_{N'x} & R_{N'y} \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -R_{1t} \\ -R_{2t} \\ \vdots \\ -R_{N't} \end{pmatrix} \tag{3.32}$$

where, for instance, $R_{2x}$ is an estimate of derivative in the direction of the $x$ axis computed on the result of some 2nd filtering. Note that the estimation of directional derivatives can be realized with convolution. The filtering is, in the context of this work, a synonym for convolution. And since convolution is associative, we don't have to take convolution of the input image with some bandpass kernel followed by convolution that estimates directional derivatives. Instead, we may *precompute* convolution of the bandpass kernel

with the derivative estimator and apply the result, a derivative filter, on the input image afterwards. The input image is then convolved only once directly producing data for the linear system above.

Referring back to the reflections of Fleet and Jepson's work in this method, we notice the resemblence of eq. (3.32) with Fleet and Jepson's system $\boldsymbol{Ra} = \boldsymbol{s}$ when $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 0$ and when the phase gradients are replaced with prefiltered intensity gradients. The note on replacing the gradients supports the view that Weber and Malik were using a new constraint, namely the set of *prefiltered* intensity gradients in contrast to the ordinary intensity gradient $\nabla I$. In this view, they were following several contours of prefiltered constant intensity gradients simultaneously rather than contour of constant intensity or phase.

Weber and Malik [104] used the first and second order Guassian derivative kernels at different scales to obtain many bandpass versions of the space-time input image, many constraints in other words. But, as mentioned above, not all of them had to be accepted for further processing in the method. The linear system was real and solved with the total least squares technique. A very similar approach by Bruno and Pellerin [98] used the same scheme but they used recursive implementation of complex Gabor filters to feed the *complex* linear system. The velocity vector is real. The linear system, associated with every pixel, was changed to become real as well,

$$\begin{pmatrix} Re[R_{1x}] & Re[R_{1y}] \\ \vdots & \vdots \\ Re[R_{N'x}] & Re[R_{N'y}] \\ Im[R_{1x}] & Im[R_{1y}] \\ \vdots & \vdots \\ Im[R_{N'x}] & Im[R_{N'y}] \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} Re[-R_{1t}] \\ \vdots \\ Re[-R_{N't}] \\ Im[-R_{1t}] \\ \vdots \\ Im[-R_{N't}] \end{pmatrix}, \tag{3.33}$$

and solved using the $M$-estimators technique [137], which minimizes sum of functions of (not squared) residuals — unlike minimizing sum of squared residuals as in the least squares technique. The solver was designed to reduce the effect of outliers, which is an alternative to discarding unreliable constraints as in the previous approaches.

The last two approaches [104, 98], that incorporate the differential approach into the filtering-based framework, are perhaps closer to the energy-based methods [105]. It means that each of the two approaches should predominantly extract and deal with magnitudes (energies) from the filtered data in order to be considered an energy-based method. It is trivially accomplished if the filtering is real, as is the case of Weber and Malik [104]. If the filtering is complex, as is the case of Bruno and Pellerin [98], the situation is somewhat more complicated. Taking derivative of result of complex filtering changes both magnitude and phase of the result. In the same fashion, taking real and imaginary parts, eq. (3.33), cannot be in general regarded as extracting information solely from either magnitude or phase alone.

#### Frequency domain analysis under the constant brightness assumption

Let us consider rather similar parallel path to the latter approach of [98]. In this path, we continue working with complex numbers, i.e., the derivations of complex filtering results,

to supply them into the (this-time-complex) linear system in eq. (3.32). In this path, we also try to minimize sum of squared residuals $E(u, v)$ over all rows of the system. This is formalized, individually for every pixel in the input space-time image, using the modified but still apparent "classical" [19, 72] differential summation:

$$E(u, v) = \sum_{k=1}^{N'} |uR_{kx} + vR_{ky} + R_{kt}|^2. \tag{3.34}$$

Note again that the positional index is dropped from all terms of the equation and further.

According to the discussion in Section 4.1.3 on page 71, if we assume that the Gabor or Gabor-like filtering mimics or approximates to some extent the computation of true Fourier transform on appropriatelly cropped (small) and centred image, given in eq. (4.22) on page 73, then we may claim $R_k \equiv \mathcal{F}_I(\vec{w}_k)$ where $\vec{w}_k$ is the frequency tuning of the $k$th filter, $\vec{w}_k = (w_{k_x}, w_{k_y}, w_{k_t})$. The operator "$\equiv$" means "is proportional". Since the $\mathcal{F}_I$ is a regular Fourier transform applied only on small image, all properties of Fourier transform hold. Namely, we will use the derivative theorem

$$\mathcal{F}_{I_x}(\omega_x, \omega_y, \omega_t) = i\omega_x \mathcal{F}_I(\omega_x, \omega_y, \omega_t) \tag{3.35}$$

where $I_x$ is the derivative of some image $I$ taken in the direction of the $x$ axis and $i^2 = -1$. Following the earlier discussion on the convolution operations used in the approach, the order of convolutions with the $k$th filter and with the derivative operator can be swapped owing to the commutativity property of convolution. Thus, we may assume that the value $R_{kx}$ is actually a value after filtering the derived image and so it is "proportional" to $\mathcal{F}_{I_x}(\vec{w}_k)$.

We rewrite eq. (3.34) according to [105]:

$$\begin{aligned} E(u, v) &\equiv \sum_{k=1}^{N'} \left| u\mathcal{F}_{I_x}(\vec{w}_k) + v\mathcal{F}_{I_y}(\vec{w}_k) + \mathcal{F}_{I_t}(\vec{w}_k) \right|^2 && (3.36) \\ &\equiv \sum_{k=1}^{N'} \left| u\, iw_{k_x}\mathcal{F}_I(\vec{w}_k) + v\, iw_{k_y}\mathcal{F}_I(\vec{w}_k) + iw_{k_t}\mathcal{F}_I(\vec{w}_k) \right|^2 && (3.37) \\ &\equiv \sum_{k=1}^{N'} \left[ (u, v, 1) \cdot \vec{w}_k \right]^2 \left| \mathcal{F}_I(\vec{w}_k) \right|^2. && (3.38) \end{aligned}$$

Note that the term in square brackets is actually eq. (3.17) on page 37, which, sort of, measures the appropriacy of the associated motion plane, which is given by the examined velocity $(u, v)$, with given coordinates both in the Fourier domain. The smaller the value is, the better fits the coordinates onto the motion plane. Value of zero indicates perfect fit. The summation in eq. (3.38) should be understood as weighted summation of the appropriacy measurements weighted with the measured energies after the filtering. The greatest contributors to the sum are filters that are tuned to frequency components where the energy (or power) in the space-time image is concentrated, i.e., where the sought motion plane is expected to form — of course, under the condition of ideally translating pattern in the whole small vicinity of the examined pixel. The task is to find velocity that

induces such motion plane that is the most appropriate to these measurements. The first term is then close to zero and the sum is kept low in this way.

Such approach, in fact, detects appropriate motion plane via some observed energies. This is why we regard it to belong among the energy-based approaches. And this is also why we regard the approach by Bruno and Pellerin [98], though this analysis was only a parallel to them, to belong more or less among the energy-based optical flow computation methods.

# Chapter 4

# Spatial filtering for optical flow computation

In this chapter we focus on filtering in the spatial domain and provide both theoretical and practical background on how to approach it. We will always bear in mind the context of the filtering, which is the motion detection in sequences of 2D images. The images in a sequence are stacked together so that a, so called, space-time (2D+t) image is created. However, technically it is always only an (3D) image and, thus, the filtering happens solely in the spatial domain. The filtering is hoped to mimic processes in the very early stages of the human visual system, at least we are going to reproduce the shape of optimal filter from it, the Gabor filter. We are heading towards the energy-based optical flow computation methods. But the filtering can be used also for the phase-based methods. We are, therefore, interested in filtering with complex filters as they are a nice special case of quadrature filter pairs, which we need for the two filtering-based methods.

## 4.1 Spatial filtering

### 4.1.1 1D Filtering

Filtering along a line, that's what a 1D filtering is, is the most basic situation. From the discussion above, we realize that 1D filtering is useless for motion estimation, for which at least two dimensions must be present. However, we shall see in next sections that any higher dimensional filtering with a separable filter can be implemented in the image domain by means of a cascade of several pieces of 1D filtering. And since we shall deal with separable filters, it is important for us to find a reasonable 1D spatial filtering implementation. The filtering or the filters used must allow for both fast and accurate computation.

We will turn our attention to the filtering with Gaussian filter, although we want primarily to efficiently filter with Gabor filter, which is essentially different from Gaussian. We will see in the next section that filtering with Gabor filter can be interchanged with filtering with Gaussian. Probably, this also the reason why many researchers focus on fast implementation of Gaussian and very occasionaly of Gabor filtering.

Despite the Gaussian filter is well-known, we define it here, at least to introduce

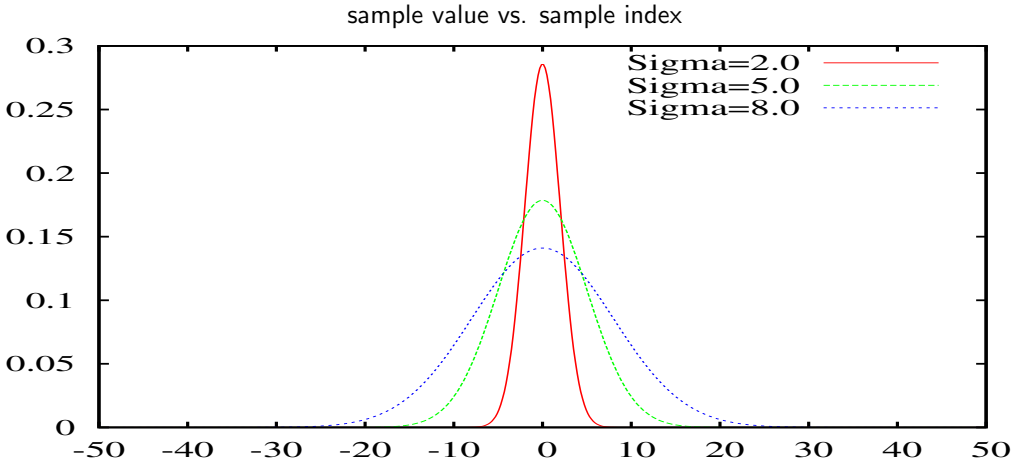


Figure 4.1: Examples of the Gaussian filter with different parameter $\sigma$. Note where each filter is attenuated.

notation. The 1D Gaussian filter is given as

$$\mathcal{G}a(x,\sigma) = \frac{1}{(2\pi)^{1/2}\sigma}\, e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}. \tag{4.1}$$

The constant fraction serves the purpose of normalization factor. The real exponential defines the shape of the filter and its properties as well. Only positive values of $\sigma$ are considered. Note that the filter is symmetric with respect to $x = 0$. Also note that both filter tails, $x \to \pm\infty$, approach zero in the limit. Moreover, they fall to zero approximately from the distance of $\pm 3\sigma$. In other words, the filter's support (or span) is often considered to be only $6\sigma$, i.e., $\mathcal{G}a(x,\sigma) \approx 0 \iff |x| > 3\sigma$, see Fig. 4.1. This is an important point for the implementation.

The convolution with this filter, the Gaussian filtering, is a collection of results $O(y)$ obtained with the following equation:

$$O(y) = \int_{-\infty}^{\infty} I(x) \cdot \mathcal{G}a(y-x,\sigma)dx. \tag{4.2}$$

The $I(x)$ is a value of pixel at $x$ in an input image, $O(y)$ is a value at $y$ in an output image. We often digitize the Gaussian filter as follows:

$$Ga(x,\sigma) = \begin{cases} \frac{1}{N}\, e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} & \text{if } x \in \langle -s, s\rangle \cap \mathbb{Z}, \\ 0 & \text{if } x \in \mathbb{Z} \setminus \langle -s, s\rangle, \end{cases} \tag{4.3}$$

$$s = \lceil 3\sigma \rceil, \tag{4.4}$$

$$N = \sum_x e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \quad \text{over } x \in \langle -s, s\rangle \cap \mathbb{Z}. \tag{4.5}$$

The $\lceil q \rceil$ is the lowest integer not smaller than $q$, the ceiling. The $\mathbb{Z}$ is the set of all integer numbers including zero. The filter's support is always $2n+1$. We often refer to the $Ga(x,\sigma)$

as to the convolution kernel. Note that the kernel is mirrored during convolution. The eq. (4.2) is then digitized to:

$$O(y) = \sum_{x=-\infty}^{\infty} I(x) \cdot Ga(y - x, \sigma). \tag{4.6}$$

The operation of convolution is often noted with $*$, e.g., $O = I * Ga(\sigma)$.

For the implementation in computers we make use of the comutativity of the convolution, $I * Ga(\sigma) = Ga(\sigma) * I$, refer to [138, 139] also for other properties of convolution. The digitized formula then becomes:

$$O(y) = \sum_{x=-s}^{s} I(y - x) \cdot Ga(x, \sigma). \tag{4.7}$$

Note that bounds of the sum were changed as a result of eq. (4.3). In order to filter at some position $y$, the program must sweep input data in the vicinity of $y$ and, during the sweep, element-wise multiply with the Gaussian $Ga(x, \sigma)$. It is preferable to use filters with small $\sigma$ because it results in small $s$, which, in turn, allows for better utilization of processor caches. Small $\sigma$ is preferrable also in terms of time complexity as we shall see in the next section. Such convolution is often called the naive, or sometimes the plain or the full, convolution. Since the filter has limited support and works only with limited number of input data values, this filtering is also denoted as the Finite Impulse Response filtering, the FIR filtering [113, 114].

We will often need to conduct the 1D Gaussian filtering in a higher-dimensional image, e.g., in 2D or 3D. In the case of a general $n$D image, the convolution may be demanded to run along arbitrary direction (column) vector, say $\vec{b} = (a_1, \ldots, a_n)^T$. In this case, eq. (4.7) is changed to:

$$O(\vec{y}) = \sum_{x=-s}^{s} I\left(\vec{y} - x\vec{b}\right) \cdot Ga(x, \frac{\sigma}{\Delta L}) \tag{4.8}$$

where $\Delta L = |\vec{b}| = \sqrt{\sum_{i=1}^{n} a_i^2}$ and $s$ is updated for the new Gaussian's parameter $\sigma/\Delta L$. For this moment, we will assume all $a_i \in \mathbb{Z}$, i.e., the $\vec{b}$ is an *integer* vector. Since the input image is at least 2D, many such convolutions must be computed in order to fully convolve the image, Fig. 4.2.

Let us make a short comment on the form of eq. (4.8). Alternatively, we may change the kernel to $G(x\Delta L, \sigma)$ to obtain a new expression that leads to exactly the same results. But it is less efficient since the original "full-size" $s$ is kept. Whenever it holds $|x| > \lceil s/\Delta L \rceil$, the kernel values drop to zero and the computation of convolution becomes only a waste of time. Another possibility would be to use the "full" Gaussian $G(x, \sigma)$ in the equation and rather normalize the $x\vec{b}$ by using the $x\vec{b}/\Delta L$ instead. This means that the direction vector $\vec{b}$ degrades in its functionality as it would only tell an orientation, azimuth in 2D, without the ability to express the magnitude, the size of convolution step. We have opted for the more general solution, the one given by the equation.

### 4.1.2 Recursive 1D filtering

Opposed to the FIR filtering, we may consider the implementation of Gaussian filtering by means of the Infinite Impulse Response filtering, the IIR filtering [113, 114]. The

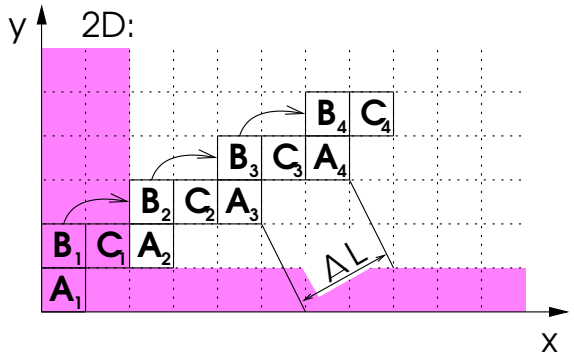


Figure 4.2: Illustration of how 1D convolutions must be repetatively applied to complete the convolution of the whole input image in the given direction. Three runs along the direction vector $\vec{b} = (2, 1)$ are shown. They are marked as A, B and C, respectively. In fact, it suffices to start any run only from the pixels in the magenta (gray in B&W print) area. Note that the width, i.e., the dimension in the $x$ axis, of the vertical magenta stripe is exactly 2. Similarly, the height, which is the dimension in the $y$ axis, of the horizontal magenta stripe is exactly 1. It will always be the case that the stripes and their dimensions will be given with the direction vector $\vec{b}$.

fundamental difference between the two is that the IIR filter utilizes recursion in the process of output value computation. The filter simply considers not only values from an input image, it also considers return values from any previous computation(s), in addition. As a result, the filter shape, i.e., the impulse response, is somewhat harder to see directly from the filter's coefficients. This is different to the FIR where the impulse response is merely a mirror of the kernel. Mentioning the kernel, in the FIR filtering the kernel size was dependent on filter's parameter, recall eq. (4.4). In particular, the greater the Gaussian's $\sigma$ is, the larger the kernel is. In the IIR filtering, we will see that the size of "kernel" does not depend on the value of $\sigma$. This is the most exciting feature about the IIR. The term recursive filtering is often used as an alias to the IIR filtering.

To the best of our knowledge, we recognize only three approaches to recursive Gaussian filtering published so far in the literature. This is the work done by Deriche [140], by Young *et al.* [126] (improved later by van Vliet *et al.* [141]) and finaly by Jin and Gao [142]. All of these are approximations to the FIR filter. Because each focuses on different criteria when seeking their approximate solution, differences in performace have arisen. We refer an interested reader to the original literature for particular details on recursive Gaussian (and its derivatives) filtering [140, 126, 141, 142] and later then on recursive Gabor filtering [143]. Fortunatelly, a review and comparison publication at the same time[1] on the three approaches [140, 126, 142] exists due to Tan *et al.* [144]. We allow ourselves to reproduce their Table 1 and Table 2 ([144], p. 225) for convenience, see Table 4.1 and Table 4.2,

[1] The authors were actually looking for fast implementation of position dependent Gaussian filters so that they could implement foveation, a process of human vision that blurs peripheral regions more intensively. They had to compare existing approaches for accuracy, speed and extensibility to their demands. Their work has become a, de facto, standard reference for the comparison on accuracy and speed of the three recursive approaches.

| | $\sigma = 1$ | $\sigma = 2$ | $\sigma = 3$ | $\sigma = 4$ |
|---|---|---|---|---|
| Deriche [140] | 2.0 | 0.84 | 0.85 | 0.79 |
| van Vliet *et al.* [141] | 19 | 7.3 | 6.0 | 5.5 |
| Jin *et al.* [142] | 0 | 18 | 41 | 56 |

Table 4.1: Reproduced Table 1 of [144]: Normalized RMS error (in %) of the three filters for a 2D impulse response.

| | $\sigma = 1$ | $\sigma = 2$ | $\sigma = 3$ | $\sigma = 4$ |
|---|---|---|---|---|
| van Vliet *et al.* [141] | 0.93 | 0.18 | 0.10 | 0.082 |
| Deriche [140] | 0.61 | 0.48 | 0.36 | 0.30 |
| Jin *et al.* [142] | 0 | 4.7 | 12 | 18 |

Table 4.2: Reproduced Table 2 of [144]: Normalized RMS error (in %) of the three filters for responses measured on test image filled with uniform random noise.

respectively. The normalized RMS used in both tables is given as

$$\frac{1}{N_D} \sum_{(i,j) \in D} \frac{\sqrt{(O_T(i,j) - O_R(i,j))^2}}{O_R(i,j)} \tag{4.9}$$

$O_T$ is a tested filtering result of an examined method, $O_R$ is a reference filtering result obtained either from an analytic formula for an expected impulse response (Table 4.1) or from FIR filtering with great support (Table 4.2), $D$ represents domain of coordinates within both images and $N_D$ is the number of such coordinates.

Based on their results we focus only on the recursive filters by Deriche and by Young *et al.* The third approach showed worse performance (almost by one grade, we would dare to say) in their tests while the other two were rather balanced if not almost equal. Strictly speaking, the Deriche's approach gave the best results in the test with impulse response while Young *et al.* was the best on randomly filled image. As Tan *et al.* pragmatically notes: it is the test with image rather than with single point impulse that is of practical importance. Anyway, there are further indicating clues to consider.

In order to continue with the discussion to finally select "the best" approach, we need now to define the two preselected. Both approaches (in fact, all four including [142, 141]) tackle the recursive filtering by means of the forward and backward *passes* or sub-filtering. To conduct 1D filtering, one has to simply convolve in two 1D passes. Using the notation $O^+$, $O^-$ and $T$ for auxiliary images and $n_i^+$, $n_i^-$, $d_i^+$, $d_i^-$ and $b_i$ for filter coefficients, the passes for Deriche's approach [140] are defined as

$$\begin{aligned}
\text{forward: } O^+(x) &= n_0^+ I(x) + n_1^+ I(x-1) + n_2^+ I(x-2) \\
&\quad - d_1^+ O^+(x-1) - d_2^+ O^+(x-2) - d_3^+ O^+(x-3), && (4.10) \\
\text{backward: } O^-(x) &= n_1^- I(x+1) + n_2^- I(x+2) + n_3^- I(x+3) \\
&\quad - d_1^- O^-(x+1) - d_2^- O^-(x+2) - d_3^- O^-(x+3), && (4.11) \\
\text{addition: } O(x) &= O^+(x) + O^-(x) && (4.12)
\end{aligned}$$

while the passes for Young's *et al.* approach [143] are defined as

$$\begin{aligned} \text{forward: } T(x) &= I(x) - b_1 T(x-1) - b_2 T(x-2) - b_3 T(x-3), && (4.13) \\ \text{backward: } O(x) &= B \cdot T(x) - b_1 O(x+1) - b_2 O(x+2) - b_3 O(x+3). && (4.14) \end{aligned}$$

Notice the presence of recursivity in all passes. For the later approach, we also note here that there are actually three variants published [126, 141, 143]. We consider here only the latest version [143], which is for any Gaussian filter the same as the oldest version [126] and which differs slightly in filter coefficients $b_{1,2,3}$ from the "middle" version [141]. Anyway, this version is the most efficient one (we'll cover that a bit latter).

There is a subtle difference in how both passes are employed in the approaches, Fig. 4.3A,B. Deriche (and Jin *et al.* as well) requires to conduct forward and backward passes on the given input image $I(x)$. Afterwards, they obtain two result images, namely $O^+(x)$ and $O^-(x)$, which they pixel-wise add to yield result of the filtering. Young *et al.* requires the backward pass to run on a result of the forward pass $T(x)$, on the other hand. Clearly, considering computation of a single 1D Gaussian filtering, the Deriche's approach requires two intermediate auxiliary image buffers while the other only one such. However, we may actually easily modify the backward pass of the Deriche's approach,

$$\begin{aligned} \text{backward: } O(x) &= O^+(x) + \quad n_1^- I(x+1) + n_2^- I(x+2) + n_3^- I(x+3) \\ &\quad -d_1^- O^-(x+1) - d_2^- O^-(x+2) - d_3^- O^-(x+3), && (4.15) \end{aligned}$$

such that it incorporates the addition of result of the forward pass to obtain the final filtering result directly from the backward pass. Note that we can do the same thing with the forward pass as well, but not with both of them at the same time. The approach got strictly serialized, exactly in the same way the other approach is. This equals both approaches in terms of the number of auxiliary image buffers and the number of processing steps required in total, Fig. 4.3B,C.

It is noteworthy that the design of Young's *et al.* passes permits to compute them, so called, in-place. The Young's *et al.* approach, unlike the other, can be easily implemented without any temporary auxilliarily image buffer(s). In particullar, the forward pass may be computed directly to the $O(x)$ image because the backward pass does not require value from any position of its input image (originally the $T(x)$) other than the one it currently modifies. To be able to use, say, $O^+(x)$ instead of $I(x)$ as we would do when attempting to process the Deriche's forward pass in-place, we would require either $n_{1,2}^+ = 0$ or $d_{1,2,3}^+ = 0$ because we can't read the original (pre-filtering) value and the modified (post-filtering) value at the same time from the exactly the same memory positions $x-1$ and $x-2$. The Young's *et al.* filtering can be more memory efficient, again.

Moreover, nowadays when parallel computing is available even in every notebook computer, we may seriously consider simultaneous processing of two (or even more) convolutions. Since we aim at Gabor bank filtering, in which each Gabor filter is implemented by means of modulation, Gaussian filtering and demodulation, we basically have to deal with many pieces of Gaussian filtering. Considering parallel implementation only for pairs of Gaussians, the Young's *et al.* approach is favourable even in this respect. The Deriche's passes can run in parallel, see Fig. 4.3D. But it has potential performance bottleneck in the synchronization before the addition. Furthermore, two filters at two processors need

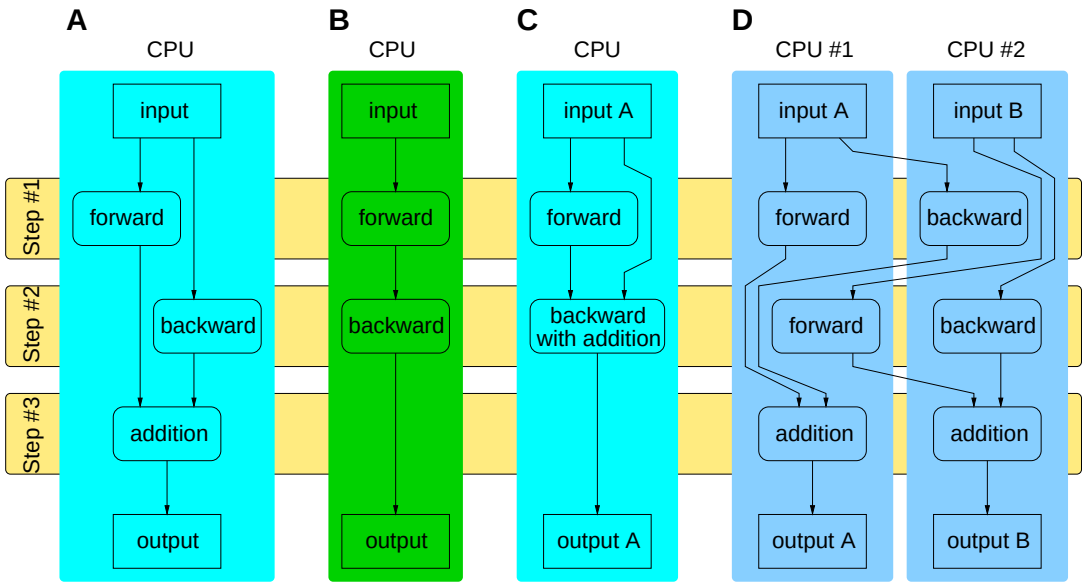


Figure 4.3: Work flows of the considered recursive IIR 1D filtering approaches. The basic and the more efficient variants of Deriche's approach are given in A and C, respectively. The modified variant strongly resembles the approach of Young *et al.* , in B. An illustration of a parallel implementation of the Deriche's approach, in D. Actually, more variants are possible for this case. However, as long as the "basic" version is considered, the work flow will require at least three processing time slots (shown with the horizontal stripes).

still one processing unit of time more when compared to two independently parallel runs of the strictly serialized filtering.

Since convolution with the recursive filters achieves linear asymptotic time complexity in size of input image, we must distinguish between the approaches by considering a finer grain measure. Indeed, any of the forward or backward passes requires only a constant, i.e., trivially upper-bounded, number of neighboring pixels for the computation of any output value. They differ only in the, so called, implementation constant, which is proportional to the constant number of considered neighboring pixels. In accordance with literature, e.g., with [144, 145, 146], we compare with the number of operations per pixel, ops/px, required for an approach to complete the filtering. We count both single real multiplication and single real addition as one operation each. Complex multiplication then results in 6 operations. In the view of this measure, the Deriche [140] requires 6 real multiplications plus 5 real additions for the forward pass, altogether 23ops/px for the whole 1D Gaussian filtering. Young *et al.* [143] needs only 13ops/px. Note that the other recursive forms of Young *et al.* filtering require 14ops/px. While it is only a single redundant operation per pixel, it must be stressed that it is per *every* pixel accessed during convolutions of the whole spatio-temporal stack with the few filter banks in which every filter is separated into several 1D filters. Every small inefficiency in the 1D filtering gets multiplied in this way. This single operation more causes an unnecessary lag of nearly 8% ops/px of the total demand, i.e., roughly about 8% of increase in computational time.

**Conclusions on 1D recursive filtering**

To sum it up, appart from less ops/px hand in hand with better memory utilization, the Young's *et al.* approach was slightly more accurate on a given test image [144]. We have also realized that it is directly suitable for parallelism-enabled computing environments. Each parallel run works only with its local memory completely independently on any other run. These are the reasons we have decided to use the 1D recursive filtering whenever we would need 1D Gaussian or Gabor filtering in any direction, eq. (4.8), within any $n$D image.

We have opted for the recursive filtering for the two reasons: the number of ops/px is small and does not depend on the value of Gaussian's $\sigma$. Indeed, all the above mentioned approaches have their formulae for the forward and backward passes firmly given with variability only in the filter coefficients not in the filter's support. The $\sigma$ is encoded in the coefficients. In their publication, Young *et al.* [143] adds to it that the accuracy of the filtering improves with increasing $\sigma$. Note that this is in accordance with our needs because we aim to obtain highly selective filters in the Fourier domain, i.e., filters with large $\sigma$ in the image (spatial) domain and so more localized in the Fourier domain.

There is even more to it when regarding the recursive filters based on Young *et al.* [126] and its successors [141, 143], a correct and fast boundary treatment has been worked out quite recently [147, 145] and [P3], see Table 4.3. In particular, Triggs and Sdika [147] have found an universal solution for Gaussian IIR filtering based on eqs. (4.13),(4.14), i.e., the Young *et al.* family of recursive filters such as [126, 141, 143]. Their solution works also for (direct) complex recursive Gabor filtering, though they haven't provided it explicitly. For the Gabor filtering computed with Gaussian filtering, i.e., for Gaussian filtering in the context of Gabor filtering, we knew only one solution published by Bernardino and Santos-Victor [145]. Their solution is capable of working for all Gaussian filters based on Young *et al.* family [P3]. But to use it, we must provide filter coefficients *and* filter poles of filter Z-transform. The requirement for poles is, however, an unnecessary barrier in the use of the solution [P3] and limits its applicability, in practice, only to Gaussian IIR filters defined in [141] where the poles are directly available. Otherwise, an automatic 3rd order polynomial root finder must be devised to obtain poles from filter coefficients automatically. A correct boundary treatment based solely on filter coefficients and so for any general IIR Gaussian in the context of Gabor filtering provided the Gaussian belongs to the Young *et al.* family, i.e., can be defined with eqs. (4.13),(4.14), is shown in [P3].

Finally, to demonstrate performance of the implemented underlying 1D recursive Gaussian filtering based on the latest approach of Young *et al.* [143], we include the following three figures: comparison of impulse responses in the 1D and 2D case in Fig. 4.4 and Fig. 4.5, respectively, and comparison of time consumption in Fig. 4.6.

### 4.1.3 2D Filtering

Once we enter into higher dimensions, starting already with 2D, we immediately face the fundamental problem with Gabor filters: they are not separable. This is only a technical problem but very important for anyone who requires fast computation of the Gabor filtering. In fact, we are left with the naive 2D convolution or with the convolution theorem together with the (fast) Fourier transforms back and forth. The former solution is, at least, problematic when filter kernels grow in size. We see from eq. (4.7) that the

| IIR machinery | general purpose Gaussian filtering | (direct) recursive Gabor filtering | Gaussian in the context of Gabor filtering |
|---|---|---|---|
| Young *et al.* [126] | Triggs and Sdika [147] | Triggs and Sdika [147] | Ulman [P3] |
| van Vliet *et al.* [141] | Triggs and Sdika [147] | Triggs and Sdika [147] | Bernardino and Santos-Victor [145] Ulman [P3] |
| Young *et al.* [143] | Triggs and Sdika [147] | Triggs and Sdika [147] | Ulman [P3] |

Table 4.3: An overview of the state-of-the-art solutions to correct 1D boundary treatment in recursive Gaussian and recursive Gabor filtering applications. The solution by Bernardino and Santos-Victor [145] can be, actually, applied also on the Young's *et al.* filters [126, 143] in the rightmost column of the table provided the solution is extended with automatic 3rd order polynomial root finder. Details on this matter are given in the text and in Section IV-D in [P3].

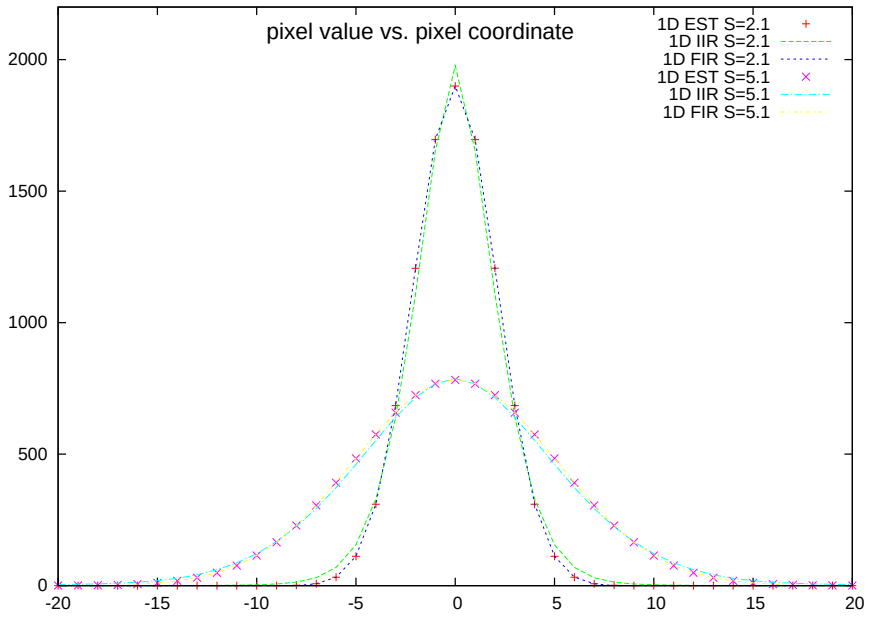


Figure 4.4: Examples of impulse responses of 1D FIR and IIR Gaussian filtering for two $\sigma$ denoted as "S". The "EST" denotes an analytic curve of what should be the correct impulse response. Note that for smaller $\sigma$ there is an apparent small error in the regions of $\pm 6$.

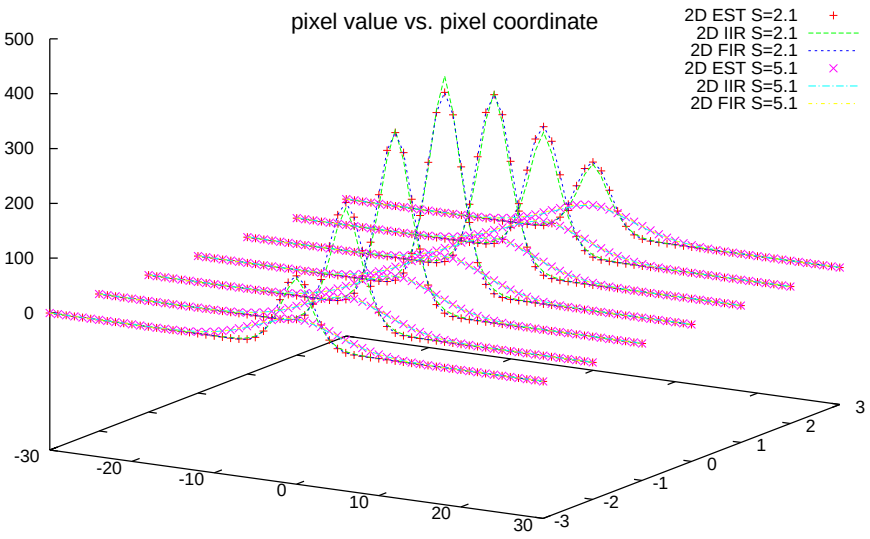


Figure 4.5: Examples of impulse responses of 2D FIR and IIR Gaussian filtering for two $\sigma$ denoted as "S". The "EST" denotes an analytic curve of what should be the correct impulse response. Note that for smaller $\sigma$ there is an apparent small error.

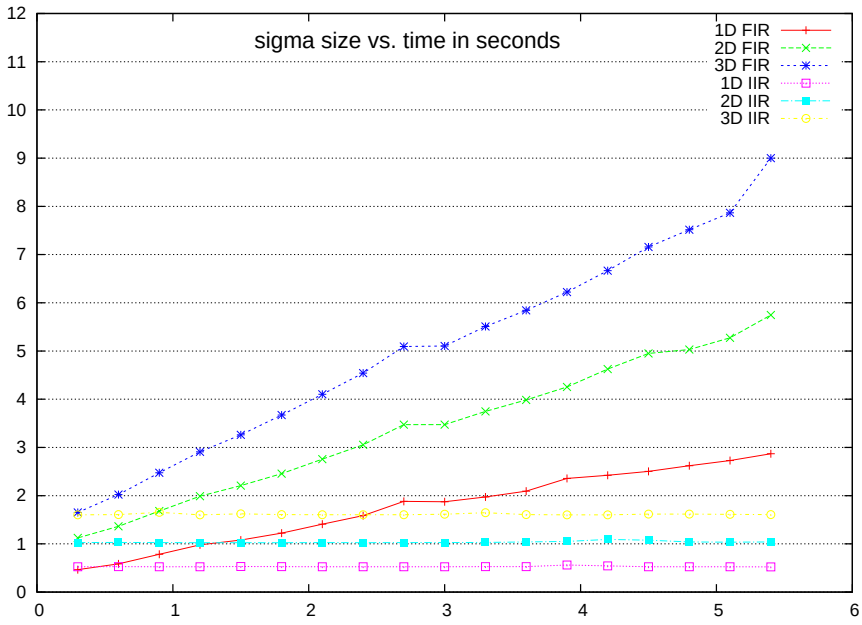


Figure 4.6: Examples of time spent with the several computations. Separable filters were used in the 2D and 3D filtering. We clearly see that the IIR filter has constant time consumption. As the filtering happens in more dimensions, from 1D to 3D, we see that its time consumption *regularly* increases suggesting that individual 1D recursive filtering implementations along the axes $x$, $y$ and $z$, respectively, shows roughly the same time consumption. The FIR filtering also shows linear dependence of the computation time on size of $\sigma$. The slope changes with dimension of filtering.

naive convolution routine, the FIR, has time complexity $\mathcal{O}(nm)$ with $n$ being the number of convolved pixels and $m$ being the number of filter coefficients. Considering convolution of a square image $n \times n$ with a square filter of kernel size $m \times m$, the complexity also enters higher dimension: $\mathcal{O}(n^2m^2)$. If the filter were separable, we would have to compute only a small number of 1D convolutions (but within a 2D image) and the (asymptotic) complexity would slightly improve to $\mathcal{O}(n^2m)$. Regarding the Fourier transform, we will cover the use of it for the filtering in Section 4.1.5. For this moment let us foretell that its time complexity is $\mathcal{O}(n^2 \log n)$ for $n \times n$ image. Comparing the two approaches, the Fourier transform is preferable when the kernels are large, i.e., when $m \gg \log n$. It is important to keep the time complexity low, naturaly. In the optical flow computation based on filtering, we need to compute many pieces of filtering, even tens of. If a single filtering would take more time, the computation of a whole bunch may not be tractable.

Fortunatelly, the convolution with a 2D, or higher dimensional, Gabor filter can be broken down into the three stages. In the following text we will always omit the constant fraction from the Gabor filter expression. The equation for convolution with a general 2D Gabor filter is

$$O(\vec{y}) = \sum_{\vec{x}} I(\vec{x}) \cdot e^{-\frac{1}{2}(\vec{y}-\vec{x})^T C^{-1}(\vec{y}-\vec{x})} \; e^{iW(\vec{y}-\vec{x})} \tag{4.16}$$

where $I(\vec{x})$ is input image real pixel value at coordinate (column) vector $\vec{x}$, $O(\vec{y})$ is output image complex pixel value, $C$ is a $2 \times 2$ Gaussian covariance matrix and $W$ is a $1 \times 2$ single row matrix with Gabor frequency tuning, $W = [2\pi w_x, 2\pi w_y]$. The equation can be rewritten:

$$O(\vec{y}) = e^{iW\vec{y}} \cdot \left[ \sum_{\vec{x}} \left[ I(\vec{x}) e^{-iW\vec{x}} \right] \cdot e^{-\frac{1}{2}(\vec{y}-\vec{x})^T C^{-1}(\vec{y}-\vec{x})} \right]. \tag{4.17}$$

The inner term in square brackets is a modulation with complex exponential $e^{-iW\vec{x}}$, the first stage. It results in a complex data that is fed into the middle stage, which is the Gaussian filtering represented with the outer square brackets. Finally, the complex filtering result is demodulated with $e^{iW\vec{y}}$, the last stage. Both modulation and demodulation operations are nothing but simple pixel-wise multiplication with position-dependent constant. Since Gaussian filter is a real domain filter, it sufficces to convolve with it both the real and imaginary parts of the modulated input independently of each other. In the thesis, we will refer to this approach as to the *staged* approach.

The greatest advantage of the staged approach is that the complex convolution with Gabor filter was replaced with two real convolutions with Gaussian filter. Clearly, real arithmetics require less operations, multiplications and/or additions, than the complex one. For instance, consider complex multiplication that consists of four real multiplications and two real additions on top of it. The greatest advantage, however, is the shift from filtering with non-separable Gabor filter to filtering with its Gaussian envelope. Note that Gaussian filter *is* separable, even when arbitrary configuration ($C$ is *not* diagonal) is required [30, 33]. Depending on what filters we use for the 1D Gaussian convolutions, different performance gain is achieved compared to the naive 2D Gabor convolution.

The separability allows for faster computation of the plain 2D convolution, eq. (4.16), with only two or three 1D convolutions. To ease the comprehension of the following, we will assume the Cartesian coordinate system is used. In the classical case when $C$ is diagonal, the Gaussian is easily separated along coordinate system axes, i.e., along the $x$ and $y$ axes.

When the desired Gaussian envelope results in a general $C$, we may decompose [30] into a convolution along the $x$ axis and another convolution along a direction vector $(x_1, 1)^T$ with $x_1 \in \mathbb{R}$. This keeps the number of convolutions lowest possible [32] at the expense of allowing the $x_1$ to take a real value. The convolution runs off a pixel grid whenever $x_1 \notin \mathbb{Z}$ because pixels in the image are spread on a grid only at integer coordinates. Some interpolation technique in the $x$ axis must be used to, firstly, obtain values off the grid so that convolution can be computed and, secondly, from these values reconstruct values at the grid. Recently, an approach appeared [33] that uses three 1D convolutions with direction vectors based solely on integer values. Such direction vectors, that can't push the convolution to fall off the pixel grid, elegantly canceled the need for interpolation as well as an artifact connected with it [33] and [P1]. On the other hand, the third 1D convolution introduced some additional ops/px outweighting the few saved ops/px originally required by the computation of interpolation.

Regarding the use of interpolation, Lam and Shi [33] pointed out that the interpolation in [30] introduces spatial inhomogeneity, i.e., the responses to the same impulses but at different image locations $\vec{x}_i$ are not identical when shifted back by the vector $-\vec{x}_i$, see Fig. 4.7. After the shift, impulse responses appear registered. They should be the same but, in fact, they are slightly differing. This artifact is also called the positional variability. They demonstrated the variability for the 2D case in their publication. Note that it is a property of the used error measure that it readily tells an *offset* to the order of measured error with respect to leading order of output data [33] and [P1]. They have found out that the variability manifests at offset of 4 orders. We have managed to reproduce their results with our implementation. Using this testbed, we have extended the study to the 3D case and published it in our original publication [P1]. The 3D filters show positional variability as well but at about two orders of magnitude higher (offset is only 2 orders) compared to 2D, probably because more interpolations must be used in 3D. We have extened according to the Lampert and Wirjadi [32] who have published a general solution for $n$D with the property that for $n = 2$ it turns into the one by [30] — the approach "criticised" by Lam and Shi [33] for the positional variability in 2D. Concluding this topic, we show in Fig. 4.8 results of a test on some real data to see if the discussion is not only at some theoretical level because the aforementioned measurements were conducted on impulse responses. Unfortunately, it seems that at least in this particular test with the 3D image the error is also present. Fortunately, we may use the other approach [33] and [P1] that does not use interpolations but is often slightly more demanding in terms of required operations per pixel.

### Gabor filter as a local Fourier transform

We first, for convenience, give the definition of the Fourier transform, given in eq. (3.12) on page 37, for a 2D image:

$$\mathcal{F}_I(\omega_x, \omega_y) = \iint_{-\infty}^{\infty} I(x, y)\ e^{-i2\pi(\omega_x x + \omega_y y)} dxdy. \tag{4.18}$$

Comparing the staged approach, given in eq. (4.17), $\vec{x} = (x, y)^T$, with this definition of the Fourier transform, we notice that the integrated function in the Fourier transform is essentially the modulation term in the first stage of the staged approach. But the

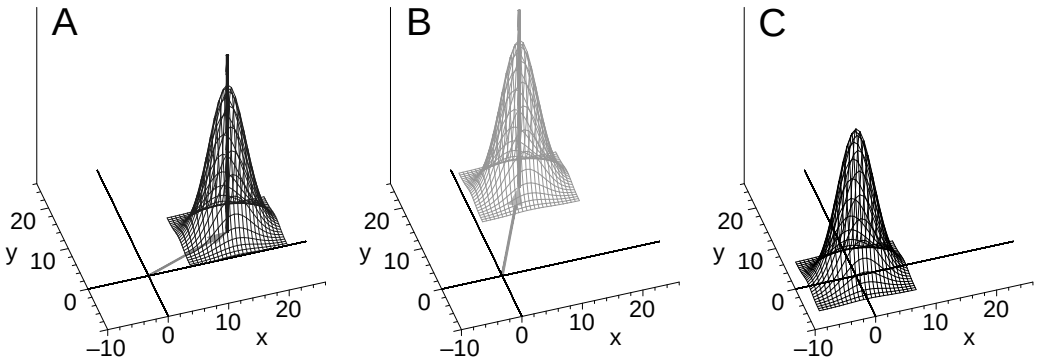


Figure 4.7: Illustration of how data is prepared for measurement of the positional variability. A sample single point impulse of constant height is placed at different positions in the image, as in A and B. Note that positions within image are given with integer coordinates. The Gaussian filtering with the same constant filter is conducted, sample results are shown in A and B over the impulses. The results are then translated to the same place, say to the centre of coordinates as in C. The variability is a logarithm of variances over all positions.

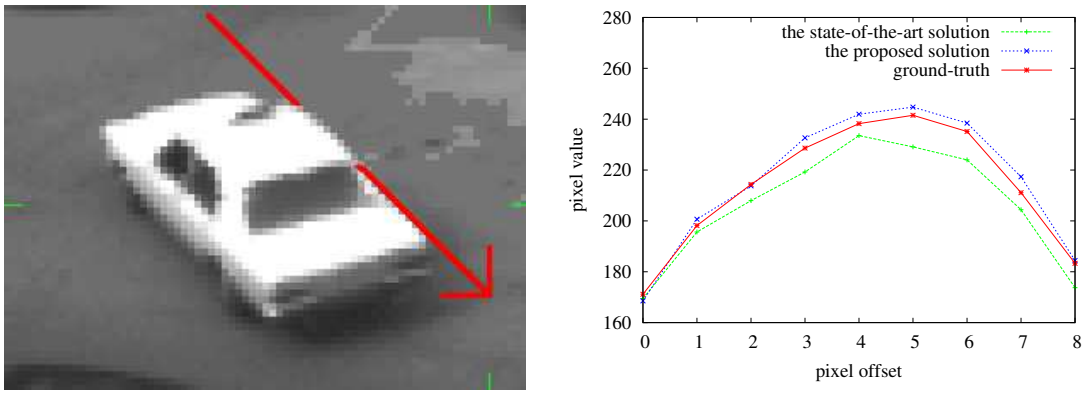


Figure 4.8: Comparison of Gaussian filtering results that we obtained on a frame, in the left, from the well-known Hamburg taxi sequence. Technically, it is a 2D slice from a 3D stack on which we computed several 3D filtering with the same Gaussian filter. The results of filtering, gathered at positions along the red line in the left image but only in its missing part, are plotted in the right. The plot compares results of naive convolution, denoted as "ground-truth", with results of the state-of-the-art method by Lampert and Wirjadi [32] and with results of the method proposed in this thesis [P1]. We see that the former method produces regurarly occuring error suggesting that the error rate depends on position. Also note that the magnitude of error is occasionally in the order of tens while the magnitude of results is in the order of hundreds, i.e., only one order of magnitude higher.

integration over the whole image is replaced with Gaussian convolution in eq. (4.17). This is, however, only a Gaussian weighted summation over local region around every pixel $\vec{y} = (x_0, y_0)^T$ in the (space-time) image, which is illustrated for 1D in eq. (4.2). The localization of summation is, perhaps, more apparent in the discrete version of it in eq. (4.7). The first and the second stages together are really only performing locally weighted evaluations, one such for every coordinate, of the Fourier transform, eq. (4.18), for a frequency pair given by $\omega_x = w_x$ and $\omega_y = w_y$, i.e., for the frequency tuning of the Gabor filter. The evaluation in both cases is done with respect to the space-time origin $\vec{x} = (0, 0)^T$. In other words, the common point where the exponential $e^{-i2\pi(\omega_x x + \omega_y y)}$ of the Fourier transform is always at zero phase for any frequency pair $\omega_x$, $\omega_y$ is exactly at the origin. A truly local Fourier transform must be computed as if $(x_0, y_0)^T$ was the common point, i.e., as if $(x_0, y_0)^T$ was at the origin — just like it would be when the (global) Fourier transform is computed on an appropriatelly cropped (small) and centred image. But this is easily achieved with pixel-wise multiplication of the result after the two stages with $e^{i2\pi(\omega_x x_0 + \omega_y y_0)}$. Note that it is equal to multiplying with $e^{iW\vec{y}}$, i.e., the third stage of the staged approach, with the filter frequency tuning $W = [2\pi w_x, 2\pi w_y]$ and $\vec{y} = (x_0, y_0)^T$. With the multiplication we change the values of all individual pixels at $\vec{y}$ to become results of localized evaluations of the Fourier transform with respect to coordinate $\vec{y}$. Afterall, the complex Gabor filtering can be regarded as computations of local Fourier transforms, all evaluated only for a single frequency pair, namely the filter's frequency tuning $w_x$, $w_y$, inside a region given by the filter's envelope.

The last stage resembles the shift theorem, which primarily relates Fourier transforms of some image $I(x, y)$ and its copy $I'(x, y) = I(x + x_0, y + y_0)$ — a shift of $I$ such that $I(x_0, y_0)$ happens to be in the origin, in that it holds [138]

$$\mathcal{F}_{I'}(\omega_x, \omega_y) = e^{i2\pi(\omega_x x_0 + \omega_y y_0)} \mathcal{F}_I(\omega_x, \omega_y). \tag{4.19}$$

It is like we were computing, for every coordinate $\vec{y}$, local Fourier transform at the origin of an image $I'$, which would be the image $I$ shifted by $(-x_0, -y_0)$.

It is still our debt to explain why we are using the word "local" while we should correctly say "locally weighted" because this is what the Gabor filtering actually computes. This is an important difference. For a given pixel coordinate $(x_0, y_0)$, the *locally weighted* Fourier transform $\mathcal{F}'_I$,

$$\mathcal{F}'_I(\omega_x, \omega_y) = \iint_{-\infty}^{\infty} I(x, y)\ e^{-i2\pi(\omega_x(x-x_0)+\omega_y(y-y_0))}\ e^{-1/2\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} dxdy \tag{4.20}$$

$$= \iint_{-\infty}^{\infty} I(x + x_0, y + y_0)\ e^{-i2\pi(\omega_x x+\omega_y y)}\ e^{-1/2\left(\frac{(x+x_0)^2}{\sigma_x^2}+\frac{(y+y_0)^2}{\sigma_y^2}\right)} dxdy, \tag{4.21}$$

does not compute exactly the same result as does the (global) Fourier transform on appropriately cropped and centred image,

$$\mathcal{F}_I(\omega_x, \omega_y) = \int_{-3\sigma_y}^{3\sigma_y} \int_{-3\sigma_x}^{3\sigma_x} I(x + x_0, y + y_0)\ e^{-i2\pi(\omega_x x+\omega_y y)} dxdy. \tag{4.22}$$

The integral limits as well as the property on limited support of the Gaussian function were explained in section 4.1.1 on page 61. It is this limited support of the weighting

function and the notion of how the Gabor filter's carrier is applied on input data that we owe for using the term *local* Fourier transform.

When computing energy after filtering with a quadrature pair, we compute the square root of sum of squared responses of the filters. A quadrature pair, in the context of Gabor filtering, are two real filters with the same Gaussian envelopes and with frequency carriers out of phase by $\pi/2$. Owing to the Euler formula, $e^{ix} = \cos x + i \sin x$, real and imaginary parts of a complex Gabor filter represent a quadrature pair. Computing energy of such filtering then equals to computing magnitude of filtering result. It is a matter of fact that when energy is computed after filtering with complex filter using the staged approach, we can omit the third stage [119]. In the third stage, we multiply results of complex filtering with complex exponentials only of the form $e^{ix}$, the filtering results then keep their amplitudes.

**Alternative approaches**

First of all, recall that despite we are talking mostly only about Gaussian filters, it is because of the fact that Gabor filtering can be efficiently computed using Gaussian filtering [148, 145].

There are only a few examples we are aware of in the literature where 2D Gabor filtering is used for motion estimation. It is rather a rare situation because the most often processed spatial dimensionality is 2D and, owing to the way filters are applied for motion extraction, the processed data is actually in a form of a spatio-temporal stack of 2D images, i.e., it is a 3D image technically. Thus, the majority of filtering techniques for optical flow computation are focused on 3D image processing. On the other hand, authors often aim to decrease the number of frames their methods need to consider. This often ends up with, we may say, classical Gabor convolutions in 2D accompanied with some modified temporal processing/filtering, a system that together resembles motion estimation systems.

An example of this may be the method by Gautama and van Hulle [129]. They suggested a phase-based method based on just *spatial* filtering, in contrast to the *spatio-temporal* filtering used in the influential well-known phase-based method by Fleet and Jepson [83]. As usual, they conduct the spatial filtering with banks of quadrature Gabor filter pairs. For every filter, the authors establish the temporal phase derivative from time-lapse sequences of filter phase responses by performing a least-squares linear regression. The support for the regression is only 5 frames, whereas the spatio-temporal filtering in [83] requires 21 frames. This method has not been the best in any of the tests authors had conducted in their publication but it has been keeping up with the majority of tested methods in terms of accuracy and number of estimated vectors (density of flow fields). But unlike the others, the method allows for shorter temporal support and smaller computational demand of the method. Recently, we have noticed its re-implementation on the GPU with real-time capability [107]. Note that their filter banks consisted of filters with isotropic Gaussian envelopes.

Bernardino and Santos-Victor [148], when seeking very fast implementation for complex Gabor filtering, have shown that it is better in terms of the number of required ops/px to approach the filtering in the staged manner. They considered the 1D recursive filtering of the form given by eqs. (4.13),(4.14), i.e., filters general in use with very short support and depth of recursion both independent of a value of $\sigma$. The savings is about 35% in

computation (ops/px) compared to a single direct recursive Gabor filtering, [148]. This is a result of avoiding an expensive complex arithmetics (convolutions with Gaussian instead of Gabor filters), it is not a result of changing time complexity (like it was when we were shifting from naive 2D convolution to a few separable 1D filters).

The same authors pushed the improvement even further, down to 62% savings, when they replaced the 1D Gaussian IIR filtering in the staged scheme with 1D FIR filtering. But it must be stressed that the FIR filter had support of only 5 pixels representing 1D Gaussian with fixed $\sigma = 0.95$, such filter requires only 9ops/px. Clearly, this is an example of application-taylored filter. In the same fashion, Nestares *et al.* [149] used four 1D kernels of 11 elements in which real or imaginary parts of a desired Gabor were stored. In order to compute a real or imaginary part of a 2D complex Gabor filtering with frequency tuning in one of the four directions (0°, 45°, 90° or 135°), a proper combination of two or four 1D convolutions along the $x$ or $y$ axes were conducted in a serial manner. The decomposition into 1D FIR convolutions is based on properties of trigonometric additions formulas, e.g.,

$$\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha), \tag{4.23}$$

$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\beta)\sin(\alpha). \tag{4.24}$$

Both publications were used to construct a particular multi-scale 2D image representations based on Gabor functions. Both had fixed pre-selected parameters of the Gabor function and, that is important, isotropic Gaussian envelope. As a matter of fact, the GPU-based optical flow computation method [107] is actually using the filtering framework by Nestares *et al.* [149]. Finally, Areekul *et al.* [150] required fast 2D *anisotropic* real Gabor filtering for fingerprint enhancement. They realized that actually only 8 directions, i.e., $0°+k \cdot 22.5°$, $k = 0, \ldots, 7$, are explored in their application. Moreover, it was held in their application that the frequency tuning of some Gabor and main axis of its Gaussian envelope were identical. It was then an easy matter to convolve with one 1D Gabor filter along an axis given by the orientation $k \cdot 22.5°$ and then convolve with one Gaussian along a perpendicular axis (with orientation $k \cdot 22.5°+90°$ modulo 180°). Furthermore, four directions were changed slightly so that the convolution could easily sweep the pixel grid in a direction very close to the originally required one. In fact, this solution is a special case of the one by [33], introduced 3 years later.

### 4.1.4 3D Filtering

The fundamental shift from simple 1D (line) filtering to higher dimension has already been done in the previous section. Indeed, the shift from 2D to 3D is not that dramatical. For instance, if a filter is found to be easily extensible to higher dimensions, e.g., the Gaussian filter with its famous exponential term is very illustrative, we may probably repeat this extension several times again to yield a variant of the filter of any desired dimensionality. The same holds for separability of filters, etc. So, this section could have had title "$n$D Filtering" but we'll stay with 3D since this is the target dimension we want to deal with.

Basically, we make use of the staged approach, eq. (4.17), even in the 3D scenario again to turn the convolution with a general Gabor filter into a modulation, then convolution with a general Gaussian filter followed with a demodulation step. The Gaussian in the middle stage is the Gaussian envelope of the given Gabor filter. Since vast majority of optical flow computation methods use the basic form of Gaussian envelope, for example

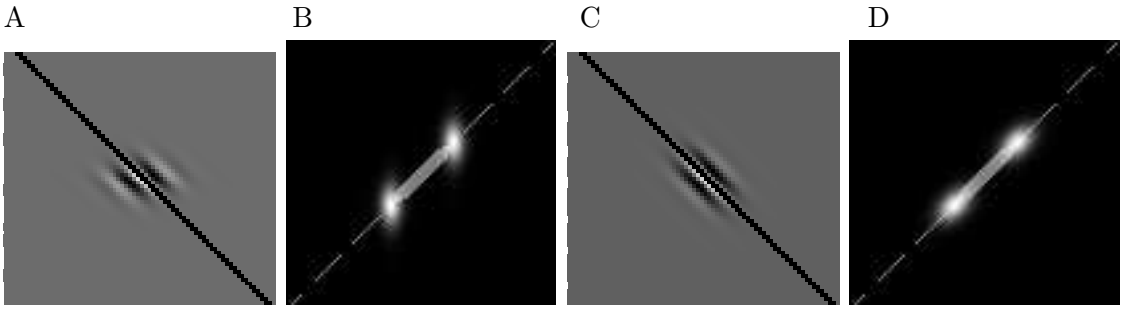


Figure 4.9: Example of motion extraction. The basic anisotropic filter, in A, and general anisotropic filter, in C, are applied to detect rightward translational 1D movement of a small dark patch. The movement is captured with the dark diagonal line. Horizontally runs the spatial coordinate $x$, vertically runs the temporal coordinate $t$. In B and D are the overlays of the Fourier transforms of the (real) filters (lighter gray) over the transform of the moving (real) pattern (darker gray). The general anisotropic filter, in D, appears to match closer than the basic anisotropic filter, in B.

all methods in Section 3.3.4, we could have finished the section right here. Note that the basic form of Gaussian envelope is an extension of the 1D form given in eq. (4.1) to 3D:

$$\mathcal{G}a(\vec{x}, \sigma_x, \sigma_y, \sigma_z) = \frac{1}{(2\pi)^{3/2}\sigma_x\sigma_y\sigma_z}\, e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+\frac{z^2}{\sigma_z^2}\right)}. \tag{4.25}$$

It allows for easy separable filtering with 1D FIR or IIR filters along the coordinate system axes, the $x$, $y$ and $z$. This is very advantageous from the computational point of view.

But the basic Gaussian envelope is rather limiting. To illustrate it, consider the application of two 2D Gabor filters (only real 2D filters with even/cosine phase for the sake of clarity) to a 1D translating dark patch, Fig. 4.9. Notice that the general anisotropic filter closely wraps around the motion pattern in the spatio-temporal represenation as well as around the motion plane in the frequency domain. Such filter is more unlikely to respond strongly to other motions, the filter is very selective allowing for finer sampling of the frequency domain. On the other hand, convolution with such general filter is more computationaly demanding.

This brings us to a question whether this is the only reason why general anisotropic filters are greatly avoided in motion estimation. Is it because the majority of publications on models of early human vision seem to consider mostly only the basic form of Gabor filter as well? This would be a theoretical limit then. But the models may be slightly inaccurate, though we have no evidence for this. But approaching it from the other side, a counter-example exists. It is the nearly 15years old recognized publication by Lee [121], inspired by the work of Daugman and others, who suggested to sample the frequency domain in a log-polar manner, i.e., with nonorthogonal anisotropic Gabor filters both in the frequency and so in the spatial domain too (a filter can't be anisotropic only in one domain). The acceptance of this publication suggests that truly general Gabor filters are probably plausible models. Another question is whether authors of the majority of optical flow computation based on Gabor filtering opted for basic Gabor filters only to ensure the computation of their method is tractable? Maybe because they published their methods prior the year 2006 — a year when optimal separability of any $n$D Gaussian filter [32] has been solved for the first time? Truth is, that the solution suffers from the positional

variability (was already discussed), which we managed to overcome just recently in 2008 [P1]. This would be a practical limit. Anyway, in the appended original publications we show how to diminish the computational burden.

In the view of the current state of the art, Table 4.4, the separably computed convolution with the *basic* complex 3D Gabor filter given in eq. (4.25) and with the aid of recursive 1D filters and the staged approach would require 86ops/px whereas the very general 3D complex Gabor filter can be computed with the staged approach with only 126ops/px or 164ops/px in the positional-*invariant* version [P2], i.e., increase of 47% or 91%, respectively.

The former is achieved when the Gaussian envelope is separated according to Lampert and Wirjadi [32]. They make use of the triangular factorization of Cholesky type decomposition of the Gaussian's covariance matrix $C$, eq. (4.17). This allowed them to rewrite $C = VDV^T$ with $V$ being an upper triangular matrix with diagonal unit,

$$V = \begin{pmatrix} 1 & x_1 & x_2 \\ 0 & 1 & x_3 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.26}$$

$x_{1,2,3} \in \mathbb{R}$ and $D$ being a diagonal matrix,

$$D = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}. \tag{4.27}$$

Since $C$ is a Gaussian covariance matrix, it is symmetric and positive definite. In the general case of $C$ being a $n \times n$ matrix, it has $n(n+1)/2$ degress of freedom (elements on diagonal and upper triangle). The authors require, for practical reasons, to use as many zeros as possible for matrix elements. If zero can't be used, they wish to use one whenever possible. They also require that $V$ has determinant of exactly 1. As $V$ is an upper triangular matrix, the multiplication along diagonal must equal 1. Thus, the diagonal of $V$ consists of ones. It has remained $n(n-1)/2$ free values in the upper diagonal of $V$ and $n$ values on the diagonal of $D$. As $VDV^T$ is equal to $C$, we see that both $D$ and $V$ couldn't have less unknowns than $n(n+1)/2$. But that is exactly what they have together. Hence, their shape is optimal. Considering the argument of the exponential in eq. (4.17), we develop it:

$$-1/2(\vec{y}-\vec{x})^T(VDV^T)^{-1}(\vec{y}-\vec{x}), \tag{4.28}$$
$$-1/2(\vec{y}-\vec{x})^T(V^T)^{-1}D^{-1}V^{-1}(\vec{y}-\vec{x}), \tag{4.29}$$
$$-1/2(\vec{y}-\vec{x})^T(V^{-1})^TD^{-1}V^{-1}(\vec{y}-\vec{x}), \tag{4.30}$$
$$-1/2(V^{-1}(\vec{y}-\vec{x}))^TD^{-1}V^{-1}(\vec{y}-\vec{x}). \tag{4.31}$$

Application of the previous equation and by defining $\vec{x} = V\vec{u}$, we may finally rewrite the staged approach, eq. (4.17), in the *new coordinate system*:

$$O(V\vec{v}) = e^{iWV\vec{v}} \cdot \left[ \sum_{\vec{u}} \left[ I(V\vec{u})e^{-iWV\vec{u}} \right] \cdot e^{-\frac{1}{2}(\vec{v}-\vec{u})^TD^{-1}(\vec{v}-\vec{u})} \right]. \tag{4.32}$$

| | real domain<br>$a_{i,j} \in \mathbb{R}$ | integer domain<br>$a_{i,j} \in \mathbb{Z}$ |
|---|---|---|
| 2D | Geusebroek *et al.* [30]<br>$m = 2$ | Lam and Shi [33]<br>$m = 3$ |
| 3D | Lampert and Wirjadi [32]<br>$m = 3$ | Ulman [P1]<br>$m = 6$ |
| $n$D | Lampert and Wirjadi [32]<br>$m = n$ | Ulman [P1]<br>$m = n(n+1)/2$<br>(bases not given explicitly) |

Table 4.4: An overview of the state-of-the-art solutions on separability of any arbitrary $n$D Gaussian filter into $m$ directions for 1D convolutions. The table is divided into two columns with respect to the domain used for the direction vectors $\vec{b}_i = (a_{i,1}, \ldots, a_{i,n})^T, i = 1, \ldots, m$, i.e., whether interpolations must be used during convolution. The table is applicable also to Gabor filtering if it is conducted via the staged approach.

We observe that the original arbitrary Gaussian has taken the basic form here. In the 3D case, this means that it is possible to conduct simple 1D convolutions along the new coordinate system axes, namely $\vec{b}_1 = (1, 0, 0)$ with $\sigma_1$, $\vec{b}_2 = (x_1, 1, 0)$ with $\sigma_2$ and $\vec{b}_3 = (x_2, x_3, 1)$ with $\sigma_3$. We remind that $W$ is, in the 3D, a $1 \times 3$ row matrix with (de)modulation frequencies along the original coordinate system, the $WV$ is a $1 \times 3$ row matrix with frequencies along the new coordinate system. The recipe on 1D convolution in $nD$ was already given earlier in eq. (4.8).

The latter increase of 91% in computation demand is due to our extension [P1] of the positional-invariant technique originally developed only for 2D by Lam and Shi [33]. The technique leads to the same principle as the one by Lampert and Wirjadi, except that the matrix $V$ is replaced with matrix $A$,

$$\vec{x} = A \cdot \vec{u} = \left[\vec{b}_1\ \vec{b}_2\ \vec{b}_3\ \vec{b}_4\ \vec{b}_5\ \vec{b}_6\right] \cdot \vec{u}, \tag{4.33}$$

in which the vectors $\vec{b}_i$ belong entirely only to the domain of integers. The direction vectors that pushed the 1D convolutions off the pixel grid were replaced with two or more direction vectors that don't do that. The rapidity of filtering was traded for stability and accuracy of the filtering as this solution is positional-invariant and offers slightly higher accuracy [P1]. Nevertheless, we will show in another our original publication that if a Gabor bank meets some basic constraints on its design, we can actually convolve with it in the positional-invariant version with the increase of up to 62% ops/px [P2] compared to the bank of Gabor filters in the basic form. So the lack in performace is decreased while the positive properties are kept.

Note that both solutions on how to separably convolve with Gaussian/Gabor filter are exact. One may use whatever he/she wants for 1D convolution routine. In this thesis we use the fast recursive filters [143].

Finally, let us return to the question of why not to use general anisotropic Gabor filters. We have learned that the practical limit has been broken since even the general anisotropic filtering has now the linear time complexity. We argue that a theoretical limit based on human vision properties, if there is such, should be avoided. The general filters

allow for "more specific" shapes, which, in turn, may aid the motion extraction and optical flow computation. If humans are limited in some respect, e.g., require longer observation period to discover motion parameters, why can't a computer program perform better? For instance, scientific cameras mounted on microscopes perform a lot better in terms of sensitivity to incoming light than human's naked eye. Regarding the Gabor filtering, consider the example with the translating white square, Fig. 4.10. The Fourier transform shows negligible short line in the direction (1,1,-2), which is a direction perpendicular to the translational vector (1,1,1) of the square. Still, the anisotropic filters managed to react on the motion: we see a clear distinction between responses of filters tuned to different spatial ($x$-$y$) orientations. The more the filter orientation declines from the orientation of the true motion, the weaker response it shows. The isotropic filters had a very decent distinction in responses. Obviously, their measured data is rather worse leaving more room for incorrect motion parameters determination in further stages of an optical flow computation.

We conclude this section by offering an example of a collection of dense Gabor banks which we can convolve with within reasonable time frame, i.e., up to couple of tens of seconds on recent desktop computers. The collection consists of three banks, each is designed to detect velocities of 1px, 1.6px and 3px per frame, respectively. Each consists of 8 filters such that the spatial halfplane is sampled in orientation by 22.5°, refer to Fig. 4.11 where the collection is shown in the Fourier domain. Every filter has its envelope of the same common size, in particular $\sigma_1 = \sigma_2 = 10, \sigma_3 = 2$. The envelopes differ only in the orientation because every envelope closely wraps the carrier part of its filter, see Fig. 4.12. The modulation frequency in this illustrative case was $1/5\text{px}^{-1}$. Note that the spatial images of the filters strongly resemble edge detection or derivative filters. We hope that setups like this or similar would help the filtering-based optical flow methods to catch up again with the derivative-based approaches, just like it once used to be and like the theory dictates [110, 105].

Regarding the alternative approaches, we have came across only one due to Wirjadi and Breuel [31] who have devised an approximate separable anisotropic Gaussian filter. Similarly to the 2D [30], they employed a cascade of three 1D convolutions in the directions along the $x$ and $z$ axes and along a general third axis within the 3D image coordinate system. We now see, due to the publication [32] published a year later, that such three axes couldn't provide an exact solution. Hence, it is only an approximate but with good error rate most of the time. Still, we would be rather conservative in its use whenever the IIR filters are to be employed along the three directions as the IIR filters are approximations as well and the error rate may cumulate.

When regarding the spatial filtering in 4D, we return to the idea from the beginning of this section: the spatial filtering in 4D, i.e., the spatio-temporal filtering in time-lapse stack of 3D images, can be approached in exactly the same manner as we have outlined for the 3D filtering.

### 4.1.5 Comparison with filtering in the Fourier domain

Before we answer the question, let us explain how to conduct Gabor filtering in the Fourier domain. We aim to make use of the convolution theorem that relates results of convolution in the image domain with results obtained in the Fourier domain. In particular, it

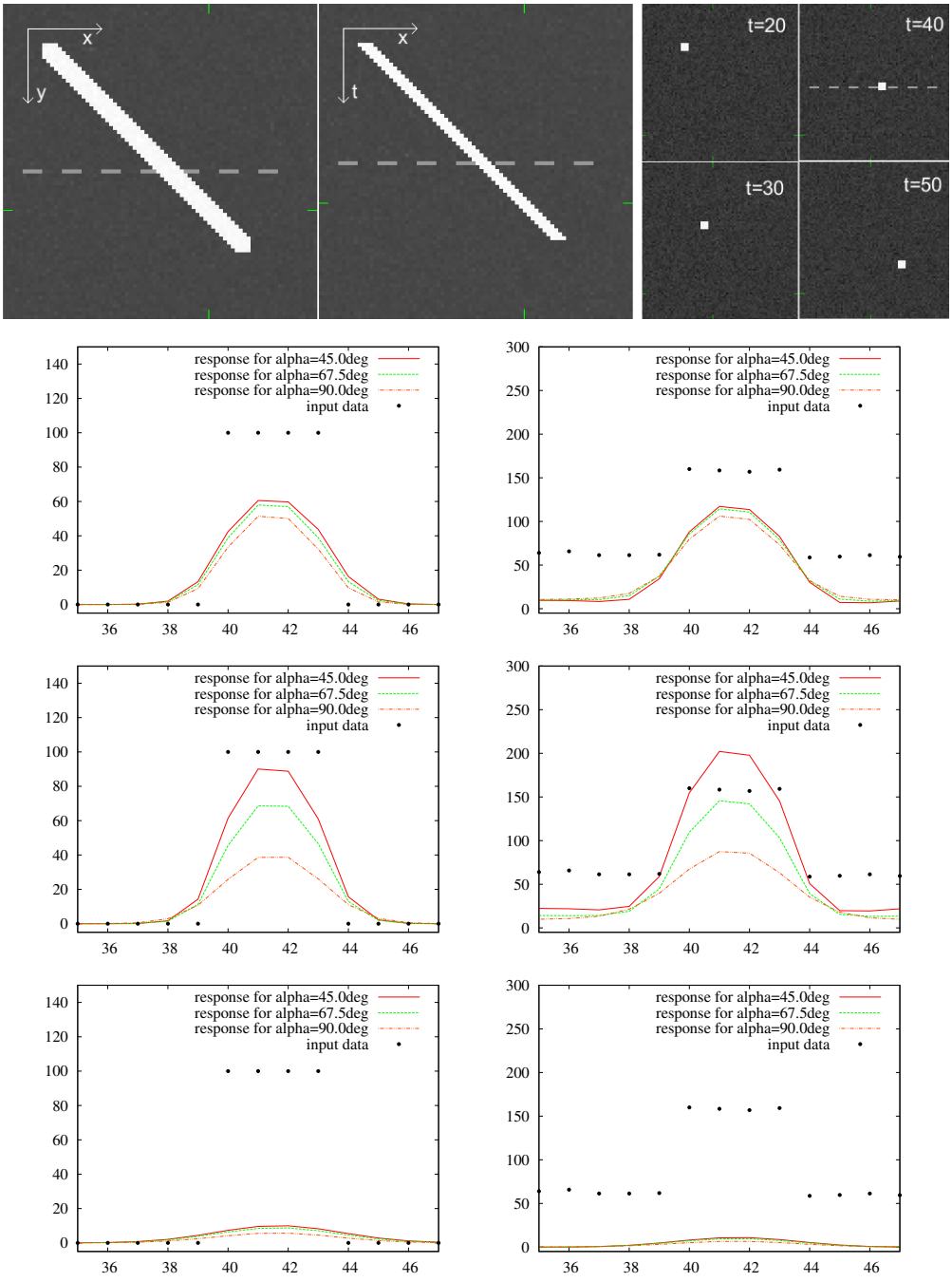


Figure 4.10: Top row: Maximum intensity projection of 2D+t image shows a moving spot on a random background. Four 2D frames from this sequence are shown as well, in the right-hand-side. The three rows of plots, from top to bottom: Intensity profiles ($x$ axis shows pixel offset, $y$ axis shows pixel value) drawn along the dashed line from filtering results of the three banks 2–2, 2–1 and 1–1 on ideal data (left column) and data with noise (right column). The magnitude (energy) of complex response is depicted. The shape of the Gaussian envelope is given with the notation A–B where A is $\sigma$ of the envelope in the direction of $(1, 1, 1)^T$ and B is $\sigma$ in the other two perpendicular directions. The parameter "alpha" tells the spatial orientation of the filter. The figures are reprinted from our original publication [P2] for convenience.

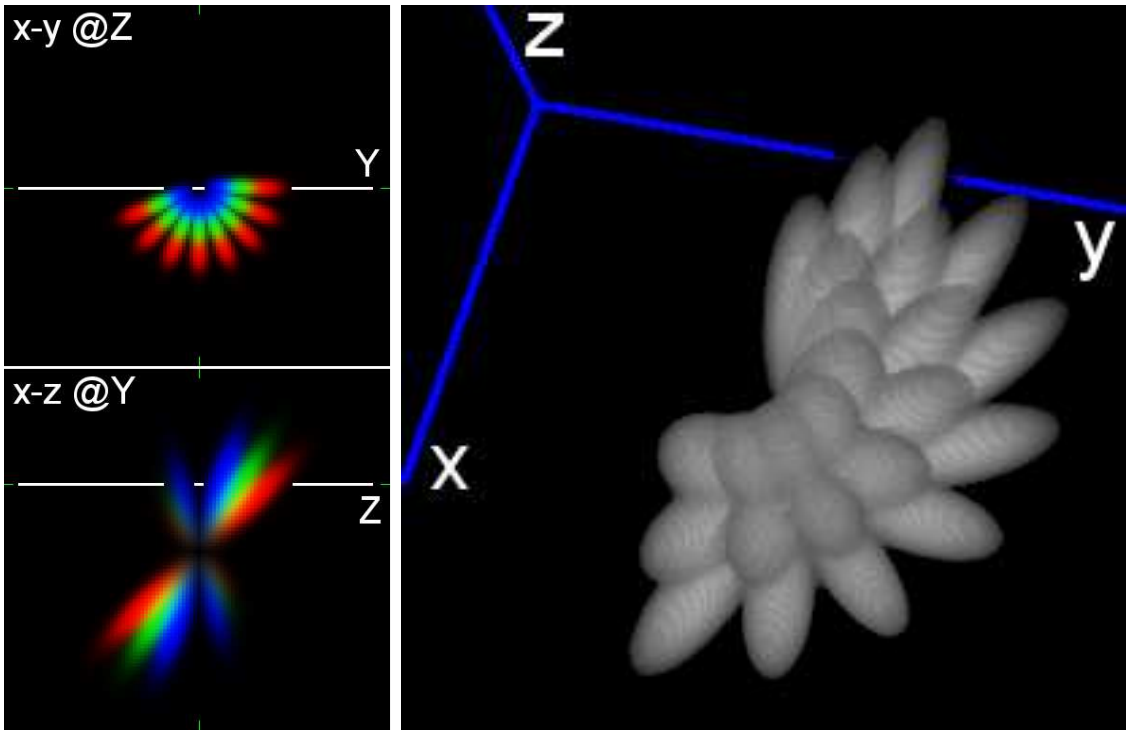


Figure 4.11: Sample collection of real Gabor banks shown in the Fourier domain. Two cross-sections in the $\omega_x$-$\omega_y$ plane in top left and in the $\omega_x$-$\omega_z$ plane in bottom left illustrate the orientation sampling and velocity tuning, respective, of the collection. Different colour (tone of gray in B&W print) encodes filters tuned to different magnitude of velocity. The banks consists of more elongated narrow-bandwidth filters which samples the Fourier halfplane at a finer grain. A 3D visualization of the collection is given in the right. For visualization purposes, its second symmetric part is intentionally missing.

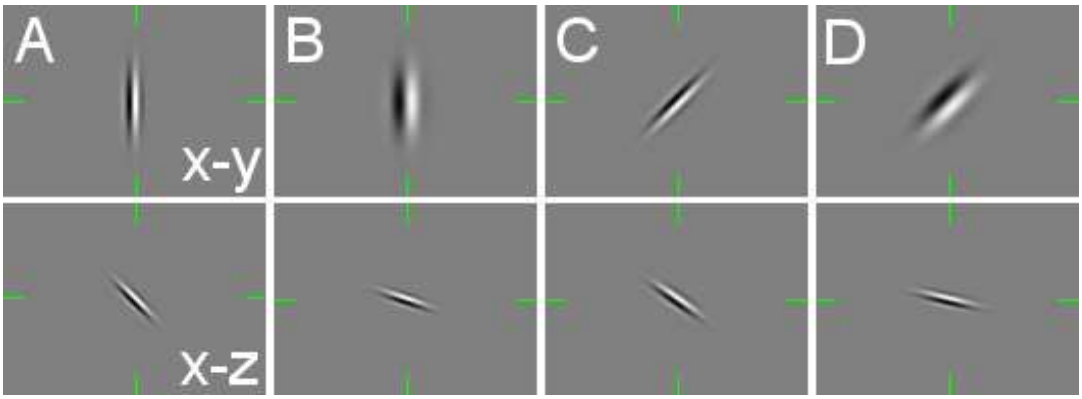


Figure 4.12: Four filters of the sample collection of Gabor banks shown in the spatial domain. Only imaginary parts are shown. The two filters, in A and B, detect the rightward motion once with velocity of 1px per frame and once with 3px per frame, respectively. In C and D, example of two filters tuned to the same velocities but different direction.

states that Fourier transform of a convolution result is equal to a result of element-wise multiplication of Fourier transforms both of the convolved image $I(\vec{x})$ and a convolution kernel $k(\vec{x})$:

$$I * k = IFT\left(\sum_{\vec{\omega}}[FT(I)](\vec{\omega}) \cdot \mathcal{F}_k(\vec{\omega})\right) \tag{4.34}$$

where $FT$ and $IFT$ stands for the operation of the (forward) Fourier trasform and inverse Fourier tranform, respectively. The $\mathcal{F}_k(\vec{\omega})$ is a result of the transform $FT(k)$. The $\vec{x}$ is image domain coordinate vector while the $\vec{\omega}$ is the Fourier domain frequency vector.

This approach consumes $\mathcal{O}(2 \cdot (n \log n) + n)$ operations, i.e., $\mathcal{O}(n \log n)$, when the fast implementation such as the one by Johnson and Frige [146] is used and when $n$ is the number of all pixels in the transformed image. The calculation is altogether for the fast Fourier transform of the input image, element-wise multiplication and fast inverse Fourier transform. The transform of the filter kernel is not included as this can be done in advance and stored in a LUT (abbreviation for a Look-Up Table). However, the size of the LUT depends on the input image because the size of the transformed kernel must be exactly the same as the size of transformed image.

We now draw our attention to the counts of operations per pixel required by the approach. Considering probably the most often used library for the fast Fourier transform[2], the algorithm implemented therein requires [146]

$$\frac{34}{9} n \log_2 n - \frac{124}{27} n - 2 \log_2 n - \frac{2}{9}(-1)^{\log_2 n} \log_2 n + \frac{16}{27}(-1)^{\log_2 n} + 8 \tag{4.35}$$

operations to transform the whole image consisting of exactly $n$ pixels. Considering a 2D image of $512 \times 512$ we see that the fast Fourier trasform requires 16621840 ops in total. After normalizing with number of transformed pixels we arrive to 63.4ops/px. Thus, utilization of the convolution theorem necessitates $2 \cdot 63.4 + 6 = 132.8$ops/px. Note again that this number is an *increasing* function of $n$. If we use the recursive filters devised by Young *et al.* [143] with the staged approach with Gaussian filtering without the use of interpolation [33] including the zero-mean correction [148, 145], we arrive to the constant of 127ops/px irrelevant to the image size and Gabor filter parameters.

This observation is also supported in the work by Bernardino and Santos-Victor [145] who has tabulated the number of required ops/px for the fast Fourier trasform for some image sizes. From their results we read that the 2D staged filtering with recursive 1D Gaussian filters and zero-mean correction is faster for any image of size $256 \times 256$ or larger. It is also noteworthy that the Fourier transform cannot naturaly compensate for boundary effects which, most of the time, results in enlargement of the original input image with proper boundary prior taking its transform, which, in turn, results in even greater number of ops/px (not mentioning also the need for auxiliary image buffers to store the enlarged copies).

When considering a development of time consumption with respect to the number of processed pixels, the fast Fourier transform also suffers from, sort of, erratic behaviour, refer to Fig. 2 in the publication by Wirjadi and Breuel [31] for comparison on 3D images. The authors measured a total time for the forward and inverse transform plus the time for multiplication in the Fourier domain not including, as expected, the time to transform the

[2] http://www.fftw.org/

convolution kernel. The erratic behaviour is due to the fact that more efficient computation of the transform was available for certain image sizes, this is a well-known feature of the fast Fourier transform. Since the expresion for total ops/px for the spatial filtering does not include any term with image size, unlike the eq. (4.35), the time consumption grows proportionally with the size of the convolved image. This is also evidenced in their work. In the similar fashion, Young *et al.* , in Fig. 5 of [126], was comparing time consumptions of 1D filtering with respect to the size of $\sigma$. They have obtained two constant curves, the one for spatial filtering showed smaller times. Again, the measure of ops/px can be used to explain this. However, we realize that recursive filters do not change their support with $\sigma$. The Fourier transform provides convolution with virtually any filter, including Gaussians with different $\sigma$, with the same rapidity provided the filter is kept smaller than the input image.

On the other hand, Rahman *et al.* [151] has just recently reported very fast implementation of Gabor bank filtering. The filtering was taken in the Fourier domain and processed, together with the Fourier transforms, on a recent graphical card (GPU) using the CUDA [152]. They report that such Gabor bank filtering with 24 filters in total together with a few other operations, e.g., normalizations and summations, to simulate the image processing in the static pathway of a spatio-temporal visual saliency model achieved 180 times shorter computation time than their former CPU-based implementation in the C programming language. To process the whole pipeline it required around 47ms for single frame of $512 \times 512$ size [151].

## 4.2 The original publications

We have presented currently available means to perform convolution with complex Gabor filters. We have provided a necessary theoretical and technical background so that we are ready to proceed with reading our original publications on filtering [P1, P2, P3] as well as the publications on the application of filtering in the optical flow computation method [P4]. The rest of this chapter will cover selected topics from the publications in more detail.

The first publication [P1] "Arbitrarily-Oriented Anisotropic 3D Gaussian Filtering Computed with 1D Convolutions without Interpolation" proposes a way to convolve with a general anisotropic Gaussian filter. The solution introduces a new coordinate system given with a set of base vectors with the following two main features: the base vectors define convolution directions that can't fall off the pixel grid and the set of such vectors is over-determined in the sense that the base vectors are not mutually linearly independent. The notation should be understood as purely a technical one with only a certain parallel in the usual terms in mathematics. Since there are more vectors defining the coordinate system, the solution contains inherently some redundancy and, as such, it is not optimal in terms of required ops/px. But it is stable, i.e., position-invariant, and slightly more accurate. The solution is general for $n$D filtering, Table 4.4 on page 78, however, it is tested and presented for the 3D case in the publication.

The second publication [P2] "Filtering with Anisotropic 3D Gabor Filter Bank Efficiently Computed with 1D Convolutions without Interpolation" follows on the results of the first one. It deals with complex Gabor bank filtering approached in the staged

manner. It shows that if a bank fulfills some constraints on its structure, the inherent redundancy of underlying Gaussian filtering can be diminished. The main result is that certain quadruples of complex Gabor filters can be computed with optimal consumption of ops/px even when the redundant but otherwise stable and more accurate Gaussian filtering is employed.

The third publication [P3] "Boundary Treatment for Young–van Vliet Recursive Zero-Mean Gabor Filtering" proposes a correct and effective initialization of 1D recursive filtering based on the Young *et al.* family of filters, eqs. (4.13),(4.14). It also gives a formula on how to easily compute the scaling coefficient for, so called, zero-mean correction of the filter [143, 148, 145]. We believe that correct use of popular Young *et al.* recursive filtering is now described completely in the literature, see Table 4.3 on page 68. This publication is currently subject to final minor revision.

The fourth publication [P4] "Improving Accuracy of Optical Flow of Heeger's Original Method on Biomedical Images" proposes two major changes to the acknowledged and recognized energy-based method for optical flow computation by David Heeger [106], which was already briefly explained in Section 3.3.4 on page 53. The method's framework was kept, only filtering and weighting "subsystems" were changed. The changes led to greatly improved performance on "set1" and slightly improved performance on "set2", both sets consisted of time-lapse fluorescence microscope images. All test images were from artificially generated ground-truth datasets (will be explained in the next chapter).

## 4.3 Additional notes to the original publications

### 4.3.1 On Gabor filter symmetries

This section is related to our second publication [P2], which deals with Gabor filtering banks. Details are given in the publication. We only remind here that the banks consist of 3D complex Gabor filters. Each is given with six tripples of the form $(\vec{b}_i, \sigma_i, w_i)$, $i = 1, \ldots, 6$ where $\vec{b}_i = (a_{i,1}, a_{i,2}, a_{i,3})^T$ is an integer *base* vector along which a 1D convolution should happen with Gabor filter with $\sigma_i$ and frequency $w_i$. The tripples are computed using our developed methods [P1, P2]. The input filters are given with the steering angles, as in Fig. 3.5 on page 44, and other parameters for which refer to the original publication Section 2.

We would like to show proof of the statement that given a filter with $\alpha$ (and the other parameters, e.g., $\bar{\sigma}_{1,2,3}$) whose six tripples are $(\vec{b}_i, \sigma_i, w_i)$, the same filter but with $\pi - \alpha$ will have tripples $(\vec{b}'_i, \sigma_i, w_i)$ with $\vec{b}'_i = (-a_{i,1}, a_{i,2}, a_{i,3})^T$. In order to do that we need to see the content of the matrix $C$, which is the Gaussian symmetric matrix and the envelope of the Gabor filter:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix} \tag{4.36}$$

where

$$
\begin{aligned}
c_{11} &= \cos^2\alpha\,\cos^2\beta\,\bar{\sigma}_1^2 + \sin^2\alpha\,\bar{\sigma}_2^2 + \cos^2\alpha\,\sin^2\beta\,\bar{\sigma}_3^2, && (4.37)\\
c_{12} &= \cos\alpha\,\cos^2\beta\,\sin\beta\,\bar{\sigma}_1^2 - \sin\alpha\,\cos\alpha\,\bar{\sigma}_2^2 + \sin\alpha\,\cos\alpha\,\sin^2\beta\,\bar{\sigma}_3^2, && (4.38)\\
c_{13} &= \cos\alpha\,\cos\beta\,\sin\beta\,(\bar{\sigma}_1^2 - \bar{\sigma}_3^2), && (4.39)\\
c_{22} &= \cos^2\beta\,\sin^2\alpha\,\bar{\sigma}_1^2 + \cos^2\alpha\,\bar{\sigma}_2^2 + (\sin^2\beta\,-\cos^2\alpha\,\sin^2\beta\,)\bar{\sigma}_3^2, && (4.40)\\
c_{23} &= \sin\alpha\,\cos\beta\,\sin\beta\,(\bar{\sigma}_1^2 - \bar{\sigma}_3^2), && (4.41)\\
c_{33} &= \sin^2\beta\,\bar{\sigma}_1^2 + \cos^2\beta\,\bar{\sigma}_3^2. && (4.42)
\end{aligned}
$$

The matrix $C'$, which is the Gaussian matrix for the second filter with $\pi-\alpha$, has the same form but with $c'_{ij}$. Note that $\cos(\pi-\alpha) = -\cos\alpha$ and $\sin(\pi-\alpha) = \sin\alpha$. We substitute that into $C'$ and observe that:

$$
\begin{aligned}
c'_{11} &= c_{11}, && (4.43)\\
c'_{12} &= -c_{12}, && (4.44)\\
c'_{13} &= -c_{13}, && (4.45)\\
c'_{22} &= c_{22}, && (4.46)\\
c'_{23} &= c_{23}, && (4.47)\\
c'_{33} &= c_{33}. && (4.48)
\end{aligned}
$$

The matrix for the second filter differs from the matrix for the first filter only in sign of the two elements. We now turn our attention to eq. (9) and, especially, to equivalent eq. (10) in the publication [P2]. Comparing the systems in eq. (10) for the first and the second filter, it is easy to see that if we change sign of all $a_{i,1}$ and consequently in the two elements $c_{12}$ and $c_{13}$ in the system of the first filter, we arrive to the system for the second filter. The solution to both systems, the column matrix with $\sigma_i$, is the same, i.e., $\sigma_i$ are kept. In the same fashion, the matrix $W$, which is given in eq. (3) in the publication as

$$W = [\cos\alpha_S \cos\beta_S, \sin\alpha_S \cos\beta_S, \sin\beta_S] \tag{4.49}$$

where $\alpha_S$ and $\beta_S$ is the orientation of the filter carrier, differs for the two filters only in the sign of its first element. Denote $W'$ as the $W$ with the changed sign. The frequencies $w_i$ in both tripples are (the same) results of multiplications $W\vec{b}_i$ respective $W'\vec{b}'_i$. To conclude, we see that by changing sign of the first element in all base vectors for the first filter with $\alpha$, we obtain the tripples for the second filter with $\pi-\alpha$.

This result is then used to provide a mask of base vectors that can be used in the computation of both filters [P2]. This enables to *share* some 1D convolutions between the two filters and to save some computation time as well. The other symmetries suggested in the publication can proved similarly.

### 4.3.2 Note on bank filtering efficiency

Considering Fig. 9, The histogram of filtering efficiency, in the publication [P2], we see that the 100% efficiency has never been reached. That may indicate that there is actually no adequate filtering quadruple that would score the 504ops/px (like it does the competing approach by Lampert and Wirjadi [32] for four Gabor filters). Problem with Fig. 9 is that

it computes efficiency for *all* convolutions with a *whole* bank. The tests in the publication [P2] were designed such that every bank contained also a pair with $\alpha = 0$ and $\alpha = \pi/2$, which we treated separately as *only* a pair. According to the publication, the optimal convolution efficiency can't be reached in practice for only a pair of filters. We have, therefore, never achieved the (overall) optimal efficiency for any bank in the tests in the publication. Despite that, the efficiency is on average very good.

### 4.3.3 Zero-mean Gabor filters

Following Section V from our last publication [P3] we realize that when conducting 1D complex Gabor filtering, the real part of the filter may offset its responses, see Fig. 7 in the publication. This is because [121] the real part of Gabor filter is even whereas the imaginary part is odd. Averaging impulse response values of the imaginary part of any 1D Gabor filter, we always obtain zero. This is where the term *zero-mean* has come from. Averaging impulse response values of the real part of any 1D Gabor filter, we actually compute its discrete Fourier transform for the "zero" frequency, the DC response. This value is not always guaranteed to be zero as well. Note that this property of Gabor filtering is important for the filtering-based optical flow computation methods because both approaches base their velocity estimations directly on real and imaginary responses of the filtering. If the real component would dominate, as a result of (artificially) increased values by the offset, estimations would be biased.

The offset depends solely on parameters of the filter used. This includes not only the shape of the filter, i.e., its Gaussian's envelope parameters and frequency tuning. If approximation to the filter is used, such as the recursive filters, the offset also depends on particular filter coefficients as different variants of the recursive filter behave slightly differently producing different Fourier DC responses. Clearly, the offset is present in arbitrary $n$D Gabor filtering.

In order to remove this offset from a filtering result, we have adopted the method of Lee [121]. For a given $n$D complex Gabor filtering, the method requires to additionally filter input image with the Gaussian envelope of the Gabor filter. The result after this additional filtering is multiplied with a scale constant and subtracted from the Gabor filtering result. Since Gaussian filtering is a real filtering, the subtraction modifies only the real part of the complex Gabor filtering result. The scale constant/coefficient is directly the DC response of the Gabor filter [121, 148].

This solution works beautifully for FIR filters. If IIR (recursive) filters are used then any formula for Fourier transform of a Gabor filter, such as eq. (3.26) on page 45, cannot be used. For 1D IIR filters from the Young's *et al.* family [126], i.e., those based on eqs. (4.13),(4.14), closed form formulae for the scale coefficients have been devised recently [145] and [P3]. They can extended to $n$D filtering only if the given Gabor filter is separable along the coordinate system axes [145]. For any filtering (FIR or IIR) with general $n$D Gabor we propose to compute the scale coefficient from an experiment.

Our experience with 3D IIR filtering shows that it suffices to use an image of size $50\times50\times50$ pixels filled with constant value. We apply once the given Gabor filtering as well as the additional Gaussian filtering on this image. Taking the two results from centre of both images and dividing them, we obtain the scale coefficient. Note that both filtering is expected to produce constant responses on constant inputs. The centre value is

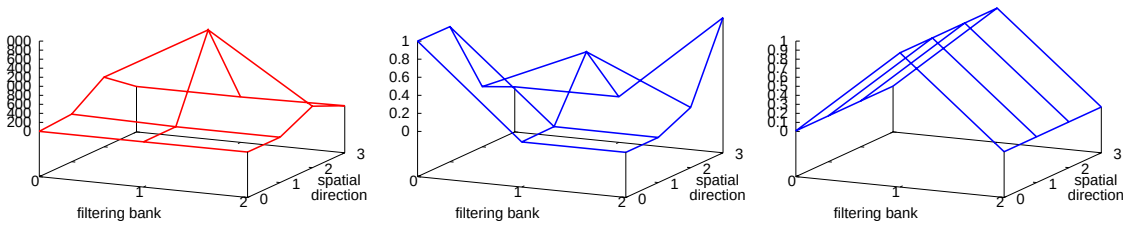


Figure 4.13: From left to right: Twelve measured responses of the original method on real data in region with no motion; ideal responses of the same filters for velocity (-2,0) that best "matches" the measured ones according to the original weighting scheme; ideal responses of the same filters for correct velocity (0,0) that should have been determined in this case. The axes show filter tuning to velocities $(v-1)$px/frame (if the filter belongs to $v$th filtering bank) in the spatial direction: 0→0°, 1→45°, 2→90°and 3→135°.

advantageous for recursive filters as it is sufficiently far from borders should the IIR filter produce spurious responses after reaching the border.

### 4.3.4 Energy-based optical flow method

In this section we would like to comment on our application of the fast spatial anisotropic filtering in the optical flow energy-based computation method [P4]. We would like to stress that the presented results are only *preliminary*. The topic currently lacks deeper analysis of filter responses on real and tested images that would discover optimal filter tuning.

The original energy-based method aims to mimic the human visual system [25] by utilizing a collection of bandpass filters. Their purpose is to provide a *coarse* preview, by means of a collection of energy responses, of the Fourier spectra present in the visual input (the image sequence) based on which the dominant motion is estimated. We remind that the method, for every pixel, basically seeks optimal velocity by seeking a collection of *ideal* responses, which is a function of velocity, that matches a collection of the *measured* responses.

We have tried to increase the size of the filter collection as well as to optimize the filter tuning to obtain a denser and a more selective preview of the Fourier spectra. The distribution of Fourier images of the proposed filtering ensemble is shown in Fig. 3 in the publication [P4]. The filters were tuned to examine 4 spatial directions (0°, 45°, 90° or 135°) and 9 velocity magnitudes (-4px/frame, ..., -1px/frame, 0px/frame, 1px/frame, ..., 4px/frame). In order to make good use of the proposed collection of filters, we had to modify the weighting scheme of the original method. The original method focused to minimize the overall error. We propose to favour collection of ideal responses whose strongests peaks correspond with strongests peaks in the measured collection, see Fig. 4.13.

Note that this work has been an early attempt, to the best of our knowledge, in fully employing the anisotropic shape of filters for optical flow. In spite of it, the filtering collection managed to achieve two promising results: it managed to distinguish between velocities faster than 1px/frame (example given in Fig. 4.14) and it managed to extract relatively correct velocities inside a poorly textured region of motion (right-most column in Fig. 4 in the publication and obtained flow fields in Fig. 6 in the publication [P4]). The former result deserves two more comments. Firstly, recalling the coarse-to-fine processing

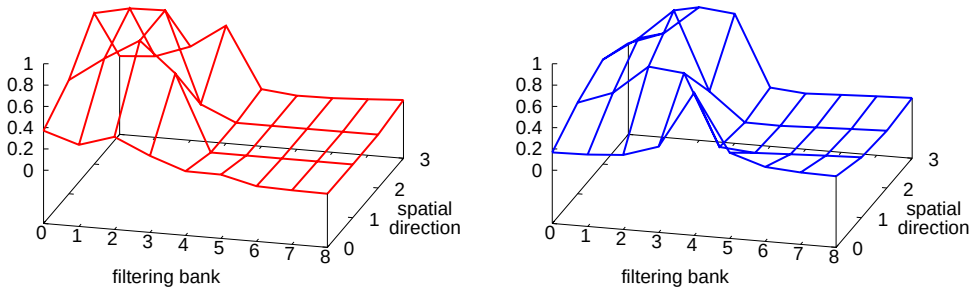


Figure 4.14: Examples of measured, in the left, and ideal, in the right, responses of the proposed collection of 9×4 filters. The measured responses were obtained on real data translating at velocity (0,-2)px/frame. The ideal responses were computed for the same velocity. Filter tuning is given as in Fig. 4.13, velocity is given as $(v - 4)$px/frame.

framework from Section 3.2 we realize that we may decrease the number of levels in the underlying pyramid as well as the number of warping steps if we can handle greater velocities at single level. Secondly, we suggest to conduct further analysis to discover why in Fig. 4.14 there is no such strong distinction between responses on real data translating at velocity of magnitude 2px/frame while there is a single strong peak in the ideal responses of exactly the same filter collection.

Afterall, the proposed method managed to improve the average accuracy from 35.9° to 13.8°, a dropdown by 61%, on the test dataset "set1". We read this as a clear indication that the proposed method has potential to provide accurate flow fields.

# Chapter 5

# Generator for evaluation of optical flow

Once we have implemented an optical flow computing method, we may be interested in the following three questions. Firstly, how good does the method perform compared to other methods, e.g., to the state-of-the-art methods or simply to other variants of the same approach. This immediately raises another question. Secondly, on what type of motions or displayed situations in image sequences the method works better or worse. For instance, can it also handle splitting of cells or only a simple movement of these? Or possibly a more low level question is on what type of image data is the method still applicable? For instance, how much of noise can the method stand up? Thirdly and last but not least, we should ask about reliability and accuracy of the method. This is an important question to answer before one is about to use the method in real applications. For instance, if the method is to be applied to provide data for some velocity measurement, it is vital to understand its accuracy. If the method should provide data for some detection of movements, it is good to understand its reliability in discovering motion, and so on.

We address these topics in the following sections. We will provide an overview of the current solutions to the topic by other authors and give some rationale to introduce our own solution, which is presented in the second half of this chapter.

## 5.1 Evaluation of optical flow methods

### 5.1.1 Evaluation against ground-truth dataset

Seeking a way to answer these questions, we realize that we are seeking a tractable, repeatable, undenyable, quantitative, representative and objective evaluation of an error of an optical flow computation method. As for a quantitative assessment of error rates, the most popular approach is to measure accuracy of computed flow vectors by means of angular error measure function, eq. (5.1), of computed and expected vectors [83]. The expected vectors are often called the ground-truth vectors to signal that they are representing a correct solution. Consequently, the ground-truth flow field is a flow field with ground-truth vectors. Thus, a complete ground-truth dataset for evaluation consists of a test image sequence with associated ground-truth flow fields for every pair of consecutive images in the sequence. An average from computed angular errors is then computed.

This was the approach to compare between methods, besides dealing with qualitative properties, in acknowledged survey publications [72, 81, 96, 97]. And it is still a common approach to validate quality of proposed methods in many individual publications nowadays [153, 94, 73, 154, 103] (citation list is not ment to be complete).

Such ground-truth benchmark datasets also enable us to perform repeatable, undenyable and quantitative evaluation. In order to meet the remaining requirements on evaluation as well, the benchmark images must closely represent the type of real images for which the performance measurements of a tested method are actually desired to be obtained [96, 155, 156, 154, 157]. In other words, one should test a method ideally on data on which the method is expected to be applied. Finally, obtaining or preparing benchmark datasets for given application should also be relatively easy and error-prone.

### 5.1.2 Error measures

Aside from the "classical" angular error measure,

$$AE(v, v_{gt}) = \arccos\left(\frac{(x, y, 1) \cdot (x_{gt}, y_{gt}, 1)}{|(x, y, 1)| \cdot |(x_{gt}, y_{gt}, 1)|}\right), \tag{5.1}$$

for a computed flow vector $v = (x, y)$ and a ground-truth vector $v_{gt} = (x_{gt}, y_{gt})$, the another widely adopted measure since Otte and Nagel [158] is the endpoint error measure:

$$EP(v, v_{gt}) = |v - v_{gt}|. \tag{5.2}$$

The $u \cdot v$ is a dot product of vectors $u$ and $v$ and the $|v|$ is the L2-norm of a vector $v$. The angular measure is sometimes denoted as a *relative* measure since it measures angular deviation of the computed flow from the correct/expected one regardless the magnitude of vectors (due to the normalization in the denumerator). The constant element is appended to avoid divisions by zero when zero-length vectors are used, what has enabled the measure to be defined for any two flow vectors. It also allows to distinguish between collinear computed and ground-truth vectors. Although relativization is often welcomed, in this case it was critized [158] because the same (absolute) deviation yields higher penalization for vectors of smaller magnitude. This was the reason to establish the measure of *absolute* error in flow endpoint, the endpoint measure, eq. (5.2). In fact, both measures are used together nowadays as exemplified in the recent publications [155, 97, 154, 103]. However, even the endpoint measure suffers from similar deficiency when normalized. McCane *et al.* [96], for example, suggests to correct both measures with appropriately tuned thresholds $\delta$ and $T$:

$$AE(v, v_{gt}) = \arccos\left(\frac{(x, y, \delta) \cdot (x_{gt}, y_{gt}, \delta)}{|(x, y, \delta)| \cdot |(x_{gt}, y_{gt}, \delta)|}\right), \tag{5.3}$$

$$EP(v, v_{gt}) = \begin{cases} \frac{|v - v_{gt}|}{|v_{gt}|} & \text{if } |v_{gt}| \geq T, \\ \left|\frac{|v| - T}{T}\right| & \text{if } |v_{gt}| < T \text{ and } |v| \geq T, \\ 0 & \text{if } |v_{gt}| < T \text{ and } |v| < T. \end{cases} \tag{5.4}$$

The values of $\delta$ and $T$ depend on how much one is not interested in measuring errors of small vectors because the greater the thresholds are, the less small vectors contribute to

the overall error, and vice versa. Occasionally, one comes accross a different flow error measure to evaluate results of some optical flow method, e.g., Galvin *et al.* [81] introduced the error normal to gradient,

$$EPvs.G(v, v_{gt}, \nabla I_{1st}) = \frac{|(v - v_{gt}) \cdot \nabla I_{1st}^{\perp}|}{|\nabla I_{1st}|}, \tag{5.5}$$

to see how effectively a method compensates for the aperture problem. The $\nabla I_{1st}^{\perp}$ is a vector perpendicular to the image gradient at the (common) origin of the computed and ground-truth vectors. Note that the component flow, i.e., vector collinear with image gradient, is often considered as a correct solution in situations with strong aperture effect.

Another way to accent evaluation of certain feature of an optical flow computation is to restrict evaluation spatially and temporally in the input sequence such as to regions of motion discontinuities or textureless regions [97]. Clearly, the default is to use the whole image. However, some methods, sort of, assign a confidence indicator to every computed flow vector. The purpose is to discount flow vectors from an evaluation that were, so to say, more guessed than computed. Thus, portions of more confident flow is only taken into consideration [72].

To complete the overview of possible approaches, we briefly mention the last two evaluation measures. An attempt by Baker *et al.* [97] was made just recently to establish a set of ground-truth benchmark sequences from vision, this is also called [103] the "Middlebury flow dataset"[1]. Four different aspects of movements were incorporated into the dataset. Besides the angular and endpoint angular error measures, they used cumulative histograms of errors for both measures as well as values at $50^{th}$, $75^{th}$ and $95^{th}$ percentile. Another aspect they focused on was the ability of computed flow fields to assist during interpolation between frames, in other words, the ability of computed flow field to predict intermediate frames. The ground-truth was not a flow field but this time it was an intermediate image that was originally part of the sequence and was omitted from it before an optical flow computation took place. A simple sum of squared differences was then applied on the flow-predicted and original intermediate frame. This is becoming increasingly interesting test for the next generation of view-based motion-compensated compression techniques [97]. Exactly the same idea was studied earlier by Lin and Barron [159] who were exploring the error associated with forward (the 1st frame is transformed onto the 2nd frame) and backward (the 2nd frame is transformed onto the 1st frame) transformations according to a given flow field. Their aim was to evaluate performance of given optical flow method on real data (with no ground-truth information) by transforming one input image according to the computed flow field and compare the transformation with the other input image using the RMS.

### 5.1.3 Obtaining appropriate ground-truth datasets

As most of the optical flow methods are originating traditionally from the field of computer vision, benchmark datasets from vision are still serving as a kind of "standartized" common ground to all researchers [96, 97]. The clear evidence for this is the popularity of

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

the Middlebury flow dataset[2], the popularity of the now-famous benchmarking sequences first used by Barron *et al.* [72] such as the Yosemite sequence (performance of modern techniques studied on this sequence even in 2005 [160]), or the translating and diverging tree as well as popularity of the cubes sequence by Otte and Nagel [158]. We have also occasionally used the well-known Hamburg taxi sequence, others did it as well [133]. Despite, researchers apply methods in different fields, e.g., for the cardiac motion estimation from 2D sequences as in [161], from 3D sequences as in [79] or for measurement of mitochondrial transport [162], to give only a few examples. And despite, they even further develop methods based on experience gained in their field [102]. This source of ground-truth datasets is useful especially when novel optical flow method is developed and should be introduced to the "optical flow" comunity.

However, it is clear that appropriate ground-truth dataset from the domain of live cell imaginery is required in order to responsibly select, develop, study and test applicability of any optical flow method for this field. This brings us to the question of how can we obtain such datasets. Naturally, the real acquired images do not have the ground-truth information included.

We reviewed, therefore, known solutions on obtaining ground-truth datasets and compared their main features in two tables. The fundamental difference between the tables is whether a method processes an existing real sequence and only adds the ground-truth flow field possibly in some automatic and unsupervised manner, see Table 5.1, or whether a method artificially creates a new image, in fact a sequence of such images, accompanied with the ground-truth flow field, see Table 5.2.

The first approach, Table 5.1, is typical for objects undergoing simple motions because complex motions are more difficult to recover. The process typically exploits certain special knowledge about the input real data which can't be generarily incorporated into any optical flow computation method. Note that to precisely recover motion is also the task of evaluated optical flow method. The scenario is such that the optical flow methods recover motion only to some certain extent because they are lacking this extra knowledge. Otherwise, if the process of ground-truth motion recovery would make it without the extra knowledge on any real input data, we don't have to search for another solution anymore.

We may always take the direct way, which is to prepare the ground-truth flow field manually. This is tedious, has low degree of reproducibility and is also rather errorneous [168]. The prepared flow field is even more unreliable when two 3D image stacks are paired. In fact, the 3D images must be paired at voxel level as a consequence of assigning a ground-truth flow vector to *every* voxel. This is extremly laboured and aggravated by inspection of 3D volumetric image on a 2D flat screen. A possible alleviation may be achieved by pairing only a few points and incorporating some, possibly elastic, transformation to compute smooth flow field, similarily to what McCane *et al.* [96] does. Theoretical possibility is to use a flow field computed by some other method as a ground-truth flow field. This, however, enables only to "tune" the developed method to work as good as the other method does leaving no room for improvement. However, Liu *et al.* [163] shows that for assessing ground-truth optical flow fields in vision images humans actually quantitatively achieve better results than state-of-the-art algorithms often do. As a consequence, their software offers means to correct computed flow field in order to turn it into a ground-

[2]Starting with only 5 compared methods in 2007, the quantitative comparison has already been made over 40 methods at the time of writing.

| Authors | Method & Images | Year |
|---|---|---|
| - | – pair of images is manually pixel-wise paired<br>– slow, tedious, unreliable | |
| Otte and Nagel [158] | – camera moves around a few cubes<br>– camera mounted on calibrated robot arm which provides precise camera coordinates and orientation<br>– static scene, only camera movements | 1994 |
| McCane *et al.* [96] | – camera moves around a few cubes<br>– utilizes properties of projective geometry and few manually established inter-frame correspondences<br>– objects can only be planar polyhedral<br>– limitations on configuration of objects in a scene<br>– static scene, only camera movements | 2001 |
| Baker *et al.* [97] | – indoor scenes with solid deformable (rubber) materials<br>– objects in a scene painted with fluorescent pattern<br>– test images captured in visible light<br>– flow field extracted from images captured in UV light<br>– dynamic scene, nonrigid motions, camera movements | 2007 |
| Liu *et al.* [163] | – semi-automatic annotation of any video sequence<br>– based on layered motion segmentation<br>– limited support for nonrigid motions | 2008 |
| Liu *et al.* [154] | – approximated GT only for road scenes<br>– expects zero roll and constant tilt of ego-vehicle | 2009 |

Table 5.1: Overview of approaches to obtain ground-truth datasets with real images

| Authors | Method & Images | Year |
|---|---|---|
| Barron *et al.* [72] | – picture of a tree transformed as a whole<br>– simulate camera movement normal/along to its line of sight resulting in translation of/"zoom" into the image<br>– the displayed reality is fixed in the sequences<br>– only smooth flow fields, no sensor noise | 1994 |
| Galvin *et al.* [81] | – man-made scenes: office and car on the street<br>– rendering scenes with modified ray-tracer Mirage<br>– ray-tracing allowed for discontinuities in the flow fields<br>– rigid motion, no sensor noise | 1998 |
| Mason *et al.* [164] | – present details on the approach of Galvin *et al.* | 1999 |
| McCane *et al.* [96] | – man-made scenes: car in the city<br>– follows up on the approach of Galvin *et al.*<br>– limited rendering (no storage for highly detailed texture and models, approximated simulation of physical lighting effects) due to capabilities of computes at that time<br>– rigid motion, no sensor noise | 2001 |
| Baker *et al.* [97] | – natural scenes: rocks with bush or tress, motion blur<br>– advanced rendering utilizing ray-tracer mental ray$^{TM}$ [165]<br>– image sequences show occlusion and large motions<br>– rigid motion, no sensor noise | 2007 |
| Baker *et al.* [166] | – natural and man-made scenes: a tree and city<br>– advanced rendering utilizing 3Delight renderer [167]<br>– nonrigid motion, occlusion, no sensor noise | 2009 |
| Hedborg and Forssén [156] | – present details on generating synthetic scenes with advanced lighting effects | 2008 |

Table 5.2: Overview of approaches to obtain ground-truth datasets with synthetic or simulated-real images

truth flow field. They use the well-established and high-ranking [97] method of Bruhn *et al.* [100]. Anyway, still the main drawback remains. And that is the fact that we are only annotating an existing real sequence. The first approach simply *do not generate* an artifical previously non-existing image sequence. For example, one can't test her method on image data showing an anticipated biological phenomenon.

The second approach, Table 5.2, is characteristic with evidently synthethic image sequences. In fact, the sequences display visually rather compelling content in the recent publications [166] but it still easily distinguishable from real images. We believe that the source of apparent deficiencies is in the demand for complex models. The world around us is complex, complex models are, therefore, necessary. But they are also too complicated to control resulting in incorporation of model simplifications. These are the clues that prevent generated images from perfection. Positive on this approach, however, is that it shoud be easy to *generate* as many and as long sequences as it is required. This aids in developing accurate and reliable statistics on behaviour of evaluated method. We refer to the first approach as to the generating pseudo-real image sequences.

Note on the subtle difference in terminology. It had been dealt with earlier in Section 3.1.3. In order to create pseudo-real image sequence, the ideas from computer graphics are always employed [81, 164, 156, 166]. In the computer graphics we often build up a virtual 3D scene with somehow placed objects. The situation is almost always represented with some 3D vector model of the scene with associated textures to faces that emerge in the model. This is where the overwhelming amount of parameters come from. A rendering algorithm is used to, let us say, convert the 3D vector model into 2D raster plane. The algorithm captures a snapshot of the scene at certain time instant. The situation in the scene slowly evolves meanwhile it is being regularily captured (rendered) by means of a sequence of 2D raster images, refer to Mason *et al.* [164] for nice illustrations. The motions in the 3D space between two consecutive time instants is represented with the motion flow field. The projection of the 3D space onto 2D imaging plane doesn't immediately yield the optical flow field. This is especially true when specular light is used in the scene. As noted in [164], the optical flow is a velocity field which transforms one image on the other, it is sensitive to apparent motion of brightness patterns. Contrast it to the motion flow field which truly represents the motion of objects regardless of lighting conditions in the scene. The computation of ground-truth optical datasets, therefore, introduce some bias in ground-truth flow fields. Since the dataset is generated by a machine, we expect the bias to be predictable. Despite that, the ground-truth flow field generally correspond with the displayed scene perfectly, even at sub-pixel accuracy.

## 5.2 On generating ground-truth datasets for microscopy

In the following text, we will focus on the approach of generating artificial pseudo-real ground-truth datasets for evaluation of optical flow methods on live cell images. The ground-truth flow fields should describe motion in test image sequences with sub-pixel accuracy. The approach should be able to generate 3D time-lapse sequences.

Unfortunately, we can't use any method of those summarized in the previous chapter exactly as it is. There are several reasons for it. First of all, none of the methods is directly capable of producing sequence of 3D images. But we may think of generalisation of some.

Anyway, aside of methods' technical limitations or incompatible assumptions as it is the case of robot arm [158], of painted objects [97], of the expectation of road [154] or just of the inherent simplicity of generated images [72], there exist a significant obstacle. It is the different understanding of the observed scene. In the computer graphics we usually assume that displayed scene is physically far greater than is the imaging device and, therefore, we assume the scene is also relatively distant in order to fit into a field of view. It is enough to look at the sort of generated objects in both tables. This assumption leaves a room for objects to appear at different physical distances from the imaging device, what, in turn, allows for occlusion or motion towards the imaging device. But more importantly, it leaves room for the difference between the motion flow and optical flow as it was discussed earlier.

If we greatly simplify the situation such that we could expect uniform ambient light with no specular reflections, that we could expect all objects in the field of view to be aligned at the same distance from the imaging device and, even more, that we could expect this distance to be so close and objects so small that we could assume parallel casting of rays from all the objects towards the imaging device, instead of the usual perspective geometry, then the motion and optical flows would be the same. In other words, the scene would be flat in depth with its normal parallel to the viewing direction, only objects' front faces would be visible without any reflections or shadows. In the microscope, as it was described in Section 2.1.2, the imaged objects are really tiny. Consider, for example, the one in Fig. 2.8 whose lateral diagonal is only 27.5$\mu$m long and axial dimension is not more than 5.0$\mu$m. It is very shallow in depth and, in fact, it is also very close to the microscope objective. Only stained objects, ideally, produce light that reaches imaging device, no considerable reflections occur. We finaly realize that, in this type of microscopy imaging, we may actually assume that motion and optical flow fields are identical. In the case of 3D images, which are essentially only stacks of 2D images, the situation is the same except that the third coordinate is added. Indeed, every 2D slice in the 3D stack represents content of single thin optical section whose normal is parallel to the optical path, i.e., the normal is in the axial direction. Sections are numbered and identified exactly with this third coordinate.

This finding, however, disqualifies any renderer designed for the computer graphics for our needs in microscopy. As a consequence, it also disqualifies all the remaining approaches in Table 5.2. It has only remained, from the methods in both tables, the approach of semi-automatic or manual annotation of time-lapse sequences. This is always a possibility. But, as mentioned earlier, this approach is limited only to existing sequences and is very tedious.

Another great issue is the generation of texture. The live cell microscopy images are characterstic with low SNR (signal-to-noise ratio), low contrast, they are rather faint monochromatic (without colours) typically with absence of sharp edges. These microscopy images are in all aspects in contrast to many man-made scenes or artificially generated natural sceneries — images for which most of the pseudo-real generators are designed to. In the same fashion, it may be rather difficult to construct sufficiently complex yet easy to control model for biomedical structures. For instance, when considering a model for the brighter (foreground) patches in Fig. 2.8 we may think of using spheres or ellipsoids until we notice the C-like shaped patch in the middle of Fig. 2.8B. The situation is again bit in contrast to scenes, for instance, with buildings or trees with many but still rather similar leaves. Fortunatelly, first solutions have already appeared [169, 170, 53, 171] and more is

expected to come. In our approach we have opted to, sort of, learn from given sample real image in order to mimic its texture. The learning is supported by layered segmentation (explained in the next paragraph). But to tell the truth, this learing is again based on some model of texture determined apriory and, therefore, its usage is limited. Theoretically, it should work well whenever the background object is displayed with intensities obeying unimodal distribution.

Moreover, it is not only to model shapes and textures when generating a sequence, we must also develop a model for motion. At least, this one seems to be easy because when displaying image from fluorescence microscopy we are actually displaying, again only in the ideal case, only the stained cellular structures. We refer to them as to the foreground objects. These displayed structures typically serve the same function and so undergo similar type of motion, e.g., translation, rotation, shrinking, splitting, joining, even no motion, or combination of some of them, etc. These movements are, however, relative to a cell as such. But the cell may move as well. If the staining of structures worked perfectly, we would see only them in the images without any contour of a cell. Unfortunatelly, this is rarely the case due to, so called, non-specific staining, see Section 2.2.1. The staining typically has the ability to delineate a cell contour creating effectively a mask of the cell in this way. We refer to this mask as to the mask of a background object, a mask for background. Its purpose is to define region on which to perform the movement of background, i.e., the motion of the whole cell. In the case of non-specificic staining missing in the image, we consider the background mask to spread over the whole image. Similarily, the foreground objects are identified in the image with mask for foreground. A two-layered segmentation is established in this way in which the background performs some global motion and in which the foreground objects perform exactly the same global motion plus their additional individual intracellular local motions. In the vision field, this could have been simulated with camera motion representing the global motion and with dynamic scene representing the additional foreground motions.

Towards this end, in order to create a new generator for ground-truth datasets in the field of live cell fluorescence microscopy we need to use different rendering technology and adjust models for shape and texture. In fact, no renderer is required because the motion and optical flows are the same. According to the recent publication by Svoboda *et al.* [171], the process of generating microscopy images can be splitted into three phases each being fairly complex. As a result, to control the whole process of generating image of particular cell, which is to apply the shape model and to generate the texture, is to supply it with many parameters. Many of these are related to the shape of the cell and its structures and to the way it is imaged in a microscope. Instead, we focus only on the control of both "background" and "foreground" motions. In our approach, the omitted parameters are replaced by supplying a sample real image. The generator then "tears" the image according to the background and foreground layers and forces the pieces to move in the image sequence. Of course, ground-truth flow fields are generated during the process as well.

## 5.3 The original publications

Now that we have explained motivation and presented background information on the process of automated generation of optical flow test images with ground-truth flow fields, we will briefly describe our related original publications [P5, P6]. The rest of this chapter will then cover selected topics from the publications in more detail as well as a more general concept for generating ground-truth test data (primarily) for live cell studies.

The first two publications are focused on the way to artificially generate a sequence of images, which are visually very close to given input (possibly real) image, and on the way to make its content appear moving. The first publication [P5] "On Generating Ground-truth Time-lapse Image Sequences and Flow Fields" is merely focused on some technical details regarding the creation of high fidelity images. The second publication [P6] "Pseudo-real Image Sequence Generator for Optical Flow Computations", on the other hand, is mainly focused on some technical details regarding the rendering of a sequence of such images while maintaing their high quality. Despite we regard the second publication as a continuation of the topic of the first publication, they both had to treated as, say, stand-alone documents and as such they have certain overlap in common.

The third publication [P7] "Estimating large local motion in live-cell imaging using variational optical flow" demonstrates rather good performance of modern differential methods on live cell time-lapse images. In spite of that the optical flow computation is the main topic of the publication, we have included it into this chapter because the accuracy measurements in the publication were conducted on artificially generated ground-truth datasets. In fact, the existence of the generator was crucial for this publication and also for the other one [P4].

## 5.4 Additional notes to the original publications

### 5.4.1 Forward versus backward transformation

The key element in our approach that performs motion of image regions is the image warping, in particular the backward transformation.

In ground-truth datasets we have typically several frames, i.e., images captured at consecutive time instants, with flow field associated to every pair of consecutive frames, one field to one pair. Considering a pair, we talk about the first (earlier) and the second (latter) frame. Using this terminology, every flow field vector $v_{gt}(\mathbf{x})$ at position $\mathbf{x}$ describe the shift of intensity $I(\mathbf{x}, t)$, found at coordinate $\mathbf{x}$ in the first image captured at time instant $t$, to the new coordinate $\mathbf{x} + v_{gt}(\mathbf{x})$ in the second image, i.e., it should hold

$$I(\mathbf{x}, t) = I(\mathbf{x} + v_{gt}(\mathbf{x}), t + 1). \tag{5.6}$$

Naturally, one would probably preferr to construct the second image at $t + 1$ from the first one at $t$, i.e., in the *forward* direction. It really seems easy: all we need is to grab a pixel value from the first image and put it at appropriate position (a vector's end) in the second image. But the opposite is true, especially when flow vectors contain elements from the real domain. In the case that $v_{gt}(\mathbf{x})$ containes real-valued elements, the coordinate $\mathbf{x} + v_{gt}(\mathbf{x})$ will fall off the pixel grid. Some interpolation technique must be used to interpolate on (nearest?) integer-valued coordinate. In fact, the forward transformation

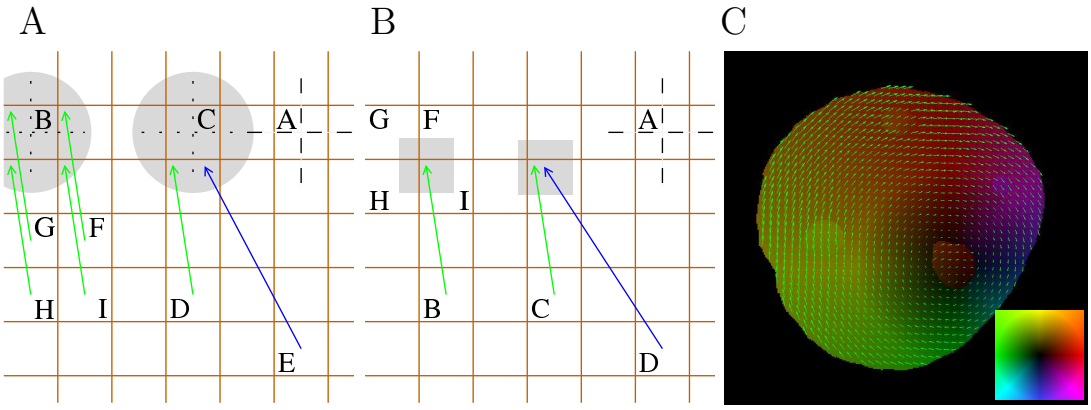


Figure 5.1: Illustration of the principle of the forward and backward transformations in A and B, respectively, when interpolation must be used. During the forward transformation, vectors originating from pixels G,F,H and I had to be sought out to enable computation of pixel B. Notice that vectors contributing in the interpolation may be originated from relatively distant coordinates when a flow field is not smooth enough. In the same fashion, two or more vectors originating from relatively distant coordinates may "fetch" the same value in the backward transformation in B. In C, example of colour-coded flow field showing clockwise rotation of the background object/cell. Every flow vector is described with colour pixel; the colour codes direction whereas the intensity codes magnitude of a vector. Figures A and B reprinted from Ulman and Hubený [P6].

is approached by processing every pixel, i.e., iterating over all valid pixel coordinates, of the second image. For each pixel, several vectors that *end* nearest to the processed coordinate are sought out. A pixel value in the second image is interpolated from the first image's values at *beginnings* of the respective vectors, see Fig. 5.1. This introduces time complexity of $\mathcal{O}(n^2)$ of the forward transformation where $n$ is the number of pixels in an image. We may only modify the "implementation constant" by changing the interpolation technique, which changes the number of nearest vectors that must be sought out, and by narrowing the search region, which changes the number of vectors that must be evaluated during the seek for nearest ones and also which changes the limit on maximum length of detectable/usable vector. If a vector is long enough, it's beginning will never be close enough to any given coordinate and the vector will be always disqualified from the seek. Consider, for instance, a flow field for 2D rotation around centre of image, see Fig. 5.1C, where vectors further from the centre are longer than vector closer to the centre. To keep the transformation general, no assumption on the flow fields must be made. This enforces, above all, to keep the search region rather large what renders the method considerably slow.

The backward transformation creates the first image at $t$ from the second one at $t+1$, i.e., in the *backward* direction. To establish a value at coordinate $\mathbf{x}$ in the first image, one has to look into the second image for pixel value at coordinate $\mathbf{x} + v_{gt}(\mathbf{x})$. In case of real-valued flow vector, some interpolation technique is used on values from the second image. The time complexity is asymptotically linear. The "implementation constant" now depends only on the number of coordinates one has to visit during the interpolation.

It is noteworthy that any of the two transformations don't perform without errors. The problem is intrinsically in the use of real-valued flow vectors that push pixel values off

the pixel grid. Interpolation techniques must be used in both cases to find values at the integer-valued grid coordinates. This essentially influences the computed value. Based on results of Lin and Barron [159], it appears that the best solution with smallest RMS errors is to use the backward transformation with bicubic spline interpolation. It performed equally well as the forward transformation with forward displacement interpolation with the exception in that the forward version has worse time complexity as discussed above. We have opted, therefore, for the faster backward transformation. In addition, as the RMS error rates reported in their publication for the backward versions were rather balanced, i.e., the difference in performance between interpolation techniques used with the backward transformation was not greater than 1.3 of intensity points, we have opted to use simplier and a bit faster bilinear (for 2D and bicubic for 3D frames) interpolation technique in our generator. The level of noise or of disturbing non-specific background staining is typically higher than 1.3 of intensity points, see the magnitude of variations in intensity profiles in Fig. 2.8A,D. On top of it, our approach assures that sample input image is transformed exactly one time to create a image/frame in the generated sequence. The amount of error due to transformations is kept as low as possible in this way.

Also note that Lin and Barron call these the *reconstruction* techniques whereas we call them the *tranformation* techniques. This difference in names is due to the difference in the way we use the techniques. While their original aim was to warp, say, the first image with the forward reconstruction technique according to a computed flow field so that the warped image should, in the event of correct or close-to-correct field, resemble the second image. They measured the difference between the *reconstructed* and the original one to judge on quality of the computed flow field. Our aim is to change the sample input image according to some flow field simply to create a next new image in the sequence rather than approximating some existing one. Hence the label transformation was adopted.

### 5.4.2 Piece-wise smooth flow fields

The use of interpolation in the backward transformation has introduced a few issues into the process of generating image sequence. When generating, the sample input image should be, sort of, iteratively transformed to produce next frames in the sequence. In our generator, the iterative transformation of frames is replaced with iterative concatenations of some helper flow field. This field describes how to change the sample input image to create currently processed one with the least number of transformations possible, i.e., with only one. Figure 5.2 demonstrates the rate of degradation of transformed image after only a few iterations. The degradation is a result of the bilinear interpolation, which is essentially nothing but the weighted average of four neigboring pixel values. The more different the four pixel values are, the more prominent the averaging becomes with every next iteration — until the transformed image is considerably smoothed. On the other hand, if the four pixel values were all the same, the interpolation would perform well. Clearly, we can't insist on, at least piece-wise, smoothness of input images. We rather demand it for the created flow fields.

The demand for piece-wise smooth flow fields that would be used with the backward transformation is twofold. Firstly, we use the backward transformation during the concatenation of two flow fields and so we would like to limit errors produced by the interpolation. Secondly, as it was demonstrated in our publications, flow field with motion boundaries

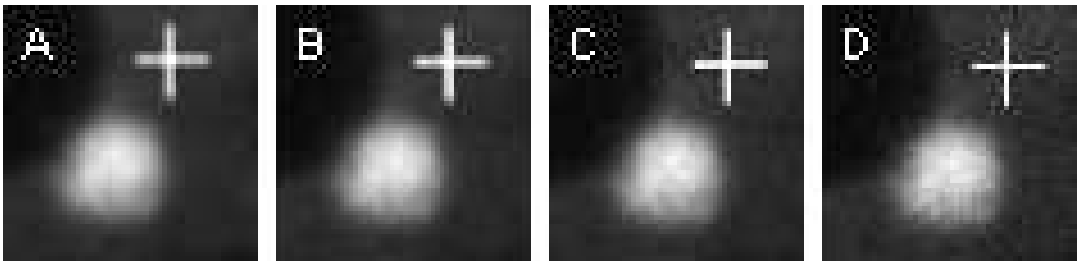


Figure 5.2: Degradation of a region of an original input image with artificial white cross added, in D, after it is iteratively transformed with flow field corresponding to the translation of (1.3,0.5). Images "translate" to the right and slightly down. Note that the image in A, which is after the 3 transformations, actually starts this 4 frames long sequence. This is a feature of the backward transformation which "generates from the end". The line width of the cross is 1px in the original image. All images were enhanced for printing purposes.

produces artifacts irrelevant of whether backward or forward transformation is used and of what interpolation technique is used.

Why do we need the backward transformation for concatenation of two flow fields? In the forward sense, we would like to transform given image first according to flow field $v_1(\mathbf{x})$, where $\mathbf{x}$ is coordinate within the image, and afterwards the result further transform according to $v_2(\mathbf{x}'), \mathbf{x}' = \mathbf{x} + v_1(\mathbf{x})$. We aim to compute flow field $v(\mathbf{x})$ that produces the same final transformed image, see Fig. 5.3. The same thing said but in the backward sense, the $v(\mathbf{x})$ should fetch pixel intensity from the same coordinate $\mathbf{x} + v(\mathbf{x})$ from which it would be fetched by (in the backward sense) the first transformation $v_2(\mathbf{x}')$ and stored temporarily at coordinate $\mathbf{x}'$, from which it would be fetched by the second transformation $v_1(\mathbf{x})$. We resolve:

$$\mathbf{x} + v(\mathbf{x}) = \mathbf{x}' + v_2(\mathbf{x}') \qquad \text{and} \qquad \mathbf{x} + v_1(\mathbf{x}) = \mathbf{x}', \tag{5.7}$$

$$\mathbf{x} + v(\mathbf{x}) = \mathbf{x} + v_1(\mathbf{x}) + v_2(\mathbf{x} + v_1(\mathbf{x})), \tag{5.8}$$

$$v(\mathbf{x}) = v_1(\mathbf{x}) + v_2^{\mathrm{BackTby}v_1(\mathbf{x})}(\mathbf{x}), \tag{5.9}$$

from which we see that concatenation can really be conducted as sum of the $v_1(\mathbf{x})$ and backward-transformed $v_2(\mathbf{x})$. The $v_2^{\mathrm{BackTby}v_1(\mathbf{x})}(\mathbf{x})$ is the vector $v_2(\mathbf{x}')$ backward fetched by the vector $v_1(\mathbf{x})$ so that it appears at the coordinate $\mathbf{x}$. As Fig. 5.3 suggests, if a flow field is smooth enough (to avoid interpolation effects) and is backward transformed according to another smooth enough flow field (to avoid transformation artifacts), the transformation error shouldn't be much in effect. For instance, flow field showing some translation is constant, i.e., all flow vectors from the field are the same. Concatening two such flow fields can't produce any error. Currently, we provide the generator only with flow fields that represent translational and rotational motions.

The use of smooth flow fields prevents from artifacts in transformed images such as the "copy" effect, examples were given in our publications. In order to be able to provide ground-truth datasets with motion boundaries, i.e., with distinct flow patches of sharp border as in Fig. 5.1C or as in Fig. 5.4C without the "copy" effect, we have proposed to split the simulated motion into two layers: the (bottom) layer with (global) motion of the background object and the (upper) layer with (global+local) additional motions

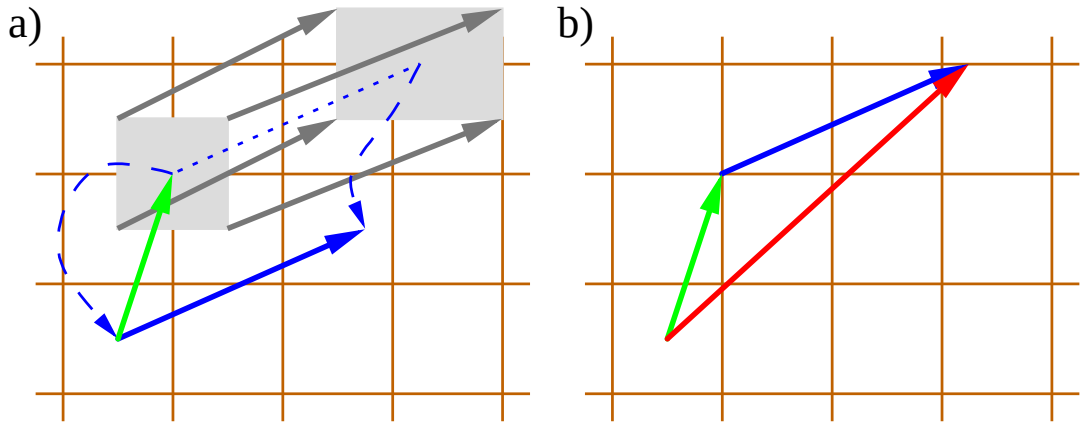


Figure 5.3: The principle of concatenation of two flow fields, a case study. Note that in our generator we are using the backward transformation which, basically, fetches pixel from vector's end to vector's beginning. In b), the first image transformation would be according to the vector $v_2(\mathbf{x}')$, drawn with the blue arrow, whose result would be transformed according to the vector $v_1(\mathbf{x})$, drawn with the green arrow. The result of concatenation is the red vector, $v(\mathbf{x})$, which is a result of addition of the green and blue one, if the blue one would be translated to the green's beginning. In a), the blue vector is fetched to the origin of the green one's beginning which is similar to what the backward transformation does with pixel values. If the green vector is real-valued, we need to interpolate. The four gray vectors would be involved in this case.

of foreground objects. We utilize separate helper *smooth* flow fields for the background and for each foreground object, Fig. 5.4B, that are updated before generation of every next image/frame. For example, the current background flow, the one relevant to the currently processed pair of the generated sequence, is concatenated with every helper foreground flow field to yield a new helper flow field. The sample real input image is then transformed individually to give images of foreground objects at updated positions, foreground objects are extracted and inserted into the generated background. Note that these helper foreground flow fields are kept as small as possible in order to keep the memory consumption low, Fig. 5.4A. This is especially critical when generating ground-truth datasets with 3D frames. Similar process happens with the ground-truth flow field relevant to the currently processed pair.

### 5.4.3 Supporting simulated coherent motions

Based on our observation of real time-lapse image sequences, cells or their intracellular structures tend to change velocity or direction of motion rather slowly provided the sequence was acquired with reasonable temporal delays. This can be accounted for the purpose with which biologists acquire such sequences because they typically take some action on a cell, mostly some infection, and they want to observe its reaction. The reaction then appears as a, sort of, controlled or preprogrammed motion, e.g., increased synthetication and transport of some proteins. But if one is to display motion trajectories, the lines typically exhibit small perturbations. This is also evidenced in various publications on tracking in time-lapse microscopy [15, 172, 173, 17, 174] and recently by Jan Hubený [47] who was comparing tracking results on real sequences and sequences artificially generated

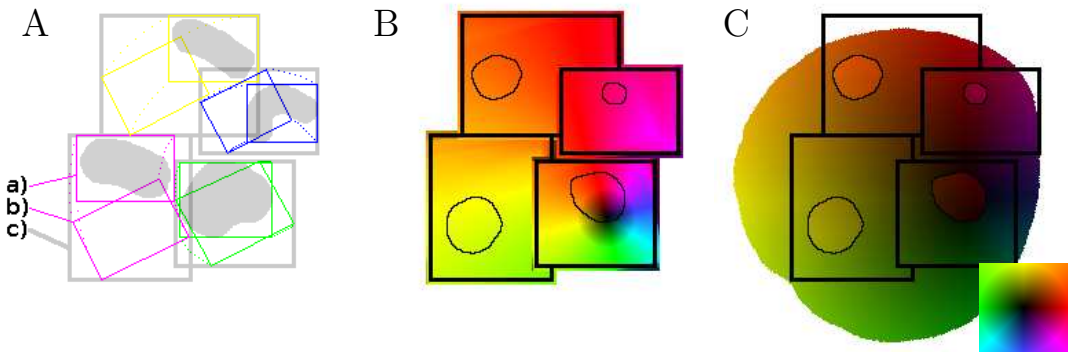


Figure 5.4: Example of local foreground 2D flow fields. Minimum local foreground region, denoted as c) in A, is determined to contain the whole local foreground flow field of given component. Basically, the mask of possible positions is split into connected components, a), and these are iteratively transformed according to the given background flow, b). In this particular case the rotation was used. The foreground flow fields, in B, are inserted into a given ground-truth flow field, in C. Clearly, only the portions corresponding to foreground objects are inserted. The colour encodes flow vector: hue tells direction and higher intensity signifies greater magnitude.

from these with our generator, Fig. 5.5.

We have, therefore, tried to implement this kind of motion, during which an foreground object seems to travel from point A to B while its route is not exactly straightforward. This means that any foreground object, based on its mask of possible positions, randomly chooses some direction and velocity at the beginning which it then tries to follow in a few consecutive frames in the generated sequence. Meanwhile, devitations in both terms are allowed. Sometimes greater deviation is forced by the mask of possible positions, e.g., when an object moves into a corner.

Technically, this is driven by two parameters of the generator. The first one is the maximum travelled inter-frame distance, given as a number of pixels per frame. One number is valid for all foreground objects. The second parameter is a mask of possible positions, which is expected to always include the mask of foreground objects. It is further expected that the foreground objects will always remain within this mask. In other words, an inversion of the mask of possible positions defines pixels, respective pixel coordinates, which are prohibited to become part of any foreground object throughout the generated sequence.

For the implementation, we make use of probabilistic decision maps. It is simply a square image (for 2D frames and a cube for 3D frames) whose edge is twice the maximum travelled distance number, Fig 5.6C. Every pixel coordinate in such image defines particular direction vector after subtracting the coordinate of image centre. Every pixel intensity defines chances of the associated vector to define direction of the upcoming movement. Example of its performance is given in Fig. 5.6.

It must be noted that the decision maps are used only for the local additional translational movement of foreground objects. The support for rotation, which was implemented just recently, is rigously driven by values found in the mask of possible positions, no fluctuations are allowed. This will be covered in the next section.

The generator, at the time of writing, also does not incorporate zero-mean Gaussian fluctuations in any of the global movements. For instance, if a cell is supposed to rotate by

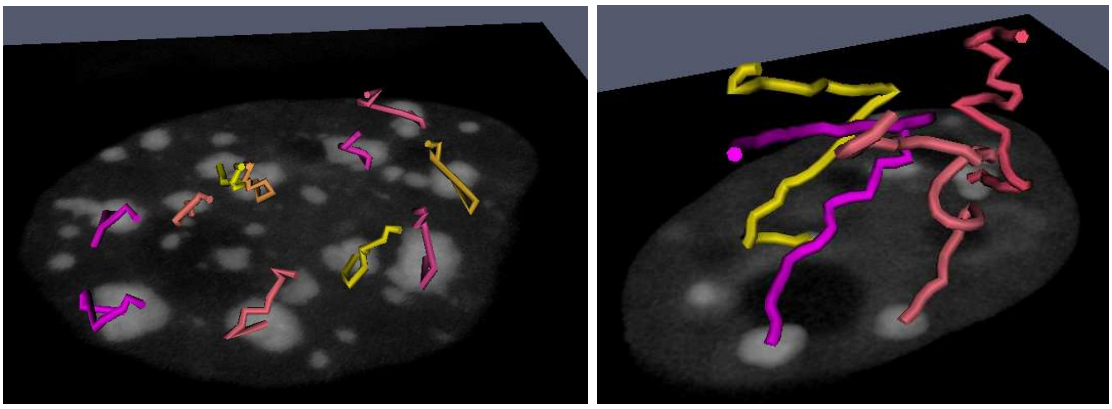

Figure 5.5: The visualization of trajectories of selected intracellular structures, HP-1 protein, within the same type of cells, HL-60 cells. The left image shows trajectories detected in real data, only 9 frames were available. The right image shows trajectories detected in a generated sequence, 50 frames were generated. No motion of the background object, i.e., the cell, was simulated. All images were enhanced for printing purposes. Reprinted with permission from Jan Hubený's dissertation [47].

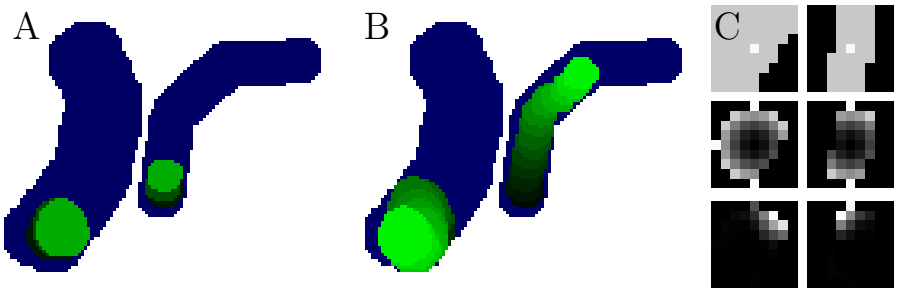


Figure 5.6: An example of development of positions (in green) of selected 2 foreground regions within the mask of possible positions (in dark blue) after only 3 frames, in A, and after 13 frames, in B, were generated. The brighter the green colour is, the more recent the position is. The three rows in C show maps for decision support of a direction of next movement of some foreground object after the 3rd frame was generated, i.e., the situation in A. Left (right) column is valid for the left broader (right narrower) region. Each pixel in the map determines a unique movement vector, pixel intensity determines the probability of this vector to be chosen, lighter means more probable. The determination starts off the top row where possible movements are outlined. The decision maps are changed, in the middle row, after involving the length of possible vectors and after taking into account the previous direction, in the bottom row.

some constant angle from frame to frame, it rotates exactly this angle from frame to frame without any random devations. Anyway, it is a simple matter to additionally change this behaviour. The generator, however, supports for additive zero-mean small-sigma Gaussian noise on flow fields. The rotation can then become a bit nonrigidly distracted in this way. Currently, the selection for this feature is driven at compile time.

### 5.4.4 Controlling the generator

We have noted earlier our aim to equip the generator with simplified control over the way it generates ground-truth datasets. One such action taken towards this aim was to design the generator to make use of user-supplied sample real image instead of many quantitative parameters describing what should appear in the generated sequence. This concept has pulled in the use of foreground and background mask images. Introducing another mask image, that simply shows where the foreground objects are allowed to appear while they are moving in the sequence, was a next straightforward logical step towards the given aim.

Allow us to summarize all controls the generator understands at the time of writing, they all can be either 2D or 3D:

| | |
|---|---|
| **A,** | sample real input image |
| **B,** | (global) inter-frame translational vector for the background object |
| **C,** | coordinate of centre of (global) rotation of the background object |
| **D,** | angle of (global) inter-frame rotation around the $z$ axis of the background object |
| **E,** | mask image showing where the background object is in the sample input image |
| **F,** | maximum allowed (local) inter-frame distance of any foreground object |
| **G,** | mask image showing where the foreground objects are in the sample input image |
| **H,** | mask of possible positions of the foreground objects |
| **I,** | multiplier used when reading (local) orientation angles |
| **J,** | length of the sequence to be generated |

Many of the controls have been introduced earlier in the text. Still, we will allow our selves to summarize and provide a brief comment on these. The sample real input image, **A**, is expected to be a real acquired time-lapse fluorescent microscopy image. It is this image that is subject to separation into two motion layers and that should appear moving in the generated sequence. The content of the two layers is given with the mask images, **E** and **G**. It is expected that one mask, **E**, delineates a cell while the other mask, **G**, delineates stained intracellular structures, hence **G** shoud be subset of **E**. The content of the cell mask, **E**, is artificially generated based on values found in the sample real image, **A**. The content of the structure mask, **G**, is always a transformed copy from the sample real image, **A**. Unfortunatelly, the structures, let us say, hide the cell underneath them. We must be able to fill in the hole that appeared after a (foreground) structure has moved elsewhere. That is why the (background) cell is generated while the (foreground) structures don't have to be. This easily assures high fidelity of the foreground layer in the generated sequence.

The motion of a whole cell, we call it the (global) motion of the background object, is given as a composition of translation, **B**, with rotation around $z$ axis, **C** and **D** (even

in the 3D, see Section 2.2.2 on page 17 for discussion). As mentioned earlier, this is an exact motion regularily occuring between any two consecutive frames in the generated sequence. The cell structures, denoted as the (local) foreground objects, undergo the same global movement between two frames. During which the mask of possible positions, **H**, is moved as well so that its relative establisment within a cell is kept fixed all the time. The foreground objects are allowed to undergo additional composition of translational motion up to some constant number of pixels per frame, **F**, with additional rotational movement. The direction of translations is driven by the probabilistic decision maps explained in the previous section as well as by the mask of possible positions, **H**.

The additional rotational movement of foreground objects is determined from a pixel values in the mask of possible positions, **H**. Every foreground object exactly as it is found in the original real image is said to be at its local orientation 0°. When a new frame is generated, the centre of mass of the foreground object at its current position is computed and a pixel value from the mask of possible positions is read at this particular coordinate. The obtained pixel value encodes new orientation that we desire for the foreground object to show over here. If there is a difference between the desired and current orientation, the object is rotated. If we were to wish to simulate a forward movement of some intracellular structure along an arc, we would split the arc into several passages. Each passage would encode orientation under which the structure orientation would appear tangential to its motion, Fig. 5.7A–C. The desired local orientation is encoded as $(O - R) * \mathbf{I}$ where $R$ is a reference value for orientation 0°, $\mathbf{I}$ is the multiplier to fine control and $O$ is the pixel value found in the mask image. This was adopted only for the reason that we use mask images with 8bit pixel depth. Thus, they have only 255 useable pixel values (values from 1 to 255 indicate an interior of a mask, 0 indicates outside a mask) to map an interval of orientations $\langle -180°, 180° \rangle$. We use $R = 100$ and $\mathbf{I} = 1.0$ to easily construct the mask images and since we think the interval $\langle -90°, 90° \rangle$ for inter-frame rotation sufficies. By setting $\mathbf{I} = 0.0$ we may disable the additional rotations. In the feature, we can adopt the similar idea and define a certain pixel value that would mark a centre of rotation in the mask of every foreground object. Currently, we feel that the centre of mass of an foregound object is an anticipated point to rotate around.

A shortcomming of the current design is that it does not support a constant rotation throughout the generated sequence of otherwise stationary, non-translating, foreground objects. When a foreground object can't change position of its centre of mass, it then can't read other rotation angle from the mask of possible foreground positions other than the one it currently reads. It is a question of design or, perhaps, of the way we may interpret the foreground rotation angle that we extract from the mask image. We may easily assume that the angle would actually tell by what angle shall we further rotate the foreground structure from the orientation it poses at the time, the angle acts as a command "always rotate by". Current assumption is that the angle tells what the local orientation should be, the angle acts as a command "assure the orientation is".

The generator was designed to require only a basic information of what is where and how it should look like, refer to Fig. 5.8 for some simple yet powerful masks. It then creates ground-truth datasets autonomously with random simulated motions based on our observations and experience. However, we may always take over the control and rather precisly specify what, when and where is going to be simulated in the generated sequence. Note that this also includes a bit more of "labour" preparation.

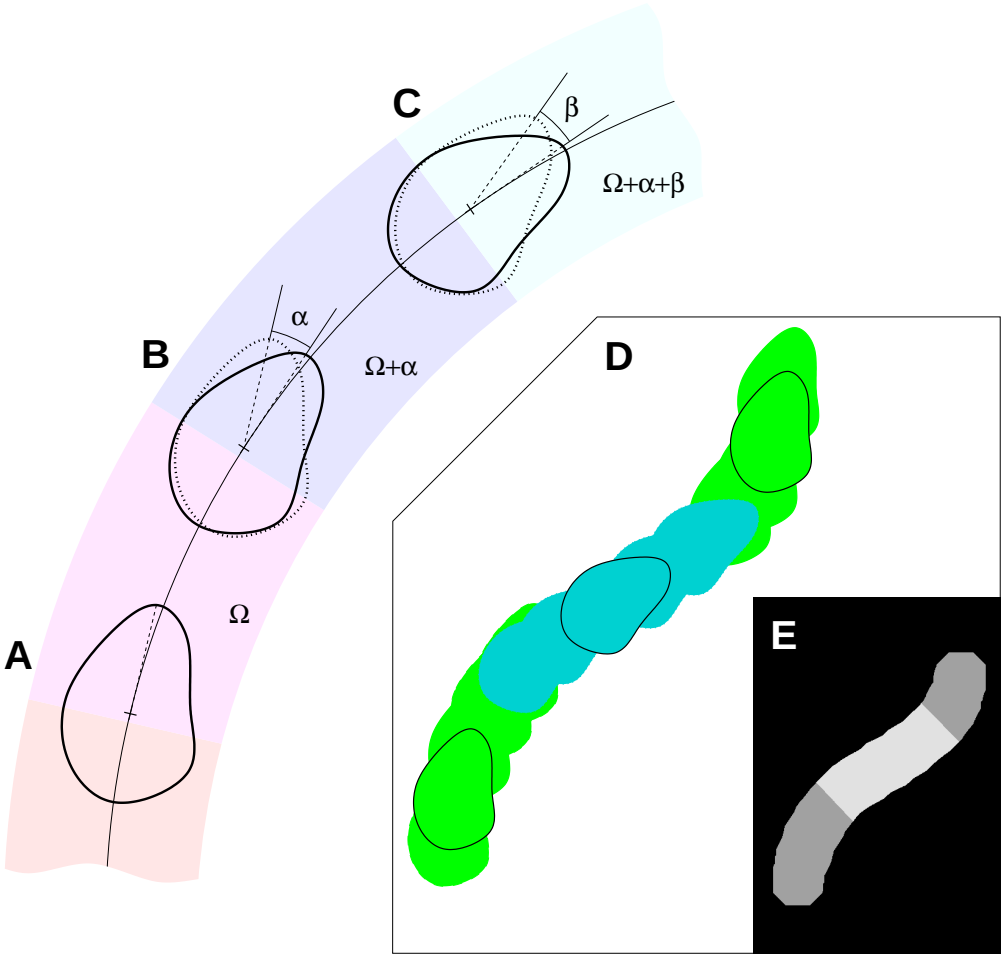


Figure 5.7: Illustration of how to setup the mask of possible positions so that it enables an appealing forward movement along some curvature. Suppose we have the foreground object at position A with its local orientation $\Omega$. The object translates up along the arc until it reaches position B where it is pictured with dotted line. At this position we, as designers, decide that if the translation along the arc would continue with the same local orientation, the movement would look less naturaly. Hence, a new passage is started here when local orientation is forced to be $\Omega + \alpha$. The generator rotates the object by $\alpha$, shown with solid line, when its centre of mass reaches this passage. We proceed similarily at the position C. In D, a preliminary mask of possible positions is being created by composing together mask of the foreground object at positions where we wish it would appear. We also take into account its local orientation. In E, such manually created mask was automatically prepared for the generator. The middle brighter passage encodes new local orientation.

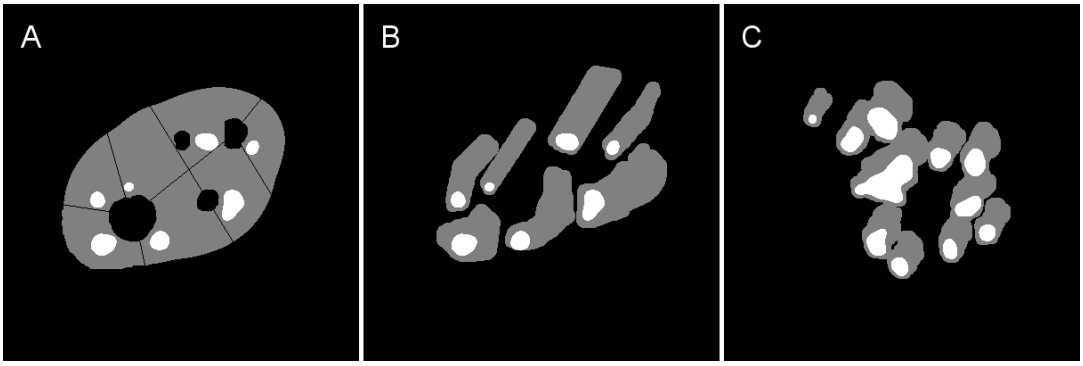


Figure 5.8: Examples of masks of possible positions (darker) overlaid with masks of foreground objects (lighter). A copy of the mask of background object was simply partitioned, in A. Though this only prevents objects from mutual collisions, moving out of a cell or moving into other cell organs (the four black spots), this often give rise to nice ground-truth datasets. Simple nearly geometric shapes were used to push foreground objects to move within a cell, in B. A method of copying and composing together the masks of foreground objects can be used as well, in C.

Returning to Fig. 5.7, we may wish to simulate movement along certain path. We give here a short suggestion on how to achieve it by means of creating the mask of possible positions for this object. Every foreground object has to be treated separately. We start with its mask, **G**, which outlines the object, and we compute its centre of mass and, perhaps, draw a line from it in arbitrary direction, just like in Fig. 5.7A. We then make copies of the mask and compose them together along the direction the object is supposed to move. We may reach some corner point and rotate the object there. We note the position (coordinate) of the centre of mass and the angle, by which the object was rotated, every time we reach such corner point. The line may be advantageous for measuring the rotation angle. Eventually, we arrive to a composition of masks, may be similar to the one in Fig. 5.7D. Problem with this mask is that it is too tight to let the object move. Recall that we are constructing a mask of possible positions. Hence the foreground object must always appear within this mask. But the mask shall not be too loose. The object movement could otherwise be disturbed by many various possible directions other than the expected one. In fact, we aim to smooth the boundary of the composition. Many solutions may be applicable. For instance, one may split the composition into many direct segments, compute convex hulls of them and compose them together. For the figure, we opted to compute morphological sceleton and dilated it with a circular element of appropriate radius. The dilation slightly widened, by 3–5 pixels, the original composition. Once we arrive to the smoothed composition, we colour-code the angles of local orientation, Fig. 5.7. Note that this can currently be done only with foreground objects. The background object undergoes regular motion, defined with parameters **B**, **C** and **D**, during the whole sequence.

### 5.4.5 Examples of generated sequences from the live cell imaging

We present here three examples of generated sequences: 2D and 3D examples and an example of controlled simulation.

In Fig. 5.9, a case study of a 2D generated ground-truth dataset is present. Eleven

frames were generated while only the 1st, 4th, 7th and 10th are shown, in E, F, G and H, respectively. The generator was supplied with the sample real image, in A, and the three masks, shown in overlay in B. The dark gray, light gray and white represent the mask of background object, of possible positions and of foreground objects, respectively. The sequence underwent 3° per frame global counter-clockwise rotation with additional local movements of up to 5 pixels per frame. A grid, undergoing the same global motion, was overlaid on the images so that local movements can be observed better. Local movements are also demonstrated with the composition of positions of foreground objects throughout the sequence, in C. The lighter the mask is, the earlier position it represents. A ground-truth flow field, shown in D, associated with the 10th frame is colour-coded, the legend is given in the inset. In this sequence, we tried, by appropriately adjusting mask of possible positions in the circle in B, to push the two foreground patches to move along each other and disjoin later. The two patches appear to move together in the early frames, in E and F, until they started, in G, to move apart.

For the 3D case, only a comparison of two consequent frames is present to illustrate local foreground movements in space, Fig. 5.10.

And finally, an example of controlled simulation of a movement of two cells is given in Fig. 5.11. This last example is particullary interesting in that it simply degraded the two-layered concept because it set a dark empty whole image as the background object (the degraded layer) and used the whole cells as the foreground objects. As a result, additional local movement of intracellular structures were not possible. However, our aim in this study was to generate dataset where two cells move close to each other and touch. We also observe in real data, Fig. 5.12, that in this particular case the local movements are very small compared to the motion of the whole cell and, as such, need not be simulated explicitly.

## 5.5 Summary and future directions

### 5.5.1 The developed generator

We will summarize the main features of the developed generator in this section. In the next section, we will present the concept of, what we call, the universal generator of optical flow ground-truth datasets for live cell studies. Since the later is merely an extension of the former, i.e., the current implementation, in the following we will often refer to Fig. 5.13 as well in order to easily identify the extensions later.

We believe there are several good points on the current approach. Above all, it is the utilization of the backward transformation that changes image data based on a given flow field. From the definition of the transformation we see that the flow field immediately becomes a ground-truth flow field, eq. (5.6). It also easily fulfills the constant brightness assumption, eq. (3.8) on page 27. Regarding the backward transform, we have developed a solution that transforms the original input data exactly once, allowing for preservation of its original resolution as much as possible.

This is achieved by utilizing several flow fields, shown with dashed rectangles in Fig. 5.13, that link the sample real image with the currently created frame, by developing a concatenation scheme for flow fields and by insisting on smoothness of these flow fields, which, in turn, allows us to keep the resolution of flow fields reasonably high even

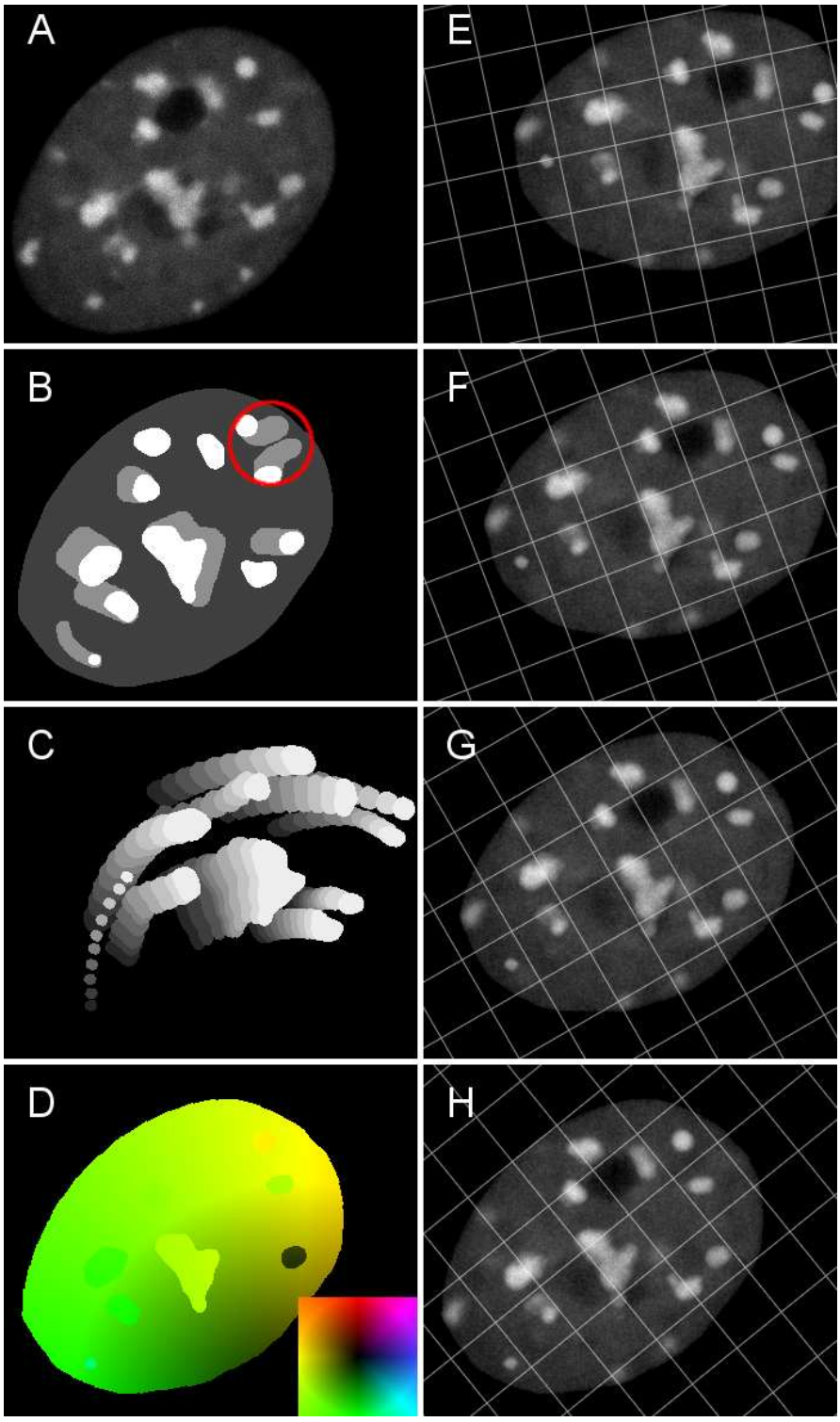


Figure 5.9: Case study on control and generated ground-truth dataset. The caption is given in the text. All images were enhanced for printing purposes.

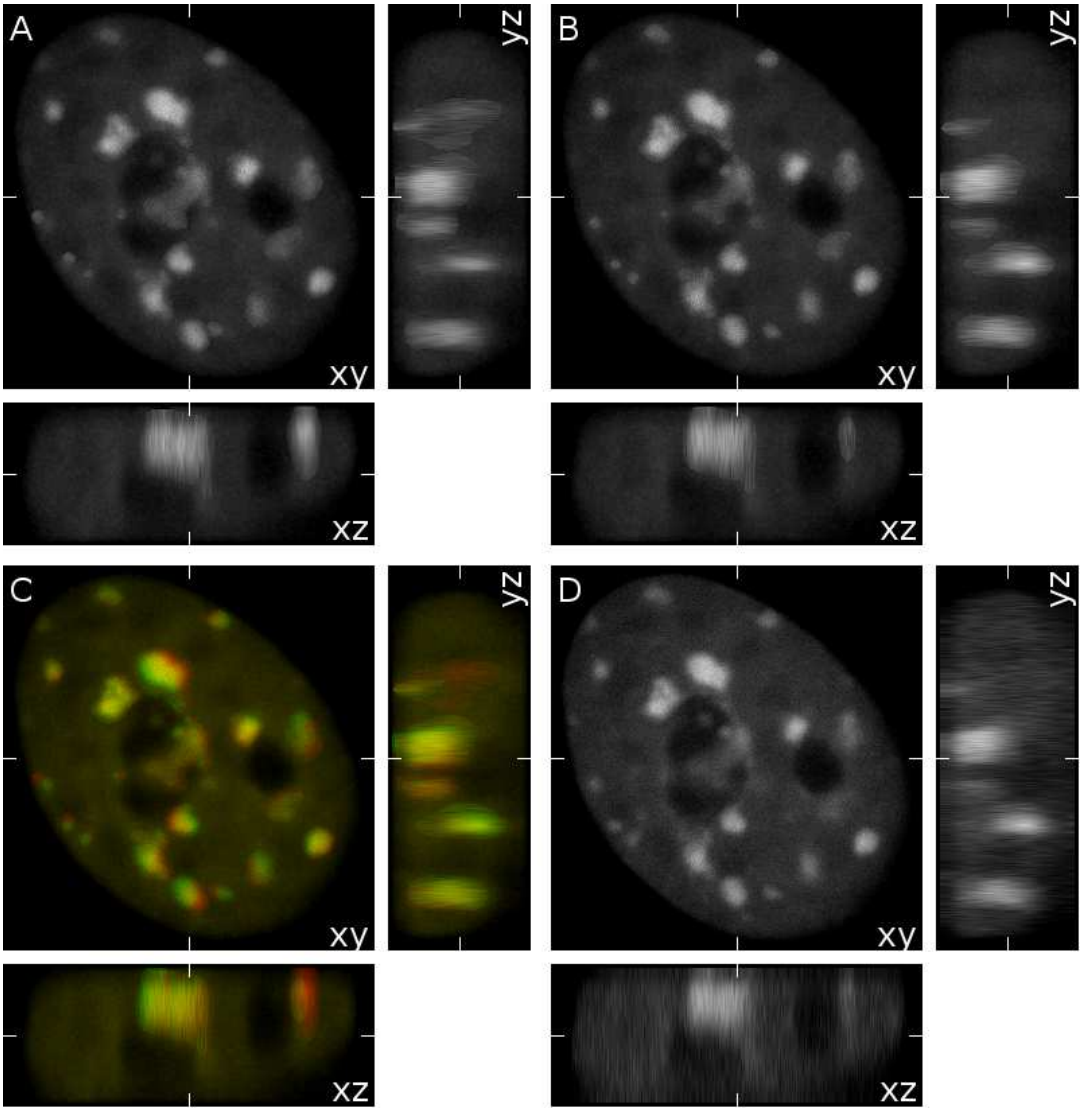


Figure 5.10: Sample cross-section of two consequent frames of a 3D example sequence, in A and B. In C, composition of two consequent frames, the former is displayed in the red colour channel while the latter is in the green channel. Visualisation of local movements can be achieved in this way. A 3D sample real image, on which the sequence was based, is shown in D. All images were enhanced for printing purposes.

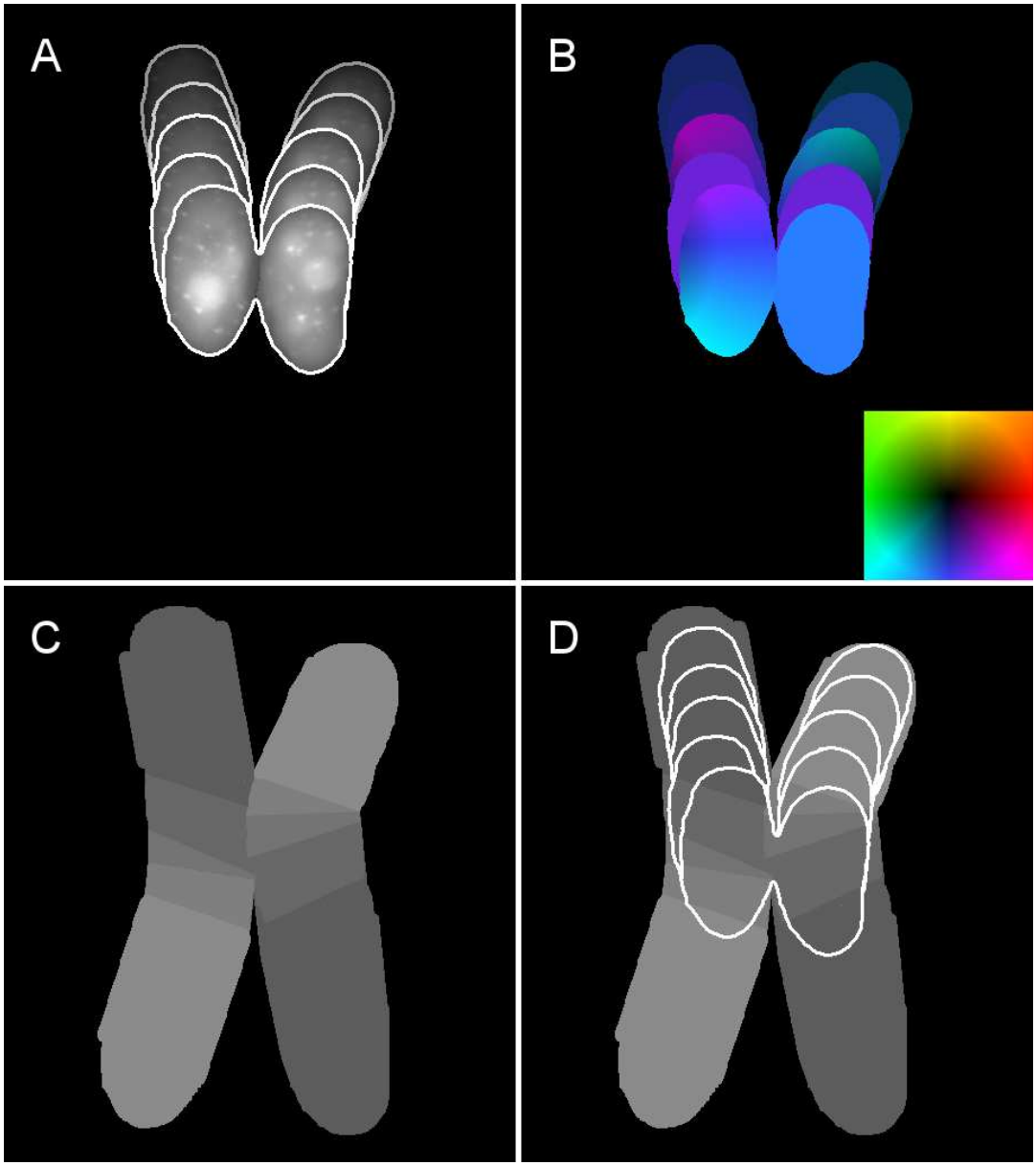


Figure 5.11: Example of tightly controlled grund-truth dataset generation. Every 4th frame of the first 13 frames from an original sequence is shown, in A, with artificially added white contours, darker image is earlier image in the sequence, and with associated flow fields, in B. Note that some flow fields are constant, i.e., the same colour is over the whole cell, denoting purely translational motion. The others describe composition of translation and rotation. The mask of possible positions alone and with overlaid contours is shown in C and D, respectively, so that we can see why the cell got rotated at certain frame. All images were enhanced for printing purposes.

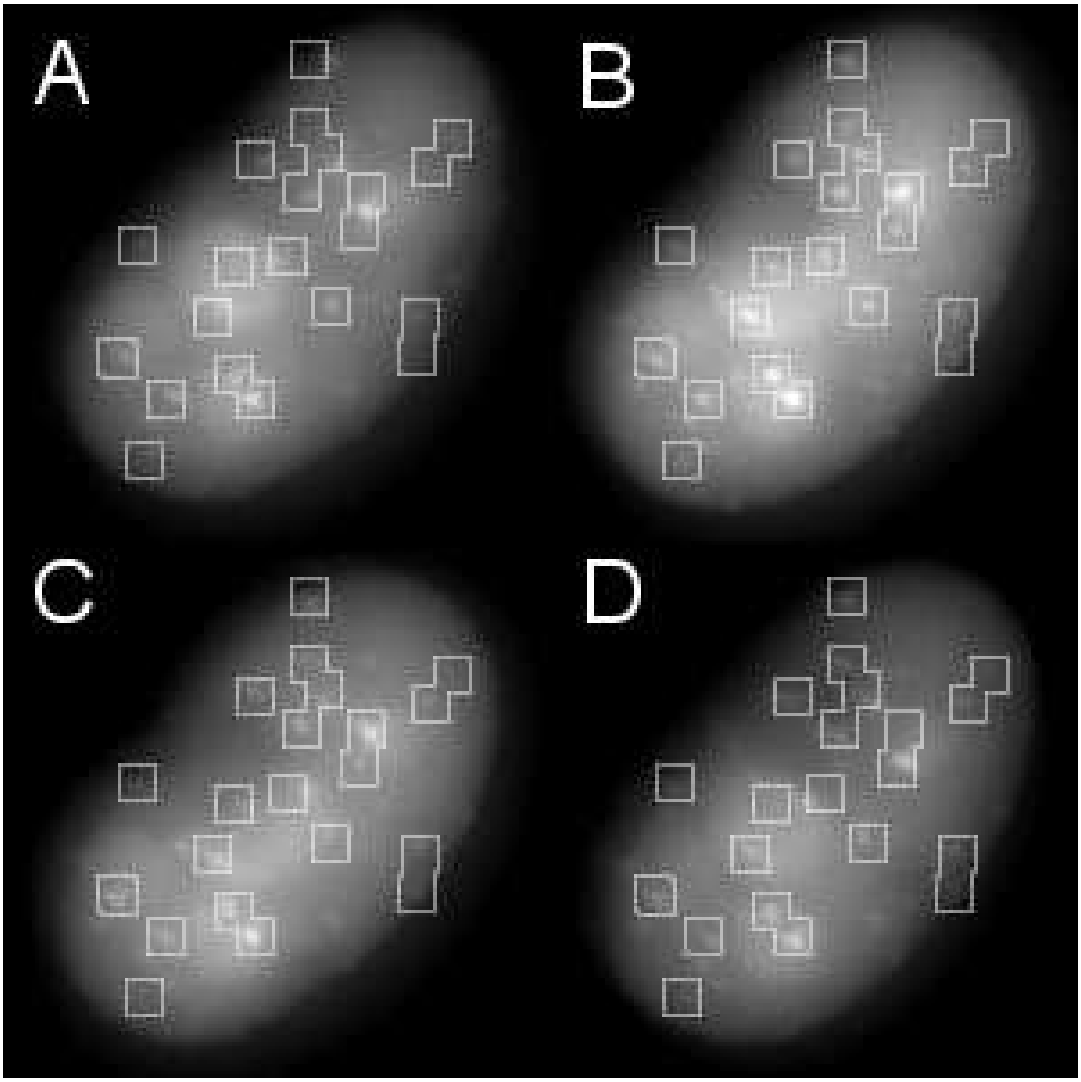


Figure 5.12: Illustration of movements of intracellular structures (dots) in the top-most cell in Fig. 2.9 on page 19. The same frames A,B,C and D as in that earlier figure are used here. The global movement of the cell was suppressed by manual registration (alignment based on cell contour) in which only translations and rotations were used. The square boxes are registered as well. They illustrate the magnitude of dot movements. Notice that many dots appear stationary within the cell, only a few are moving between frames. Box edge is 10px, the diameter of brighter dots is on average 5px. All images were enhanced for printing purposes.

for long sequences. We have tested the generator for various magnitudes of motion and for sequences with even 100 frames, which are approximately 8 times longer than typical real acquired sequences, and have noticed no degradation of flow fields and/or image quality. Motion boundaries in a generated dataset are supported by utilizing smooth motion in two independent layers. Since one layer is defined to be below the other, composing them together has turned into overlaing the one over the other allowing for sharp motion boundaries to appear in the data.

The bottom layer, denoted with BG in Fig. 5.13, is supposed to conduct a global motion, with respect to the frame coordinate system, whereas the top layer, denoted with FG, is supposed to contain an *additional* independent local motion, i.e., motion relative to the moving background. Hence, the flow field induced by the top layer, FG_i FF, must be concatenated to the global flow field, BG FF, to obtain a fully defined independent local motion, BG+FG_i FF, with respect to the frame coordinate system. In fact, there is one such foreground layer associated with every foreground objects, marked with its foreground mask FG_i Mask. That is why the "subscript" $i$ is used. Theoretically, there can more layers added but it seems to us that two are sufficient for simulating motion of cells together with motion of its intracellular structures. Additional layers would be welcomed if we would allow, for example, for occlusion of objects in the sequences.

The control is simplified by using images instead of numerical values, at least for the case of the foreground layer where mask of objects as such, FG Mask, and mask of possible positions, FG MoPP, are used.

The generated inter-frame motion supports for motion coherency both in magnitude and direction of motion. This enables us to control the motion not only in space, by means of defining where an object is allowed to appear, but also in time. We can, sort of, programm two cells to meet at certain point in space and time (Fig. 5.11). We conclude the summary where we stared. Important aspect of the generated motion is that it is encoded directly with the flow fields, the BG FF and FG_i FF.

We consider the generator presented in this work as a solid foundation prepared for further extensions. These may be well arbitrary, e.g., the control over generated sequences may be extended or completely changed. Valuable is the core of the generator where inner states are represented and where content of a static reference image is "made to move". We have implemented the generator for 2D and 3D time-lapse sequences. That is, it can be used for testing optical flow computation methods on simulated wide-field (2D) or confocal (3D) microscopy data.

### 5.5.2 Concept of universal generator

In this section we present a concept of, what we call, the universal generator for live cell studies. It must be stressed that currently it is a vision to some extent of a generator that we haven't fully implemented and tested yet. Nevertheless, we base the concept on our experience and on approaches that have proved good in the current implementation so far.

The concept is outlined in Fig. 5.13. In fact, it is, to some extent, an extension of the successful and proven concept with highly specific and non-trivial modules. The scheme consists of the input and output area, in light and dark blue, respectively, of the generator itself, in the green areas, and of the Modules section, in yellow. As a matter of fact, the

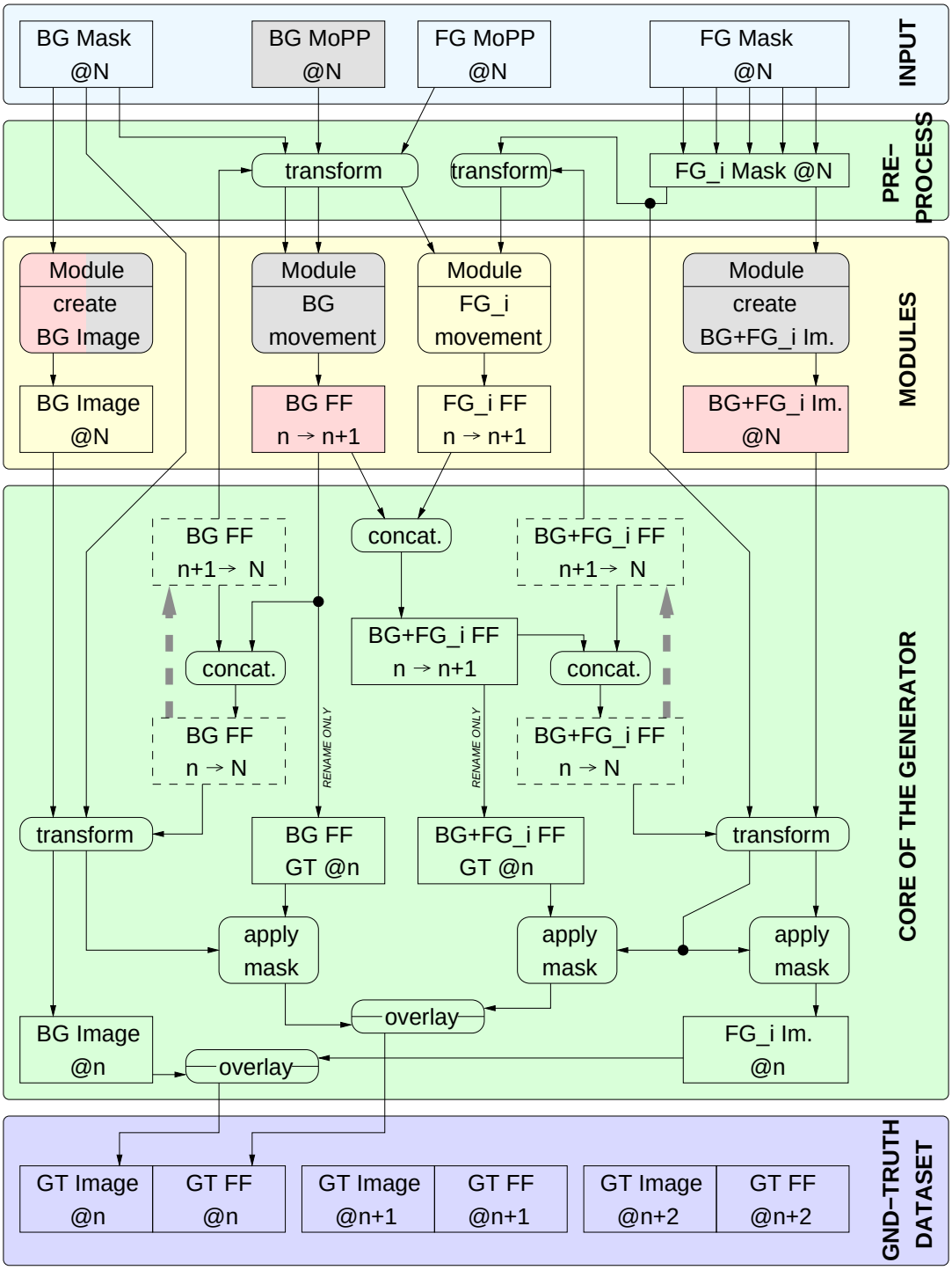


Figure 5.13: An overview of the universal generator concept. The rectangles represent images (sometimes shortened to Im.) and flow fields, denoted as FF. The rectangles with round corners represent operations such as the backward tranformation or concatenation of two flow fields. Comments are given in the text.

current implementation of generator is depicted in the green section. The Modules section, on the other hand, is subject to the proposed extensions.

Following the summary on the current generator from the previous section, the universal generator starts with generating the last frame of the sequence. Say, this would be the Nth frame. The input images, be it the masks (input area) or the pseudo-real image (the BG Image and the BG+FG_i Im.), must be, therefore, compatible with the situation in the Nth frame. It then proceeds from every (n+1)th to create a new frame, the nth. It modifies the mask images to comply with the current, n+1, situation in the pre-process stage. This allows for the Modules to accomodate for the current situation when generating images and flow fields. Once the inter-frame fields are ready, they are processed in the generator core. The flow fields linking the new situation, n, with the reference, N, are updated. These are the BG FF (n→N) and the BG+FF_i FF (n→N). Using the updated fields, the reference mask and image data are transformed and composed together. Also the current inter-frame fields are composed. The linking fields represent an inner state of the generator. They are kept for the next iteration, what is denoted with the thick dashed gray lines in the scheme. Before the first frame is created, they are initialized with zero vectors.

We propose to use the mask images together with the frame number to drive the Modules. As we have achieved good results with our Module for creating autonomous coherency-preserving movements for foreground objects, we suggest its use also for the background object. Hence, the mask of possible positions was introduced even for the background object. Such masks may still provide additional information such as the orientation angle, Fig. 5.7. We may increase the number of masks to increase the number of position-dependent parameters to improve the control. In this way, we could have obtained a vector of parameters for any position within the frame coordinate system. On the other hand, many parameters may disturb the nature-simulating variability.

Regarding the Modules for generating inter-frame motion of either the background or the foreground objects, virtually any motion can be simulated provided its flow field is smooth enough such that the "copy" effect is avoided. The other two Modules for creation of the reference background and foreground images are the tough ones. Their purpose is it provide with pseudo-real reference image which the generator will use in the sequence. For this reason, the generator requires to have an image of solely the background object, which is typically the image with non-specific staining outlining a cell and/or noise in the fluorescence microscopy images, and an image of the foreground objects, which is typically the stained intracellular structures. Owing to the properties of the backward transform on maintaing high quality transformed images, FG_i Im. at n, it is important that the reference image, FG_i Im. at N, contains also the context of the foreground objects. This explains why the current implementation uses a sample real image for the foreground objects, as it trivially fullfils this requirement, and that it generates a new artificial image of background, as the real one can't be used because of the displayed foreground objects.

Anyway, based on the recent publication by Svoboda *et al.* [171] and also on the survey part of it, we see that first attempts on generating artificial high-fidelity images from the field of fluorescence microscopy have been already made. For instance, cell populations were investigated by Lehmussola *et al.* [169, 170] and the HL-60 cell line and granulocytes were investigated by Svoboda *et al.* [175, 171]. We envision that the image-generating modules based on the above publications would work in one of the two modes. They can

either generate an image only once at the beginning of generating of the ground-truth dataset and for every new frame they modify the image by addding some random noise. Or, they can generate a new image for every new frame but it would exactly repeat the privous run with only a few appropriately adjusted small random deviations, which would eventuelly produce the same image with decent variations in it.

Our current implementation differs from the proposed one in that it misses the gray rectangles and uses the pink ones in addition. In particular, it lacks the support for mask of possible positions of a background object. Instead, it only generates (background) flow fields on a regular basis, the pink BG FF, based on the fixed input parameters. A cell motion is then regular as noted at the end of Section 5.4.3. It uses a real sample image, the pink BG+FG_i Im., as a reference one for tranforming the foreground objects to given positions when creating a new frame. At last, a simple model is used for the creation of the reference background image.

In the end, we give an example of a preliminary result of the universal generator. The example aims to show that, despite our generator is targeted for live cell microscopy, it can also provide datasets one would expect to emanate from the field of computer vision. At the same time, the example also shows two shortcommings of the generator in the view of computer vision needs. In Fig. 5.14, a few artificially generated frames of the well-known Hamburg taxi sequence are shown, for which we simply changed the Module for creation of background images in the generator to always provide the same background image with no cars, Fig. 5.14D. The van is moving a bit up and down during its reversing, which is manifested with different colour in the flow field, due to the unchanged Module for foreground motion. The white taxi cab is only translating during its turn. We cannot turn the car naturaly with respect to the camera view because, in the context of our generator, this would require to have another sample real image where the car is shown at appropriate position. But this is not expected to happen in live cell microscopy. Note that the cars are actually going backward in our simulation. This is because we have intiated it with the original first frame, which is where the generating should end up (since the generator proceeds from the last to the first synthetic frame).

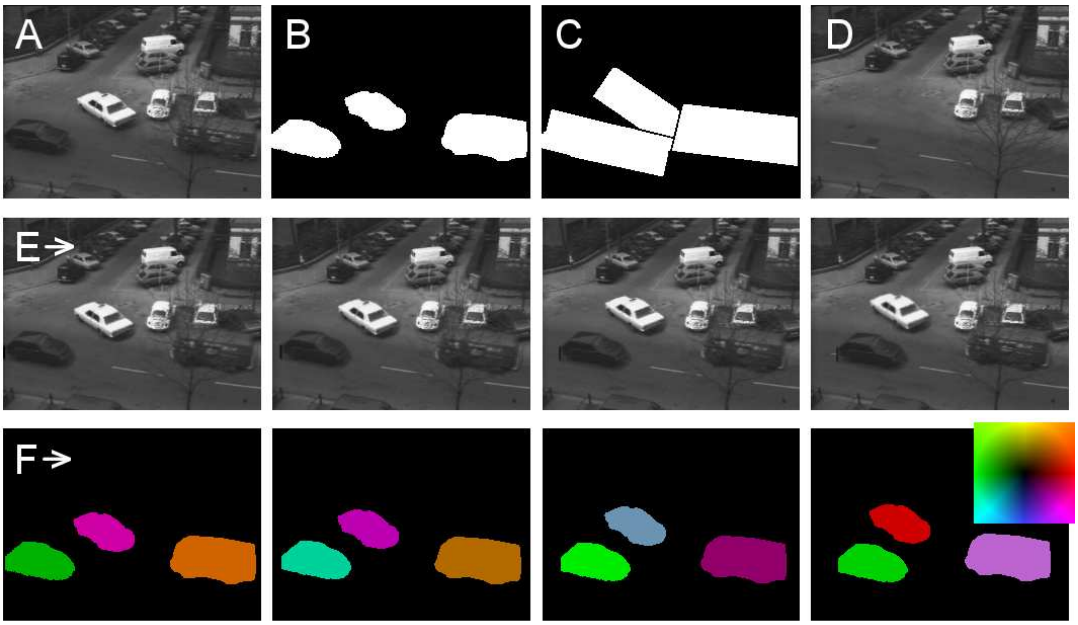


Figure 5.14: Example of the artificially generated Hamburg taxi sequence with the used sample real image, mask of foreground objects and of possible positions in A, B and C, respectively. In D, the image used for the background is shown. The last four frames of the generated sequence and associated flow fields are shown in E and F, respectively. The earlier the frame is, the more right in its row it is. All images were enhanced for printing purposes.

# Chapter 6

# Summary of the thesis

In this thesis we have focused on computing optical flow and on designing a tool for the evaluation of optical flow methods both in the context of live cell studies.

In live cell studies we typically deal with sequences of images that are acquired periodically over time, the time-lapse sequences. This work, specifically, concerns the image sequences acquired using the time-lapse fluorescence light microscopy. The time-lapse observation opens new views on cells to explore, e.g., we may study the growth of cells or, generally, any parameter as a function of time. In order to do this in an automatic way, we need to solve segmentation and tracking of objects in the image sequences. In this work, we aim only towards the tracking alone.

After introducing the principles of fluorescence image acquisition together with main components of modern automated fluorescence microscopes and their properties and limits, we presented a case study of the type of image data we should expect to deal with in this field of science. We then surveyed approaches to tracking in general and in the time-lapse microscopy. We have found that the predominantly used techniques can be classified either as image registration, optical flow or combination of both. As a result, we inspected the image registration and optical flow more closely. We concluded this theme with reasoning why we think it is worth using the optical flow for tracking. Note that an optical flow estimates a flow field in which a vector is assigned to every pixel in an image. The vector represents the difference in position of the same pixel content between two images. The idea is to track a given position in the image by simply following flow vectors. Finally, we presented a theory on representation of motion in both space-time images as well as in the Fourier domain. We discussed some of its aspects with respect to the human visual system as well as with respect to motion estimation based on Gabor filtering.

We have opted to use the optical flow computation methods based on spatial filtering. These methods rely on intensive use of orientation-selective Gabor filters, which is a concept evidenced in the early stages of human visual system. The filtering in humans seems to work in quadrature pairs. We have modelled it with complex Gabor filters. The parallel with human visual system and also the good results of filtering-based optical flow methods in earlier comparison studies were the main motivating factors to choose these methods. In particular, we focused on the energy-based method by David Heeger as it appears to model the human visual system plausibly.

The intensive use of the complex filtering seems to us to be also the bottleneck of the approach. We have, therefore, put emphasis on efficient and accurate image filtering for

optical flow. The complex Gabor filters were studied as well as Gaussian filters because the Gabor filtering can be efficiently computed as modulation, Gaussian filtering and demodulation. Firstly, we analyzed recursive 1D filtering to show it is very fast, efficient and accurate. On the other hand, handling of boundary conditions is somewhat complicated but we demonstrated that it is feasible. Secondly, we investigated separability of Gaussian and Gabor filters. As a result, we introduced a framework which utilizes many recursive 1D image filtering tasks along generally oriented axes. The framework allows for filtering with general Gaussian and Gabor filters, that is even with anisotropic filter, which manifests itself with elliptical kernel shape with distinguished main axis. Important achieved result is that this axis can be arbitrarily oriented. The framework is more accurate but slightly less efficient compared to an optimal solution available. Nonetheless, for the target case of Gabor bank filtering we presented a scheme that is shown to give an almost-optimal efficiency. To sum it up, we managed to find a way to conduct position-invariant filtering with bank(s) of anisotropic complex 3D Gabor filters very efficiently and accurately. We have also demonstrated that the anisotropy allows for increased orientation-sensitivy of the filters, what is expected to lead to an improvement in the filtering-based optical flow methods. Our results were demonstrated with measurements.

Unfortunatelly, at the time of writing we have only *preliminary* results on the energy-based optical flow computation with our advanced filtering employed. The results show some improvement over the original method but we admit that it *currently* can't compete with results achieved with state-of-the-art optical flow methods. This shall be the subject of our future work.

In the last part of the thesis we have focused on a tool for the evaluation of any optical flow method for the popular field of biomedical image processing. While there exist attempts to establish benchmark datasets for optical flow evaluation focused on issues from general computer vision field, no one, to the best of our knowledge, has provided benchmark datasets for biomedical images. We started by identifing goals for such performace evaluation followed by an introduction to accuracy measurements. In the survey part we overviewed and discussed the available approaches in the context of biomedical imaging. In the practical part we described our solution: the generator of sequences of test images with associated ground-truth flow fields. We also presented detailed discussion on important aspects of the generating procedure. Owing to the diversity and specificity of visual appearance in biomedical images, we have opted to design a generator rather than establishing set of selected images. Our generator works with one global motion layer to move the whole cell and several independent local motion layers to additionally move selected interior cell structures. Movements are described using flow fields, which allows to simulate complex processes in the cell. Our solution requires an input sample image which is "set to motion". The similarity with real images is kept in this way. We conclude this part with directions to future work on this topic. Results were exemplified with generated sequences.

Selected author's original publications are reprinted at the end of the thesis. All algorithms were implemented in C++ and are available under the GNUv3 licence as part of the OpticalFlow library at the web pages of the Centre for Biomedical Image Analysis: http://cbia.fi.muni.cz.

# Bibliography


[1] M. Kozubek, "Image acquisition and its automation in fluorescence microscopy," in *From Cells to Proteins: Imaging Nature across Dimensions* (V. Evangelista, L. Barsanti, V. Passarelli, and P. Gualtieri, eds.), vol. 3 of *NATO Security through Science Series*, pp. 227–270, Springer Netherlands, 2005.

[2] M. Kozubek, "FISH imaging," in *Confocal and Two-Photon Microscopy: Foundations, Applications and Advances*, pp. 389–429, Wiley-Liss, Inc., 2002.

[3] M. M. Frigault, J. Lacoste, J. L. Swift, and C. M. Brown, "Live-cell microscopy - tips and tools," *J Cell Sci*, vol. 122, no. 6, pp. 753–767, 2009.

[4] C. Vonesch, F. Aguet, J.-L. Vonesch, and M. Unser, "The colored revolution of bioimaging," *Signal Processing Magazine, IEEE*, vol. 23, pp. 20–31, may 2006.

[5] D. B. Murphy, *Fundamentals of Light Microscopy and Electronic Imaging*. John Wiley, 2001.

[6] Q. Wu, F. Merchant, and K. Castleman, *Microscope Image Processing*. Elsevier Science Inc., 2008.

[7] J. W. Lichtman and J.-A. Conchello, "Fluorescence microscopy," *Nature Methods*, vol. 2, pp. 910–919, 2005.

[8] D. E. Wolf, "Fundamentals of fluorescence and fluorescence microscopy," in *Digital Microscopy, 3rd Edition* (G. Sluder and D. E. Wolf, eds.), vol. 81 of *Methods in Cell Biology*, pp. 63 – 91, Academic Press, 2007.

[9] T. Stearns, "Green fluorescent protein: The green revolution," *Current Biology*, vol. 5, no. 3, pp. 262–264, 1995.

[10] K. Rohr, W. J. Godinez, N. Harder, S. Wörz, J. Mattes, W. Tvaruskó, and R. Eils, "Tracking and quantitative analysis of dynamic movements of cells and particles," in *Live Cell Imaging* (R. D. Goldman, J. R. Swedlow, and D. L. Spector, eds.), pp. 239–256, NY, USA: CSHL Press, 2nd ed., 2010.

[11] D. Gerlich, J. Mattes, and R. Eils, "Quantitative motion analysis and visualization of cellular structures," *Methods*, vol. 29, no. 1, pp. 3–13, 2003.

[12] R. M. Rangayyan, *Biomedical Image Analysis*. CRC Press, 2005.

[13] R. Eils and C. Athale, "Computational imaging in cell biology," *The Journal of Cell Biology*, vol. 161, pp. 447–481, 2003.

[14] C. Zimmer, B. Zhang, A. Dufour, A. Thebaud, S. Berlemont, V. Meas-Yedid, and J.-C. O. Marin, "On the digital trail of mobile cells," *Signal Processing Magazine, IEEE*, vol. 23, pp. 54 –62, may 2006.

[15] H. Bornfleth, P. Edelmann, D. Zink, T. Cremer, and C. Cremer, "Quantitative motion analysis of subchromosomal foci in living cells using four-dimensional microscopy," *Biophysical Journal*, vol. 77, no. 5, pp. 2871 – 2886, 1999.

[16] P. Matula, P. Matula, M. Kozubek, and V. Dvořák, "Fast point-based 3-D alignment of live cells," *IEEE Transactions on Image Processing*, vol. 15, pp. 2388–2396, 2006.

[17] E. Meijering, I. Smal, and G. Danuser, "Tracking in molecular bioimaging," *Signal Processing Magazine, IEEE*, vol. 23, pp. 46 – 53, may 2006.

[18] K. Miura, "Tracking movement in cell biology," in *Microscopy Techniques* (J. Rietdorf, ed.), vol. 95 of *Advances in Biochemical Engineering/Biotechnology*, pp. 267–295, Springer Berlin / Heidelberg, 2005.

[19] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[20] C. Cédras and M. A. Shah, "Motion based recognition: A survey.," *Image and Vision Computing*, vol. 13, no. 2, pp. 129–155, 1995.

[21] D. S. Zhang and G. Lu, "Segmentation of moving objects in image sequence: a review," *Circuits, Systems and Signal Processing (Special Issue on Multimedia Communication Services)*, vol. 20, pp. 143–183, 2001.

[22] A. B. Watson and A. J. Ahumada, "A look at motion in the frequency domain," in *Motion83*, pp. 1–10, 1983.

[23] A. B. Watson and J. A. J. Ahumada, "Model of human visual-motion sensing," *J. Opt. Soc. Am. A*, vol. 2, no. 2, pp. 322–341, 1985.

[24] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *journal of the optical society of America A*, vol. 2, no. 2, pp. 284–299, 1985.

[25] D. J. Heeger, *Models for Motion Perception*. Dissertation thesis, University of Pennsylvania, 1987.

[26] R. L. D. Valois and K. K. D. Valois, *Spatial Vision*. Oxford Univ. press, 1988.

[27] E. P. Simoncelli and D. J. Heeger, "A model of neuronal responses in visual area MT," *Vision Research*, vol. 38, no. 5, pp. 743 – 761, 1998.

[28] L. K. Cormack, *Handbook of Image and Video Processing*, ch. Computational Models of Early Human Vision, pp. 325–345. Elsevier, Academic Press, 2005.

[29] S. Marat, T. H. Phuoc, L. Granjon, N. Guyader, D. Pellerin, and A. Guérin-Dugué, "Spatio-temporal saliency model to predict eye movements in video free viewing," in *In Proc. of 16th European Signal Processing Conference EUSIPCO 2008*, p. 5, 2008.

[30] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic Gauss filtering," in *Proceedings of the 7th ECCV*, pp. 99–112, Springer-Verlag, 2002.

[31] O. Wirjadi and T. Breuel, "Approximate separable 3D anisotropic Gauss filter," in *IEEE International Conference on Image Processing (ICIP 2005)*, pp. 149–152, 2005.

[32] C. H. Lampert and O. Wirjadi, "An optimal nonorthogonal separation of the anisotropic Gaussian convolution filter," *IEEE Transactions on Image Processing*, vol. 15, pp. 3501–3513, nov 2006.

[33] S. Y. M. Lam and B. E. Shi, "Recursive anisotropic 2-D Gaussian filtering based on a triple-axis decomposition," *IEEE Trans. on Image Processing*, vol. 16, no. 7, pp. 1925–30, 2007.

[34] M. A. Rizzo, M. W. Davidson, and D. W. Piston, "Fluorescent protein tracking and detection: Applications using fluorescent proteins in living cells," in *Live Cell Imaging* (R. D. Goldman, J. R. Swedlow, and D. L. Spector, eds.), pp. 3–34, NY, USA: CSHL Press, 2nd ed., 2010.

[35] J. B. Pawley, "Sources of noise in three-dimensional microscopical data sets," in *Three-Dimensional Confocal Microscopy: Volume Investigation of Biological Specimens* (J. K. Stevens, L. R. Mills, and J. E. Trogadis, eds.), pp. 47 – 93, Academic Press, 1994.

[36] H.-W. Ai, S. Olenych, P. Wong, M. Davidson, and R. Campbell, "Hue-shifted monomeric variants of clavularia cyan fluorescent protein: identification of the molecular determinants of color and applications in fluorescence imaging," *BMC Biology*, vol. 6, no. 1, p. 13, 2008.

[37] R. E. Campbell, "Fluorescent proteins," *Scholarpedia*, vol. 3, no. 7, p. 5410, 2008. http://www.scholarpedia.org/article/Fluorescent_proteins (Oct 2010).

[38] O. Shimomura, F. H. Johnson, and Y. Saiga, "Extraction, purification and properties of aequorin, a bioluminescent protein from luminous hydromedusan, aequorea," *J of Cellular and Comparative Physiology*, vol. 59, no. 3, pp. 223–251, 1962.

[39] D. Davenport and J. Nicol, "Luminescence in hydromedusae," *Proc. of the Royal Society of London Series B-Biological Sciences*, vol. 144, no. 916, pp. 399–411, 1955.

[40] D. C. Prasher, V. K. Eckenrode, W. W. Ward, F. G. Prendergast, and M. J. Cormier, "Primary structure of the aequorea-victoria green-fluorescent protein," *Gene*, vol. 111, pp. 229–233, feb 1992.

[41] M. Chalfie, Y. Tu, G. Euskirchen, W. W. Ward, and D. C. Prasher, "Green fluorescent protein as a marker for gene-expression," *Science*, vol. 263, pp. 802–805, feb 1994.

[42] D. J. Stephens and V. J. Allan, "Light microscopy techniques for live cell imaging," *Science*, vol. 300, no. 5616, pp. 82–86, 2003.

[43] D. Gerlich and J. Ellenberg, "4D imaging to assay complex dynamics in live specimens," *Nat Cell Biol.*, vol. 5, pp. S14–S19, Sep 2003.

[44] M. Kozubek, S. Kozubek, E. Lukášová, A. Marečková, E. Bártová, M. Skalníková, and A. Jergová, "High-resolution cytometry of FISH dots in interphase cell nuclei," *Cytometry*, vol. 36, no. 4, pp. 279–293, 1999.

[45] M. Kozubek, S. Kozubek, E. Lukášová, E. Bártová, M. Skalníková, P. Matula, P. Matula, P. Jirsová, A. Cafourková, and I. Koutná, "Combined confocal and wide-field high-resolution cytometry of fluorescent in situ hybridization-stained cells," *Cytometry*, vol. 45, no. 1, pp. 1–12, 2001.

[46] M. Kozubek, P. Matula, P. Matula, and S. Kozubek, "Automated acquisition and processing of multidimensional image data in confocal in vivo microscopy," *Microscopy Research and Technique*, vol. 64, pp. 164–175, 2004.

[47] J. Hubený, *Applications of PDE - Based Image Processing in Fluorescence Microscopy*. PhD thesis, Faculty of Informatics, Masaryk University, 2008.

[48] P. Nipkow, *Elektrisches teleskop*. Patentschrift 30105, Kaiserliches Patentamt, Germany, 1884.

[49] M. Petráň, M. Hadravský, M. D. Egger, and R. Galambos, "Tandem-scanning reflected-light microscope," *J of Opt. Soc. Am.*, vol. 58, pp. 661–664, 1968.

[50] S. Inoue and T. Inoue, "Chapter 2, direct-view high-speed confocal scanner: The csu-10," in *Cell Biological Applications of Confocal Microscopy* (B. Matsumoto, ed.), vol. 70 of *Methods in Cell Biology*, pp. 87 – 127, Academic Press, 2002.

[51] O. Dzyubachyk, W. A. van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, "Advanced level-set-based cell tracking in time-lapse fluorescence microscopy," *Medical Imaging, IEEE Transactions on*, vol. 29, pp. 852–867, mar 2010.

[52] S. H. Chang, F. H. Cheng, W. H. Hsu, and G. Z. Wu, "Fast algorithm for point pattern-matching: Invariant to translations, rotations and scale changes," *PR*, vol. 30, pp. 311–320, feb 1997.

[53] A. Dufour, V. Shinin, S. Tajbakhsh, N. Guillen-Aghion, J.-C. Olivo-Marin, and C. Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces," *Image Processing, IEEE Transactions on*, vol. 14, pp. 1396–1410, sep 2005.

[54] J. Konrad, *Handbook of Image and Video Processing*, ch. Motion detection and estimation, ch. 3.10, pp. 253–274. Academic Press, 2nd ed., 2005.

[55] W. Tvaruskó, J. Mattes, and R. Eils, "Analyzing live cell data and tracking dynamic movements," in *Live Cell Imaging: A Laboratory Manual* (R. D. Goldman and D. L. Spector, eds.), pp. 303–326, CSHL Press, 2005.

[56] S. Dubuisson, "An adaptive clustering for multiple object tracking in sequences in and beyond the visible spectrum," *Computer Vision and Pattern Recognition Workshop*, vol. 0, p. 142, 2006.

[57] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen, "Tracking in cell and developmental biology," *Seminars in Cell & Developmental Biology*, vol. 20, no. 8, pp. 894–902, 2009.

[58] N. Harder, F. Mora-Bermudez, W. J. Godinez, J. Ellenberg, R. Eils, and K. Rohr, "Automated analysis of the mitotic phases of human cells in 3D fluorescence microscopy image sequences," in *9th International Conference on Computing and Computer-Assisted Intervention MICCAI 2006*, vol. 4190 of *LNCS*, pp. 840–848, 2006.

[59] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, dec 2006.

[60] M. Bertalmío, G. Sapiro, and G. Randall, "Morphing active contours," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 733–737, jul 2000.

[61] A.-R. Mansouri, "Region tracking via level set pdes without motion computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 947–961, jul 2002.

[62] D. Cremers and C. Schnörr, "Statistical shape knowledge in variational motion segmentation," *Image and Vision Computing*, vol. 21, no. 1, pp. 77–86, 2003.

[63] K. Rangarajan and M. Shah, "Establishing motion correspondence," *CVGIP: Image Underst.*, vol. 54, pp. 56–73, jun 1991.

[64] B. Zitová and J. Flusser, "Image registration methods: a survey," *IVC*, vol. 21, pp. 977–1000, oct 2003.

[65] L. G. Brown, "A survey of image registration techniques," tech. rep., Columbia University, Jan. 1992.

[66] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[67] A. Roche, G. Malandain, X. Pennec, and N. Ayache, "The correlation ratio as a new similarity measure for multimodal image registration," in *Proceedings MICCAI'98*, vol. 1496 of LNCS, Springer Verlag, 1998.

[68] P. Viola and W. M. Wells, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, pp. 137–154, 1997.

[69] B. S. Reddy and B. N.Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *Image Processing, IEEE Transactions on*, vol. 5, pp. 1266–1271, Aug. 1996.

[70] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995.

[71] D. J. Fleet and Y. Weiss, *Mathematical Models in Computer Vision: The Handbook*, ch. Optical Flow Estimation, ch. 15, pp. 239–258. Springer, 2005.

[72] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[73] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg, "A survey on variational optic flow methods for small displacements," in *Mathematical Models for Registration and Applications to Medical Imaging*, vol. 10 of *Mathematics in Industry*, pp. 103–136, Springer Berlin Heidelberg, 2006.

[74] P. J. Burt, C. Yen, and X. Xy, "Multi-resolution flow-through motion analysis," in *CVPR83: Proceedings of the Conference Computer Vision and Pattern Recognition*, pp. 246–252, 1983.

[75] J. L. Barron, S. S. Beauchemin, and D. J. Fleet, "On optical flow," in *6th Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots*, pp. 3–14, 1994.

[76] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. COM-31,4, pp. 532–540, 1983.

[77] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, 1984.

[78] J. L. Barron, "Experience with 3D optical flow on gated MRI cardiac datasets," *Computer and Robot Vision, Canadian Conference*, vol. 0, pp. 370–377, 2004.

[79] J. Barron, "3D optical flow in gated MRI cardiac datasets," in *Imaging Beyond the Pinhole Camera* (K. Daniilidis and R. Klette, eds.), pp. 331–344, Springer Netherlands, 2006.

[80] A. Verri and T. Poggio, "Motion field and optical flow: Qualitative properties," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 490–498, may 1989.

[81] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms," in *In Proc. of the 9th British Mach. Vis. Conf. (BMVC '98)*, vol. 1, pp. 195–204, 1998.

[82] C. Stiller and J. Konrad, "Estimating motion in image sequences, a tutorial on modeling and computation of 2D motion," *IEEE Signal Process. Mag.*, vol. 16, pp. 70–91, 1999.

[83] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information," *Int. J. Comput. Vision*, vol. 5, no. 1, pp. 77–104, 1990.

[84] I. Austvoll, "Directional filters and a new structure for estimation of optical flow," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2, pp. 574 –577 vol.2, sep 2000.

[85] E. Kristoffersen, I. Austvoll, and K. Engan, "Dense motion field estimation using spatial filtering and quasi eigenfunction approximations," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3, pp. 1268–71, sep 2005.

[86] P. Quelhas, A. Mendonça, and A. Campilho, "Optical flow based arabidopsis thaliana root meristem cell division detection," in *Image Analysis and Recognition* (A. Campilho and M. Kamel, eds.), vol. 6112 of *Lecture Notes in Computer Science*, pp. 217–226, Springer Berlin / Heidelberg, 2010.

[87] B. Rieger, C. Molenaar, R. W. Dirks, and L. J. van Vliet, "Alignment of the cell nucleus from labeled proteins only for 4D in vivo imaging," *Microscopy Research and Technique*, vol. 64, pp. 142–150, 2004.

[88] A. E. Carlsson, A. D. Shah, D. Elking, T. S. Karpova, and J. A. Cooper, "Quantitative analysis of actin patch movement in yeast," *Biophysical Journal*, vol. 82, no. 5, pp. 2333–2343, 2002.

[89] S. Yang, D. Kohler, K. Teller, T. Cremer, P. L. Baccon, E. Heard, R. Eils, and K. Rohr, "Nonrigid registration of 3-D multichannel microscopy images of cell nuclei," *Image Processing, IEEE Transactions on*, vol. 17, pp. 493–499, apr 2008.

[90] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000. Available at Intel Corporation Microprocessor Research Labs, Janurary 2011.

[91] I.-H. Kim, Y.-C. Chen, D. L. Spector, R. Eils, and K. Rohr, "Non-rigid registration of 2D and 3D dynamic cell nuclei images for improved classification of subcellular particle motion," *Image Processing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2010.

[92] A. Sacan, H. Ferhatosmanoglu, and H. Coskun, "CellTrack: an open-source software for cell tracking and motility analysis," *Bioinformatics*, vol. 24, no. 14, pp. 1647–1649, 2008.

[93] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *International Journal of Computer Vision*, vol. 67, pp. 141–158, 2006.

[94] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Computer Vision - ECCV 2004* (T. Pajdla and J. Matas, eds.), vol. 3024 of *Lecture Notes in Computer Science*, pp. 25–36, Springer Berlin / Heidelberg, 2004.

[95] M. Maška, O. Daněk, C. Ortiz de Solórzano, A. Muñoz-Barrutia, M. Kozubek, and I. F. García, "A two-phase segmentation of cell nuclei using fast level set-like algorithms," in *Proceedings of the 16th Scandinavian Conference on Image Analysis*, SCIA '09, pp. 390–399, Berlin, Heidelberg: Springer-Verlag, 2009.

[96] B. McCane, K. Novins, D. Crannitch, and B. Galvin, "On benchmarking optical flow," *Comput. Vis. Image Underst.*, vol. 84, pp. 126–143, oct 2001.

[97] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, oct 2007.

[98] E. Bruno and D. Pellerin, "Robust motion estimation using spatial Gabor-like filters," *Signal Process.*, vol. 82, no. 2, pp. 297–309, 2002.

[99] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *DARPA81*, pp. 121–130, 1981.

[100] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods," *Int. J. Comput. Vision*, vol. 61, pp. 211–231, feb 2005.

[101] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L1 optical flow," in *Proceedings of the 29th DAGM conference on Pattern recognition*, (Berlin, Heidelberg), pp. 214–223, Springer-Verlag, 2007.

[102] W. Chen and J. L. Barron, "High accuracy optical flow method based on a theory for warping: 3D extension," in *Proceedings of the 7th International Conference on Image Analysis and Recognition, ICIAR 2010* (A. Campilho and M. Kamel, eds.), vol. 6111 of *Lecture Notes in Computer Science*, pp. 250–262, 2010. LNCS 6111.

[103] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *CVPR*, pp. 2432–2439, 2010.

[104] J. Weber and J. Malik, "Robust computation of optical flow in a multi-scale differential framework," *Int. J. Comput. Vision*, vol. 14, no. 1, pp. 67–81, 1995.

[105] D. J. Heeger, "Notes on motion estimation," tech. rep., CiteSeerX - Scientific Literature Digital Library and Search Engine (United States), 1998.

[106] D. J. Heeger, "Optical flow using spatiotemporal filters," *International journal of computer vision*, vol. 1, no. 4, pp. 279–302, 1988.

[107] K. Pauwels and M. M. van Hulle, "Realtime phase-based optical flow on the gpu," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1–8, jun 2008.

[108] H. H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," *CVGIP*, vol. 21, pp. 85–117, jan 1983.

[109] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *IJCV*, vol. 2, pp. 283–310, jan 1989.

[110] E. H. Adelson and J. R. Bergen, "The extraction of spatio-temporal energy in human and machine vision," in *Motion86*, pp. 151–155, 1986.

[111] D. J. Heeger, "Model for the extraction of image flow," *J. Opt. Soc. Am. A*, vol. 4, no. 8, pp. 1455–1471, 1987.

[112] C. W. G. Clifford and K. Langley, "Recursive implementations of temporal filters for image motion computation," *Biological Cybernetics*, vol. 82, pp. 383–390, 2000.

[113] J. Jan, *Digital Signal Filtering, Analysis and Restoration.* The Institution of Electrical Engineers, London, 2000.

[114] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Prentice–Hall, 1975.

[115] N. Grzywacz and A. Yuille, "A model for the estimate of local velocity," in *Computer Vision, ECCV 90* (O. Faugeras, ed.), vol. 427 of *Lecture Notes in Computer Science*, pp. 331–335, Springer Berlin / Heidelberg, 1990.

[116] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *J. Opt. Soc. Am. A*, vol. 2, no. 7, pp. 1160–1169, 1985.

[117] J. P. Jones and L. A. Palmer, "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex," *J Neurophysiol*, vol. 58, no. 6, pp. 1233–1258, 1987.

[118] J. Bigün, "Speed, frequency, and orientation tuned 3-D Gabor filter banks and their design," in *Pattern Recognition, 1994. Vol. 3 - Conference C: Signal Processing, Proceedings of the 12th IAPR International Conference on*, pp. 184 –187 vol.3, oct 1994.

[119] A. Spinei, D. Pellerin, and J. Hérault, "Spatiotemporal energy-based method for velocity estimation," *Signal Process.*, vol. 65, no. 3, pp. 347–362, 1998.

[120] D. Gabor, "Theory of communications," *J. of IEE*, vol. 93, pp. 429–459, 1946.

[121] T. S. Lee, "Image representation using 2D Gabor wavelets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, pp. 959 –971, oct 1996.

[122] J. R. Movellan, "Tutorial on Gabor filters," tech. rep., UC San Diego, 2010. document available at http://mplab.ucsd.edu/.

[123] G. H. Golub and C. van Loan, *Matrix computations.* John Hopkins University Press, 1993.

[124] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[125] B. E. Shi, "Focal plane implementation of 2D steerable and scalable Gabor-type filters," *The Journal of VLSI Signal Processing*, vol. 23, pp. 319–334, 1999.

[126] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal processing*, vol. 44, no. 2, pp. 139–151, 1995.

[127] D. J. Fleet and K. Langley, "Recursive filters for optical flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 1, pp. 61–67, 1995.

[128] C. Clifford, K. Langley, and D. J. Fleet, "Centre-frequency adaptive IIR temporal filters for phase-based image velocity estimation," in *IEE International Conference on Image Processing and Applications*, pp. 173–178, jul 1995.

[129] T. Gautama and M. M. van Hulle, "A phase-based approach to the estimation of the optical flow field using spatial filtering," *IEEE Trans. Neural Networks*, vol. 13, no. 5, pp. 1127–1136, 2002.

[130] P. P. Vaidyanathan, *Multirate systems and filter banks*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[131] I. Austvoll and N. Nayar, "Comparison of FIR and IIR directional filters for estimation of optical flow," in *Norwegian Signal Processing Symposium*, Norsk Forening for Signalbehandling, 2005.

[132] M. Felsberg and G. Sommer, "The monogenic signal," tech. rep., Institut für Informatik und Praktische Mathematik,Christian-Albrechts-Universität, 2001.

[133] M. Felsberg, "Optical flow estimation from monogenic phase," in *IWCM04: First International Workshop on Complex Motion*, vol. LNCS 3417, 2004.

[134] G. B. Whitman, *Linear and Nonlinear Waves*. John Wiley & Sons, Inc., 2nd ed., 1999.

[135] A. V. Oppenheim and J. S. Lim, "The importance of phase in signals," in *IEEE Proceedings Special Issue on Digital Image Processing*, vol. 69, pp. 529–541, 1981.

[136] D. J. Fleet and A. D. Jepson, "Stability of phase information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 12, pp. 1253–1268, 1993.

[137] P. J. Huber, *Robust statistics*. John Wiley & Sons, Inc., 1981.

[138] R. Bracewell, *The Fourier Transform & Its Applications*. McGraw-Hill Science, 3rd ed., 1999.

[139] M. Šonka, V. Hlaváč, and R. Boyle, *Image Processing: Analysis and Machine Vision*. O'Reilly, 1999.

[140] R. Deriche, "Recursively implementing the Gaussian and its derivatives," Tech. Rep. 1893, INRIA, May 1993.

[141] L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive Gaussian derivative filters," in *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, p. 509, IEEE Computer Society, 1998.

[142] J. S. Jin and Y. Gao, "Recursive implementation of LoG filtering," *Real-Time Imaging*, vol. 3, no. 1, pp. 59–65, 1997.

[143] I. T. Young, L. J. van Vliet, and M. van Ginkel, "Recursive Gabor filtering," *Signal processing*, vol. 50, no. 11, pp. 2798–2805, 2002.

[144] S. Tan, J. L. Dale, and A. Johnston, "Performance of three recursive algorithms for fast space-variant Gaussian filtering," *Real-Time Imaging*, vol. 9, no. 3, pp. 215–228, 2003.

[145] A. Bernardino and J. Santos-Victor, "Fast IIR isotropic 2-D complex Gabor filters with boundary initialization," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3338–3348, 2006.

[146] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Trans. Signal Processing*, vol. 55, no. 1, pp. 111–119, 2007.

[147] B. Triggs and M. Sdika, "Boundary conditions for Young - van Vliet recursive filtering," *IEEE Transactions on Signal Processing*, vol. 54, may 2006.

[148] A. Bernardino and J. Santos-Victor, "A real-time Gabor primal sketch for visual attention," in *2nd Iberian Conference on Pattern Recognition and Image Analysis*, p. I:335, jun 2005.

[149] O. Nestares, R. Navarro, J. Portilla, and A. Tabernero, "Efficient spatial domain implementation of a multiscale image representation based on Gabor functions," *J. Electronic Imaging*, vol. 7, pp. 166–173, jan 1998.

[150] V. Areekul, U. Watchareeruetai, and S. Tantaratana, "Fast separable Gabor filter for fingerprint enhancement," in *ICBA'04: Proceeding International Conference on Biometric Authentication*, vol. LNCS 3072, pp. 403–409, Springer, 2004.

[151] A. Rahman, D. Houzet, D. Pellerin, S. Marat, and N. Guyader, "Parallel implementation of a spatio-temporal visual saliency model," *Journal of Real-Time Image Processing*, pp. 1–12, 2010.

[152] NVIDIA, *Compute Unified Device Architecture — Programming Guide*. NVIDIA corp., 2007.

[153] J. Chamorro-Martinez and J. Fdez-Valdivia, "Optical flow estimation based on the extraction of motion patterns," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 1, pp. 925–928, sep 2003.

[154] Z. Liu and R. Klette, "Approximated ground truth for stereo and motion analysis on real-world sequences," in *Advances in Image and Video Technology* (T. Wada, F. Huang, and S. Lin, eds.), vol. 5414 of *Lecture Notes in Computer Science*, pp. 874–885, Springer Berlin / Heidelberg, 2009.

[155] C. McCarthy and N. Barnes, "Performance of optical flow techniques for indoor navigation with a mobile robot," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, pp. 5093–5098, apr 2004.

[156] J. Hedborg and P.-E. Forssén, "Synthetic ground truth for feature trackers," in *SSBA 2008*, 2008.

[157] L. Bagnato, P. Frossard, and P. Vandergheynst, "Optical flow and depth from motion for omnidirectional images using a TV-L1 variational framework on graphs," in *ICIP09*, pp. 1469–1472, 2009.

[158] M. Otte and H. Nagel, "Optical flow estimation: Advances and comparisons," in *Computer Vision, ECCV '94* (J.-O. Eklundh, ed.), vol. 800 of *Lecture Notes in Computer Science*, pp. 49–60, Springer Berlin / Heidelberg, 1994.

[159] T. Lin and J. L. Barron, "Image reconstruction error for optical flow," in *Vision Interface*, pp. 73–80, 1994.

[160] I. Austvoll, "A study of the yosemite sequence used as a test sequence for estimation of optical flow," in *Image Analysis* (H. Kalviainen, J. Parkkinen, and A. Kaarna, eds.), vol. 3540 of *Lecture Notes in Computer Science*, pp. 659–668, Springer Berlin / Heidelberg, 2005.

[161] V. Tavakoli, N. Sahba, A. Ahmadian, and J. Alirezaie, "An evaluation of different optical flow techniques for myocardial motion analysis in B-Mode echocardiography images," in *4th Kuala Lumpur International Conference on Biomedical Engineering 2008* (N. A. Abu Osman, F. Ibrahim, W. A. B. Wan Abas, H. S. Abdul Rahman, and H.-N. Ting, eds.), vol. 21 of *IFMBE Proceedings*, pp. 506–510, Springer Berlin Heidelberg, 2008.

[162] A. A. Gerencser and D. G. Nicholls, "Measurement of instantaneous velocity vectors of organelle transport: Mitochondrial transport and bioenergetics in hippocampal neurons," *Biophysical Journal*, vol. 95, pp. 3070–3099, 2008.

[163] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss, "Human-assisted motion annotation," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 1–8, 2008.

[164] D. Mason, B. McCane, and K. Novins, "Generating motion fields of complex scenes," *Computer Graphics International Conference*, vol. 0, p. 65, 1999.

[165] T. Driemeyer, *Rendering with mental ray.* Springer, 3rd ed., 2005.

[166] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," Tech. Rep. MSR-TR-2009-179, Microsoft Corporation, 2009.

[167] DNA Research, "3Delight, RenderMan™-compliant rendering software." Company web pages, nov 2010. http://www.3delight.com/.

[168] D. Webb, M. A. Hamilton, G. J. Harkin, S. Lawrence, A. K. Camper, and Z. Lewandowski, "Assessing technician effects when extracting quantities from microscope images," *Journal of Microbiological Methods*, vol. 53, pp. 97–106, apr 2003.

[169] A. Lehmussola, J. Selinummi, P. Ruusuvuori, A. Niemisto, and O. Yli-Harja, "Simulating fluorescent microscope images of cell populations," in *IEEE Engineering in Medicine and Biology 27th Annual Conference*, pp. 3153–3156, sep 2005.

[170] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen, and O. Yli-Harja, "Computational framework for simulating fluorescence microscope images with cell populations," *Medical Imaging, IEEE Transactions on*, vol. 26, pp. 1010–1016, jul 2007.

[171] D. Svoboda, M. Kozubek, and S. Stejskal, "Generation of digital phantoms of cell nuclei and simulation of image formation in 3D image cytometry," *Cytometry Part A*, vol. 75A, pp. 494–509, 2009.

[172] R. Eils, D. Gerlich, W. Tvaruskó, D. L. Spector, and T. Misteli, "Quantitative imaging of pre-mRNA splicing factors in living cells," *Molecular Biology of the Cell*, vol. 11, pp. 413–418, feb 2000.

[173] C. Molenaar, K. Wiesmeijer, N. P. Verwoerd, S. Khazen, R. Eils, H. J. Tanke, and R. W. Dirks, "Visualizing telomere dynamics in living mammalian cells using pna probes," *The EMBO Journal*, vol. 22, pp. 6631–6641, 2003.

[174] W. H. D. Vos, G. H. Joss, W. Haffmans, R. A. Hoebe, E. M. M. Manders, and P. van Oostveldt, "Four-dimensional telomere analysis in recordings of living human cells acquired with Controlled Light Exposure Microscopy," *Journal of Microscopy-Oxford*, vol. 238, pp. 254–264, jun 2010.

[175] D. Svoboda, M. Kašík, M. Maška, J. Hubený, S. Stejskal, and M. Zimmermann, "On simulating 3D fluorescent microscope images," in *Computer Analysis of Images and Patterns* (W. Kropatsch, M. Kampel, and A. Hanbury, eds.), vol. 4673 of *Lecture Notes in Computer Science*, pp. 309–316, Springer Berlin / Heidelberg, 2007.

# Publication P1

Reprinted with kind permission from WSEAS:

V. Ulman, "Arbitrarily-oriented anisotropic 3D Gaussian filtering computed with 1D convolutions without interpolation," in *Proceedings of 8th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision*, (Athens), pp. 56–62, 2008. ISSN 1792-4618.

# Publication P2

Reprinted with kind permission from IASTED and ACTA Press:

V. Ulman, "Filtering with anisotropic 3D Gabor filter bank efficiently computed with 1D convolutions without interpolation," in *Proceedings of the Seventh IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, (Calgary), pp. 33–42, 2010.

# Publication P3

This publication is an open-access publication available on the web pages of the Journal on Advances in Signal Processing, http://www.hindawi.com/journals/asp/:

V. Ulman, "Boundary treatment for Young–van Vliet recursive zero-mean gabor filtering," *EURASIP Journal on Advances in Signal Processing*, 2011. Ready to be published after minor changes.

# Publication P4

Reprinted with kind permission from Springer Science + Business Media:

V. Ulman, "Improving accuracy of optical flow of Heeger's original method on biomedical images," in *Proceedings of the 7th International Conference on Image Analysis and Recognition, ICIAR 2010*, pp. 263–273, 2010. LNCS 6111.

# Publication P5

Reprinted with kind permission from INSTICC:

V. Ulman and J. Hubený, "On generating ground-truth time-lapse image sequences and flow fields," in *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, (Angers), pp. 234–239, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007.

# Publication P6

Reprinted with kind permission from Springer Science + Business Media:

V. Ulman and J. Hubený, "Pseudo-real image sequence generator for optical flow computations," in *Proceedings of 15th Scandinavian Conference on Image Analysis*, (Heidelberg), pp. 976–985, 2007. LNCS 4522.

# Publication P7

Reprinted with kind permission from INSTICC:

J. Hubený, V. Ulman, and P. Matula, "Estimating large local motion in live-cell imaging using variational optical flow," in *VISAPP: Proc. of the Second International Conference on Computer Vision Theory and Applications*, pp. 542–548, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007.