Development and application of a system for very long-time microrheology experiments

Dissertation zur Erlangung des Doktorgrades Dr. rer. nat. der Fakultät für Naturwissenschaften der Universität Ulm vorgelegt von

Jonas Pfeil geboren in Ulm

Institut für Experimentelle Physik

2020
Amtierender Dekan zum Zeitpunkt der Abgabe:

Prof. Dr. Thorsten Bernhardt

Gutachter:

1. Prof. Dr. Othmar Marti
2. Prof. apl. Dr. Berndt Koslowski

Tag der Promotion:

23.02.2021
Contents

1 Abstract 1

2 Introduction 2

3 Theory 4
   3.1 Cell cytoskeleton .................................................. 4
      3.1.1 Cell overview ................................................ 4
      3.1.2 Cytoskeleton ................................................... 5
         Actin filament system ........................................... 5
         Intermediate filament system ............................... 7
         Microtubuli .................................................. 7
   3.2 Microrheology ................................................... 7
      3.2.1 Active microrheology ..................................... 8
      3.2.2 Passive microrheology ................................... 11
   3.3 Particle tracking ............................................... 13
      3.3.1 Peak finding .............................................. 15
      3.3.2 Centroid of intensity .................................... 15
      3.3.3 Surface fitting with 4th degree polynomial .............. 16

4 Sensor 20
   4.1 Hardware .......................................................... 20
      4.1.1 Imaging sensor ............................................. 20
      4.1.2 Processing unit ........................................... 22
      4.1.3 Custom interfacing board ............................... 22
   4.2 Software ........................................................... 24
      4.2.1 User interface .............................................. 26
      4.2.2 Particle tracking implementation ....................... 27
4.3 Tracking algorithm optimizations .................................................. 31
  4.3.1 Optimize fitting the polynomial to the image .............................. 31
     Vandermonde matrix and polynomial fitting .................................. 31
     Modified Vandermonde matrices for vector \( \rightarrow \) scalar functions .... 34
     Fast approximate fitting of a polynomial to the image ..................... 35
  4.3.2 Optimized radius determination ................................................ 43
  4.3.3 Optimized computation of \( \sqrt{x} \) ........................................... 46
  4.3.4 Adaptive active sensor region adjustment .................................. 47
4.4 Validation .................................................................................... 48
  4.4.1 Synthetic images .................................................................... 50
     Convergence of particle detection ................................................ 50
     Simulated random walk ............................................................ 50
     Tracking with increased noise level .............................................. 51
  4.4.2 Real measurements ................................................................. 55
  4.4.3 Discussion of the simulated tracking performance ..................... 58
5 Experiments ................................................................................... 59
  5.1 Experimental setup .................................................................... 59
     5.1.1 Microscope ........................................................................ 60
     5.1.2 Sample preparation ........................................................... 60
  5.2 Experimental results .................................................................. 61
     5.2.1 Very long time tracking of a cell treated with blebbistatin ......... 62
     5.2.2 Very long-time tracking of cells before and after treatment with
         blebbistatin ........................................................................... 67
     5.2.3 Temperature of the imaging solution while measuring ............. 76
  5.3 Discussion .................................................................................. 78
6 Summary ....................................................................................... 82
7 Outlook .......................................................................................... 84
  7.1 Further improvements ................................................................. 85
  7.2 Further applicable fields ............................................................. 86

List of Figures ................................................................................... 88
<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>89</td>
</tr>
<tr>
<td>Bibliography</td>
<td>90</td>
</tr>
<tr>
<td>Glossary</td>
<td>99</td>
</tr>
<tr>
<td>Acronyms</td>
<td>100</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>102</td>
</tr>
<tr>
<td>Publications</td>
<td>103</td>
</tr>
<tr>
<td>1 Paper</td>
<td>103</td>
</tr>
<tr>
<td>2 Patents</td>
<td>103</td>
</tr>
<tr>
<td>3 Posters</td>
<td>103</td>
</tr>
<tr>
<td>4 Talks</td>
<td>105</td>
</tr>
</tbody>
</table>
1 Abstract

In this work, a new tool for passive microrheology is presented. Passive microrheology is currently limited in speed and measurement duration by the used video camera system. Normal cameras are limited in speed while high-speed cameras are limited in the recording time. To overcome these limitations, a complementary metal-oxide-semiconductor (CMOS) image sensor was coupled with a field programmable gate array (FPGA) to directly compute the position of the particles in the image and only save the position data instead of the complete image to reduce the data rate significantly, allowing basically infinite recording times with very high speed.

In this work, the algorithms and optimizations for real-time particle tracking and the implementation into an FPGA is demonstrated. Then the system is validated for precision and usability and tested with passive microrheology measurements of polystyrene beads in NIH/3T3 mouse fibroblasts partially treated with blebbistatin to demonstrate the function and versatility of the device.
2 Introduction

Microrheology is a well established technique [MS99; Mas+97] to probe material properties [Nar+13; SA13], especially biological materials [Gar+03; XPW98; Nec+16]. Within the last years, several methods have been developed [Alc+03; Bra+07], each having their own advantages and disadvantages [Wil+09; Bra+07]. The method of passive video microrheology looks very promising because its simple experimental setup [BP03], but proves to be challenging due to the storage bandwidth, speed and size requirements, especially when used in the context of cell stiffness. To overcome these problems, a new tool is developed, harnessing the power of a fast and flexible complementary metal-oxide-semiconductor (CMOS) image sensor and a powerful field programmable gate array (FPGA) to attain the required speed while drastically reducing the storage bandwidth and size requirements.

Traditionally, passive microrheology consists of recording a video of particle movement and computing the position of these particles after the recording offline. This two-step approach is necessary because of the high computing demand to precisely track the particles. This limits the online analysis methods, when high image rates necessary to resolve the important frequencies. Solutions for single particle tracking [Tas+12] or with clever frame time modulation schemes exists [Phi+16], but currently no continuous high frame rate multi particle tracking tool is available. In this work the classical two-step setup is substituted with an online, direct tracking approach to realize such a tool. This can be achieved by reformulating the state-of-the-art tracking algorithms to decrease the computing performance demands and by only saving the relevant data, mostly consisting of the position and radius, instead of recording the entire images. The second benefit is a reduced hardware demand of the imaging sensor as the pixel readout rate can also be substantially reduced by only recording the direct surrounding of the particles. This is possible as the particle position is known while the experiment is running, which is traditionally not the case. Combining the online particle tracking with an adaptive senor
readout, the newly developed tool allows for easy passive microrheology experiments which span more than seven decades of resolution in the time domain without sacrificing precision on the particle position detection and computation.

This novel tool is then tested and compared in performance and precision with state-of-the-art algorithms. The test consists of artificially generated images as well as simulations with an model implementation on real data acquired with an high speed camera. To show a possible application, microrheological measurements with NIH/3T3 fibroplasts affected by blebbistatin are conducted.

Firstly, after a concise overview of the cell cytoskeleton and cell mechanics the theory behind microrheology is explained. Then, some widely used algorithms for particle tracking in images are presented and discussed. One of these algorithms is optimized and improved to accelerate and adapt it to run in real time on dedicated hardware. This implementation is then integrated into a compact sensor which is evaluated and tested for accuracy and performance. In the end this sensor is used in microrheological experiments using NIH/3T3 mouse fibroblasts and blebbistatin to show the time evolution of cell stiffness affected by drug-induced cytoskeleton alterations.
3 Theory

To understand the mechanics of cells, it is crucial to know the main mechanical components of cells. Therefore, a short introduction of the cytoskeleton will be given in chapter 3.1. Then, the most common methods for microrheology measurements for living and fixed cells will be presented in chapter 3.2. One of the main challenges to master for these technologies will then be discussed in more depth in chapter 3.3. This will outline a basic framework and motivation for improving and introducing a new method to do economic viable, long and fast microrheological measurements.

3.1 Cell cytoskeleton

3.1.1 Cell overview

Life has as a basic building block deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). For a proper function, they need to be in a special environment. This environment is provided by cells. They consist of a bilipid layer, forming the cell membrane dividing the cell interior containing the cytoplasm and organelles from the extracellular world. The cell contains proteins forming the various structures and provide most of the functionality. The information for the cell protein formation is stored in the DNA. Eukaryotic cells store their DNA in a separate compartment, called the cell nucleus, while prokaryotic cells do not have a cell nucleus. The most important organelles of cells are lysosomes, ribosomes, mitochondria, the endoplasmatic reticulum and the Golgi apparatus, and for plants chloroplasts. They are responsible for the majority of biochemical processes in cells [Alb02].

The DNA and RNA itself is mostly a data storage, but the transcription proteins in cells are able to read the DNA and in turn produce proteins built from amino acid building
blocks to again form proteins, some of which are able to replicate DNA and RNA, thus live is able to self replicate and multiply and therefore being self sustaining.

The organelles and all the other parts of the cell are tightly packed inside the cell, giving an unique environment. Measuring physical properties of this conglomerate is challenging due to active chemical processes running all the time and the vast count of different proteins interacting with each other mechanically and chemically.

The cells need to react to their environment, as the organelles mainly react to the chemical changes. The cytoskeleton allows cells to adapt to mechanical changes in the environment. It allows, like a skeleton, the cell to stiffen itself and with the use of motor proteins to actively change the cell shape and therefore do mechanical work. This allows cells to migrate and – with the help of focal adhesions build out of integrines – attach to each other, forming more complex structures and building macroscopic living beings like animals, plants and fungi.

The cells used in this work are NIH/3T3 fibroblast cells from Swiss albino mouse embryo tissue. Fibroblasts primarily produce collagen for the extracellular matrix. The cellular matrix is important for organism, as it stores water, transduces signals between cells, helps with wound healing among others. The cell line was established in 1962 [TG63] and is since used as a standard cell line (e.g. [AHC14; Lei+13]).

### 3.1.2 Cytoskeleton

The cytoskeleton is responsible for the majority of the mechanical properties of cells. It can be divided into three main components: The actin filament system, the intermediate filament system and the microtubuli. A fluorescently labeled cytoskeleton and nucleus of a cell is shown in figure 3.1.

#### Actin filament system

The actin filament system responsible for the mechanical behavior is the filamentous-Actin (F-actin), which consists of many strands of globular-actin (G actin), which is pairwise assembled and forms a helical structure. An F-actin monomer has a typical length of 290 nm [CLZ91] and a diameter of 5 nm to 9 nm [Alb02]. Actin can rapidly assemble and disassemble. Additionally, motor proteins can attach to actin fibers, moving them
Figure 3.1: Fluorescent labeled cell cytoskeleton and nucleus. F-actin is labeled red (Texas Red X-Phalloidin), tubulin is labeled green (Bodipy FL goat anti-mouse IgG Red), the nucleus is labeled blue (DAPI). The intermediate filaments are not labeled, as they are distributed throughout the cell, and labeling them would make it difficult to observe the other structures. [Nat]
against each other, making it the part of the cytoskeleton which can actively change its shape (thus the name actin). Therefore they are mainly responsible for cell shape changes and migration. In muscle cells they work together with myosin to allow these cells to contract.

Actin is mainly present in the outer regions of the cell and plays a major role in connecting the cell’s interior to the outer world by adherens junctions and focal adhesions.

**Intermediate filament system**

The intermediate filament system consists of polymers with a diameter of approximately 10 nm [Alb02]. They have a persistence length of 300 nm to 1000 nm [Mü+04]. These values lie in between the values for actin and the microtubuli persistence lengths, hence the name “intermediate”. In cells they tend to form cross-linked structures with an increased persistence length of up to 1 mm. As they form a dense mesh within the cell, they help the cell to maintain the general shape and hold everything in place. At high mechanical stresses the intermediate filament deforms plastically, while on small stresses they deform elastically. The intermediate filament polymerizes at the cell membrane and depolymerizes at the cell nucleus [Win+11]. The intermediate filament was also studied in the past in this institute [Mar+15; Nec+16].

**Microtubuli**

The microtubuli are the elephants of the cell cytoskeleton by having a persistence length of up to several milimeters [Pam+06] due to their thickness of 25 nm [Alb02] and tubular structure (hence the name microtubuli). Microtubuli are formed in the microtubulus organizing centers (MTOCs) and stretch radially outwards to the cell boundaries. They are mainly used as highway to transport proteins within the cell and do not stretch to the rim of the cell.

**3.2 Microrheology**

Rheology originating from old greek “πέρα” (flow) and “γράφω” (study of) is “the study of deformation and flow of matter” [Wik20] [BHW93, p. 1]. Microrheology is therefore the
study of deformation and flow of matter in a small sample volume. Typically, microrheology is done using a microindenter like an atomic force microscope (AFM) [Rot+14] or by observing [Miz+08] and/or applying forces on beads embedded in the material embedded (e.g. polystyrene beads [DW05] or magnetic beads [WGB03]). The methods used in the field of microrheology can be grouped into two categories, active and the passive methods. While on the passive method the movement of the tracing particles is mostly due to thermal fluctuations and therefore Brownian, active methods try to increase the signal-to-noise ratio by actively displacing the beads and/or the material. Some of the typically used techniques are presented in the following.

### 3.2.1 Active microrheology

The methods in the class of active microrheology are either mechanical [Rot+14], optical [Miz+08], or magnetical [WGB03]. These methods can be combined for the force application and the feedback readout [WGB03]. Each has its distinctive advantages and disadvantages, which are discussed in the following after giving a short introduction of the underlying mathematical framework.

**Mathematical framework**

All active techniques exert a force on parts of the sample with a known amplitude, direction, velocity and frequency. The modulation of this forces is not limited to sinusodial excitation, but for this theoretical overview this is assumed without loss of generality. The following is based on [GVW05; Pau13]. Assuming a spherical particle with the radius $a$ is embedded in a viscous, frequency dependent medium with the viscosity $\eta(\omega)$ and elasticity $k(\omega)$. A simple equation of motion is then formed by

$$m^* \ddot{x} + 6\pi \eta^* a \dot{x} + (k_E + k) x = k_E A \cos \omega t$$  \hspace{1cm} (3.1)

with the left part being the response of the system to the periodic driving force of the right side. $m^*$ and $\eta^*$ denote effective mass and viscosity after correcting for the inertia effects of dragging the surrounding medium with the particle. With a density of $\rho$ of the surrounding medium and $m$ the mass of the particle, the effective mass is given as

$$m^* = m + \frac{2\pi}{3} a^3 \rho + 3\pi a^2 \sqrt{\frac{2\eta \rho}{\omega}}$$  \hspace{1cm} (3.2)
and $\eta^*$ is

$$\eta^* = \eta \left(1 + \sqrt{\frac{a^2 \rho \omega}{2\eta}}\right).$$  \hfill (3.3)$$

This model assumes that the particle is coupled to the driving system with a spring with the spring constant $k_E$.

Equation 3.1 has a steady state solution

$$x(t) = D(\omega) \cos(\omega t - \delta(\omega))$$  \hfill (3.4)$$

with $D$ being the response amplitude and $\delta$ the phase shift due to the surrounding medium. Both can be described by the coupling constant, the effective mass and viscosity

$$D(\omega) = \frac{k_E a}{\sqrt{(k_E + k(\omega) - m^* \omega^2) + m^* \beta^2 \omega^2}}$$  \hfill (3.5)$$

$$\delta(\omega) = \tan^{-1} \left(\frac{m^* \beta \omega}{k_E + k(\omega) - m^* \omega^2}\right)$$  \hfill (3.6)$$

$$\beta = \frac{6\pi \eta^* a}{m^*}.$$  \hfill (3.7)$$

$D$ and $\delta$ can be measured in the experiment by applying sinusoidal external stimuli at various frequencies. This determines the storage $G'$ and loss modulus $G''$ [HOY99] and together the complex shear modulus $G^*$

$$G'(\omega) = \frac{k(\omega)}{2\pi a}$$  \hfill (3.8)$$

$$G''(\omega) = \omega \eta(\omega) = \frac{k_E \sin \delta}{6\pi a D}$$  \hfill (3.9)$$

$$G^* = G' + iG''.$$  \hfill (3.10)$$

$G'$ describes the elastic response of the material to external shear. While shearing energy is stored and released when 'unshearing', thus the term storage modulus. $G''$ describes the viscous response. Energy going into this term is mechanically lost and normally simply heats the material, thus the name loss modulus. For non-Newtonian fluids this equations might need a modification as they assume a linear response over the whole shearing range and speed.
These equations apply nearly universal for the different kinds of probes used in active microrheology. Modifications are needed for non-linear probe behavior. Commonly the following three probe types are used.

**Mechanical probes**

Mechanical microindenters (e.g. AFMs) can be used to probe the surface of materials to gain insight into their mechanical properties [Rig+17]. By rastering the probe over a grid, this information can be obtained over the entire surface [Hag+00; Fuh+11]. Mechanical microindenters have the main disadvantage that it is not possible to only probe the interior of a living cell as there is no way to get the probe tip into the sample without destroying it. Also the AFM tip directly interacts physically and chemically with the cell, which might induce a bioresponse in the cells [You+00].

**Optical probes**

Optical probes including optical tweezers use the photon pressure and impulse transfer by refraction to exert forces on a particle [Pau13]. If the particle has a higher refractive index than the surrounding medium, it is pulled towards higher light intensities and can be trapped in a highly focused light spot, creating an optical trap or optical tweezers. The trapped particle can be used to probe the surrounding medium similar to an AFM by steering the focus spot and has the advantage to be placed everywhere in the sample including inside of living cells. The disadvantage is the high intensity light, possibly inducing phototoxicity and heating cells. Additionally, the technical complexity when trapping multiple particles at once are very high.

Instead of catching a particle in a trap and probing the surrounding, an optical contrast in the sample can also be used to exert a stimulus. This has the advantage of not needing a probe to exert the force. This effect is used with optical stretchers [Guc+01; Nec18], stretching an entire cell. To read out the response of the sample, contrast analysis, video microscopy or particle tracking with apparent radius estimation can be used.

**Magnetic probes**

A force can also be applied by embedding magnetic beads within the sample and applying a magnetic field [WGB03]. To read out the response of the sample, either the magnetic beads themselves or other features of the sample like incorporated non-magnetic beads or cell features can be used. This is normally performed by optical means with a video camera. This method has the advantage of the possibility to spin the magnetic
beads, therefore introducing a torque as a measurement force. The main disadvantage is the need to combine an optical setup with a magnetic one.

### 3.2.2 Passive microrheology

In contrast to active microrheology, where the response of a well defined stimulus from the outside is analyzed, the passive microrheology observes the response of a statistically random force by the thermal noise of the specimen, called Brownian motion. Only this motion is observed and analyzed. As the Brownian motion is only practically observable for small objects, passive microrheology is only working on microscopic specimen. The following description is taken from [MW95; Mar16; GVW05].

The most important statistical metric for passive microrheology is the mean squared displacement (MSD) computed as

\[
\text{MSD}(\tau) = \langle \Delta \vec{x}^2(\tau) \rangle = \left\langle |\vec{x}(t+\tau) - \vec{x}(t)|^2 \right\rangle_t
\]  

(3.11)

with \(\vec{x}\) being the position of the particle in \(d\) dimensions, \(\tau\) being the so-called lag-time or time delay, and \(\Delta\) denotes discrete time steps. \(\langle \rangle_t\) symbolizes averaging over the time \(t\). Averaging over \(t\) is only allowed if the system is in a thermal and mechanical equilibrium and the material properties of the surrounding does not change.

This MSD can be directly correlated to the diffusion equation via

\[
\langle \Delta \vec{x}^2(\tau) \rangle = 2dD\tau
\]

(3.12)

with \(D\) being the diffusion coefficient. Using a spherical bead with radius \(a\) and assuming a non-slip boundary condition, the viscosity of the surrounding medium can be obtained using the Stokes-Einstein-equation

\[
D = \frac{k_B T}{6\pi \eta a}
\]

(3.13)

if the temperature \(T\) is known.
More complex materials than simple Newtonian fluids have non-linear viscous and elastic properties that might change with frequency and strain. For those materials, the MSD of embedded particles reflects both the elastic and the viscous properties. Most often, the MSD no longer scales linearly with the lag-time $\tau$, but follows a power law behavior

$$\langle \delta \mathbf{x}^2(\tau) \rangle \sim \tau^\alpha. \quad (3.14)$$

$\alpha$ is called the diffusive component ($\alpha < 1$). If $0 < \alpha < 1$, the particle exhibits sub-diffusive motion. $\alpha = 0$ is a sign of a locally constrained particle. The case $\alpha > 1$ is called super-diffusive behavior, which might happen e.g. in modified random walks [Bou+90], in geometrical confined crowded systems [Bé+13], biological systems [MK+18] and within cells [Rev+15].

If the material is completely elastic, an embedded bead will encounter a plateau for the MSD for long time steps

$$\langle \Delta \mathbf{x}^2(\tau \to \infty) \rangle = \frac{k_B T}{\pi G' a} \quad (3.15)$$

with $G'$ being the elastic modulus of the material.

Analogous to an harmonic oscillator, the elastic energy of the network can be modeled using a spring with a spring constant proportional to $G' a$ and the energy in the spring is $k_B T$. The elastic responses can be integrated into the Stoke-Einstein equation by modeling it with a complex shear modulus $G(\omega) = i\omega \eta$ [GVW05; LL00]. The forces on a small particle with mass $m$ embedded in this network with a (relative) velocity $v(t)$ can be expressed using a generalized Langevin equation

$$m \dot{v}(t) = f_R(t) - \int_0^t \zeta(t - \tau)v(\tau)d\tau \quad (3.16)$$

with $f_R$ representing all forces on the particle, including the inter-particle forces and the Brownian motion. The integral represents the damping of the material which needs a time-depended memory function $\zeta(t - \tau)$ to model the elasticity. By using the equipartition theorem and the Laplace transform it can be related to the autocorrelation function

$$\langle v(s)v(0) \rangle = \frac{k_B T}{\zeta(s) - ms} \quad (3.17)$$
with \( s \) denoting the frequency in the Laplace domain. Usually, the term \( ms \) is small, as it represents the ballistic inertial term. Thus it can be omitted for frequencies \( \ll 1 \text{ MHz} \) [LL00] yielding

\[
\langle v(s)v(0) \rangle = \frac{k_B T}{\zeta(s)} \tag{3.18}
\]

\[
\zeta(s) = \frac{6k_B T}{s^2 \langle \Delta\tilde{r}^2(s) \rangle} \tag{3.19}
\]

with \( \langle \Delta\tilde{r}^2(s) \rangle \) denoting the MSD in the Laplace domain. Modifying Stokes law to include complex viscous elements [MW95], the complex shear modulus \( \tilde{G} \) can be related to \( \zeta \) via

\[
\tilde{G}(s) = \frac{s \zeta(s)}{6\pi a} \tag{3.20}
\]

By combining equation 3.18 and 3.20, a relationship between the bead motion and the bulk properties of the investigated material can be formed

\[
\tilde{G}(s) = \frac{k_B T}{\pi a s \langle \delta\tilde{r}^2(s) \rangle} \tag{3.21}
\]

As discussed in [GVW05, p. 23ff] it is not trivial to perform the inverse Laplace transform to accurately calculate \( \tilde{G} \). A variety of methods to compute this has been developed [MW95; Eva+09; Pau13].

### 3.3 Particle tracking

All optical microrheology methods presented in chapter 3.2 need a way to identify and track particles with a speed faster than the characteristic frequency of the material. Tracking of the particles needs to be as precise as possible to improve the measurement result. The current state-of-the-art techniques to measure particle positions can be split into two categories. The first category consists of intensity-based methods using four-quadrant diodes or higher resolution sensors and the scattered light of a particle. These diodes and sensors can be sampled at high speed and are therefore suitable for high frequency analysis. In addition, the position can be determined with little delay while the experiment is running and it is possible to react to changes and hence to control the
experiment. The main disadvantage is the limitation to the tracking of only one particle or the employment of more complex time multiplexing tools using an acusto-optical-modulator (AOM) or other suitable tools. Furthermore the particles must be limited to a small region of the sample to reach them. This can amongst others be achieved by the use of an optical trap, with all its advantages and disadvantages.

The second category is the video camera analysis method. A high-speed camera records the experiment through a microscope. Afterwards the video is analyzed with a tracking software to detect the particles and their position in the video. The main advantage is the almost unlimited number of particles that can be tracked. The limit is given by the density and the field of view of the used optics. The main disadvantage is the limited sampling rate of typically less than 10 kHz for reasonably expensive sensors and lower light sensitivity of the camera compared to a diode. Furthermore, most high-speed cameras can only record for a limited time as the image data needs to be buffered before it is transferred to permanent storage. This limits the frequency bandwidth that can be observed simultaneously. Additionally, it is not possible to react to changes as the particle position is computed “offline” after the experiment has been conducted.

One way to improve the video camera approach is a system that analyzes the acquired images from the video camera directly and computes the particle positions. This eliminates the need to save the video frame and reduces the buffering requirement. Such a system must be capable of analyzing all selected particles visible in a picture with the speed of the camera system, requiring an analysis duration in the range of 100 μs per frame. This limits the possible particle tracking algorithms to those that are suitable for a fast implementation. This has been done before for single optically trapped particles [Tas+12; Gib+08; Ott+10], but not for multiple non-trapped particles at speeds up to 10 kHz or use time multiplexing and speed variation techniques [Phi+16].

Some tracking algorithms that are suitable for fast particle tracking are discussed in the following sections. They differ in the complexity and in the accuracy. The optical system normally does not project the particles onto a single pixel. This gives the possibility to calculate the center of the particles with a higher precision than a single pixel.

The particle tracking algorithm needs to fulfill multiple goals:

- It needs to calculate the particle center with high precision
• It should be stable for image features near the particle, which are not part of the particle

• It should be robust against noise. This includes shot noise from the pixels, readout noise from the analog-digital-converter (ADC), and noise from the photon statistic

3.3.1 Peak finding

Fast improvements can be achieved by calculating a sub-pixel extremum by using the brightness of the neighboring pixels. Assuming \( I(x, y) \) is the intensity of the pixel at the position \( x, y \) and \( x_0, y_0 \) is the position of the extremal pixel, then the sub-pixel peak components \( x_s, y_s \) can be calculated in the first order using

\[
\begin{align*}
  x_s &= \frac{I(x_0 - 1, y_0) - I(x_0 + 1, y_0)}{I(x_0 - 1, y_0) + I(x_0 + 1, y_0) + 4I(x_0, y_0)} \quad (3.22) \\
  y_s &= \frac{I(x_0, y_0 - 1) - I(x_0, y_0 + 1)}{I(x_0, y_0 - 1) + I(x_0, y_0 + 1) + 4I(x_0, y_0)} \quad (3.23)
\end{align*}
\]

This is a simple and fast method but only uses 5 pixels in total to compute the position, which makes it sensitive to noise and background inhomogeneities even with the use of heavy smoothing. This can be improved by incorporating more surrounding pixels into the computation. The main disadvantage of this algorithm lies in the single pixel peak search. A particle that appears as a ring has many pixels suitable for the extremum. This will – in combination with noise – introduce a jumping of the apparent center, making this method unsuitable for the general case.

3.3.2 Centroid of intensity

An estimation of the particle center can also be computed by mimicking the approach of the four quadrant diode. Therefore, the integral mean \( \bar{x}, \bar{y} \) of the image \( I(x, y) \) (size \( M, N \)) is computed in two directions

\[
\begin{align*}
  \bar{x} &= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} i I(x, y) / \sum I(x, y) \quad (3.24) \\
  \bar{y} &= \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{M} j I(x, y) / \sum I(x, y). \quad (3.25)
\end{align*}
\]
This method will only yield the absolute center position when the background of the image is black. This can be achieved by using a suitable setup (e.g. dark field or phase contrast microscopy) or pre-processing of the image and subtracting the background from it. Also the position is computed inverted if a black particle is tracked in front of a white background.

To track multiple particles, every particle needs to have its own image area. This might induce problems when having an inhomogeneous background. When the image areas are moved with the particles, the apparent particle position might shift considerably depending on image features included in the areas. As the image region for a particle might move with the particle or other features of the image might change while recording the video, this is prone to introduce artifacts.

### 3.3.3 Surface fitting with 4th degree polynomial

An important improvement of tracking precision can be achieved by fitting a function to the particles image. The clear advantage compared to the peak finding algorithm is the inclusion of many more image data points, making it very robust against noise. It also has the advantage over the centroid approach by being much less susceptible to image features not directly at the particle position. Additionally by using suitable functions it is possible to extract further parameters of the particle.

One commonly used approach is described by [Rog+07]. The algorithm pre-processes the image by applying a smoothing step and inverting the intensity if dark particles should be detected. The smoothing makes the algorithm less susceptible to noise and sharp borders, which might hinder a good convergence of the algorithm. The inversion is not necessary for the core algorithm, but is required by the starting particle search algorithm to determine the initial position and radius estimations. This initial search calculates a rough estimation of the particle center by peak-finding (see 3.3.1) and searching for the first inflection (change of sign in the first derivative) along the \( x \)- and \( y \)-axis as a rough radius estimate. Then an iterative fine search is performed by fitting the
3.3. Particle tracking

Polynomial $P$ of fourth degree described by

$$P(x, y) = p_{00} + p_{10}x + p_{01}y +$$

$$p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3 +$$

$$p_{40}x^4 + p_{31}x^3y + p_{22}x^2y^2 + p_{13}xy^3 + p_{04}y^4$$

(3.26)

with the fitting coefficients $p$. The polynomial is fitted against the surface using Gaussian weights $w$ correlated to the estimated radius $r$ of the particle with

$$w(x, y) = \exp \left( -\frac{x^2 + y^2}{2\sigma r^2} \right).$$

(3.27)

$\sigma$ is a factor determining how much of the image beyond the assumed radius is used for the computation of the new position and radius.

The fitting consists of minimizing the squared error of

$$P(x - x_0, y - y_0) = I(x, y)$$

(3.28)

while weighting the individual pixels of the image $I(x, y)$ by the weights $w(x - x_0, y - y_0)$.

The estimated pixel position $x_n, y_n$ is computed by using the extremum of the second degree coefficients with

$$J = p_{20}p_{02} - \frac{p_{11}^2}{4},$$

(3.29)

$$x_n = \frac{p_{11}p_{01} - 2p_{02}p_{10}}{4J},$$

(3.30)

$$y_n = \frac{p_{11}p_{10} - 2p_{20}p_{01}}{4J}.$$  

(3.31)

It is also possible to compute the pixel position using the higher degree factors, but this normally does not improve the performance and precision, as commented in the source code of the accompanied program of [Rog+07].

The new radius $r_n$ can be computed by using the “geometric mean of the distance from the peak [...] to the four points of inflection in the particles major and minor axes”
[Rog+07]. Given the angle $\theta$ it is computed the following

$$A = p_{20} \cos^2 \theta - p_{11} \cos \theta \sin \theta + p_{02} \sin^2 \theta$$

$$B = p_{20} \sin^2 \theta + p_{11} \sin \theta \cos \theta + p_{02} \cos^2 \theta$$

$$C = p_{40} \cos^4 \theta - p_{31} \cos^3 \theta \sin \theta + p_{22} \cos^2 \theta \sin^2 \theta - p_{13} \cos \theta \sin^3 \theta + p_{04} \sin^4 \theta$$

$$D = p_{40} \sin^4 \theta + p_{31} \sin^3 \theta \cos \theta + p_{22} \sin^2 \theta \cos^2 \theta + p_{13} \sin \theta \cos^3 \theta + p_{04} \cos^4 \theta$$

$$r_n = \sqrt[4]{\frac{AB}{36CD}}.$$  

(3.32) (3.33) (3.34) (3.35) (3.36)

To improve the radius and the position estimation, the computation is repeated with $x = x_0 + x_n, y = y_0 + y_n, r = r_n$ until the parameters converge. The accompanied program of [Rog+07] also does two final rounds of fitting when the set threshold of change is no longer reached.

Further parameters can be computed from the fitting coefficients. The estimated eccentricity $e$ at the angle $\theta$ and the skewness $s$ of the particle is computed using the third and fourth degree components

$$e = \frac{1 + \sqrt{(P_{20} - P_{02})^2 + p_{11}^2} + (p_{20} + p_{02})}{\sqrt{(P_{20} - P_{02})^2 + p_{11}^2} - (p_{20} + p_{02})},$$

$$\theta = \frac{1}{2} \cot^{-1} \frac{p_{20} - p_{02}}{p_{11}}.$$  

$$s = \frac{|p_{30}| + |p_{21}| + |p_{12}| + |p_{03}|}{J}.$$  

(3.37) (3.38) (3.39)

Interestingly, the accompanied program of [Rog+07] uses a different approach to compute the eccentricity $e'$. Defining the variables $a, b, c, d, f, g$ and $J$ as

$$a = p_{20}, b = \frac{p_{11}}{2}, c = p_{02}, d = \frac{p_{10}}{2}, f = \frac{p_{01}}{2}, g = p_{00}$$

$$J = ac - b^2.$$  

(3.40) (3.41)
two values $s_1$ and $s_2$ are computed via

$$s_1 = \frac{2 \left(af^2 + cd^2 + gb^2 - 2bdf - acg\right)}{-J \left((c-a)\sqrt{1 + \frac{4b^2}{(c-a)^2}} - c - a\right)},$$

(3.42)

$$s_2 = \frac{2 \left(af^2 + cd^2 + gb^2 - 2bdf - acg\right)}{-J \left((a-c)\sqrt{1 + \frac{4b^2}{(c-a)^2}} - c - a\right)},$$

(3.43)

The eccentricity $e'$ is finally calculated as

$$e' = \sqrt{1 - \frac{\min(s_1, s_2)}{\max(s_1, s_2)}},$$

(3.44)

For this thesis, the second approach using $e'$ is used to compute the eccentricity. $e'$ is 0 for a perfect circle and 1 for an infinitely elongated ellipse. The minor radius $r_m$ of an ellipse with an apparent and major radius $r$ and an eccentricity $e'$ can be calculated as

$$r_m = 2r - \frac{\sqrt{1 - e'^2}}{1 + \sqrt{1 - e'^2}},$$

(3.45)

To completely automate a particle tracking process it is possible to use a peak finding algorithm using a heavily smoothed version of the image and use the inflection point of the first spatial derivatives in the horizontal and vertical direction as an estimate for the radius.

The approach by [Rog+07] as described here has the advantage of providing more parameters – most importantly the radius $r$, the eccentricity $e$, the skewness $s$ and the particle orientation $\theta$ – besides the position of a particle to allow further discrimination and analysis of these particles. Also by the computation of the radius this algorithm can also be used to estimate the height of the particle relative to the focus height of the microscope after calibration [Nec18]. All this is possible by using a sophisticated algorithm which needs significantly more computing power than the other algorithms described in this thesis.

As this algorithm gives very precise position estimates and is very robust against complicated background [Rog+07] it is a good choice for particle tracking. This algorithm is therefore used as the basis to improve and evolve to a real time particle tracking algorithm.
4 Sensor

The sensor is a custom-developed system which allows completely autonomous operation. This is necessary because of the tight timing and latency requirements, making it very challenging to use external hardware. It also reduces the used space around the setup, thus giving more freedom to the experimenter to incorporate the sensor in existing setups.

4.1 Hardware

The hardware consists mostly of a Xilinx field programmable gate array (FPGA) and an On Semiconductor CMOS imaging sensor.

4.1.1 Imaging sensor

The sensor used for this project is a PYTHON1300 (NOIP1SN1300A-QDI, Semiconductor Components Industries LLC). The sensor is a CMOS imaging sensor with a pixel pitch of 4.8 μm and a resolution of 1280 × 1024 pixels. It is configured by an SPI interface at maximum 12 MHz and sends the image data over 6 LVDS pairs at a clock rate of 360 MHz. The LVDS is composed of one differential clock channel, one status channel and 4 data channels. The status channel and the data channels transmit 10 bit words in the DDR mode. This yields a pixel rate of 288 Mpix/s.

The sensor needs an external clock which can be configured to be either a 3.3 V TTL clock with 72 MHz or a LVDS 360 MHz clock. The TTL clock is converted to the 360 MHz clock by an internal PLL, thus reducing the jitter requirement of the clock itself. The PLL clock is used, as it can be easily generated by one of the internal FPGA clock generator blocks.
An important feature of this sensor is the possibility to configure up to 8 different regions of interest (ROI), which can be nearly freely chosen. The sensor frame rate increases significantly with a smaller readout area. In the fastest mode (“ZROT” mode), a full frame image readout can be done in 4.5 ms yielding 220 FPS. Limiting the readout area to $256 \times 256$ pixels, the maximum obtainable framerate is 2235 FPS, or a roughly tenfold increase. Reducing the readout region further gets limited by the minimal readout time of $1.17 \mu s$ needed per row and a fixed overhead per frame of $44 \mu s$. This also implies that there is no further advantage by reducing the row length below 160 pixels. The sensor has a kernel size of $8 \times 1$. So it reads out 8 pixels in the $x$-direction at the same time. This also limits the granularity of the readout region to multiples of 8. In the $y$-direction this granularity is 1.

To achieve high frame rates, it is therefore necessary to limit the number of lines to read out. The length of these lines has a small impact on the maximum speed of the sensor. The sensor has a global shutter with very short programmable shutter times down to $1 \mu s$. The sensor supports parallel readout and illumination, increasing the speed and decoupling the illumination time from the maximum obtainable speed to some extent. If the illumination time is increased towards the frame time, the frame time limits the speed.

To obtain a good compromise between sensor readout speed and trackable particle sizes, the ROI size per particle was fixed at $64 \times 32$ pixels, theoretically a size up to 160 pixels would be possible without loss of imaging speed, but then the requirement of internal SRAM increases. Also often particles lie grouped together and therefore the actual readout length of a lines increases, thus then limiting the readout speed faster. This allows the tracking of particles of up to 20 pixels apparent diameter without information loss at up to 10 kHz. In theory, this size can be changed, but increasing the size will likely exhaust the available block memory (BRAM) resources while a smaller size will only return minute gains in sensor speed and will reduce the possible sizes of tracked particles to an apparent sizes of less than 10 pixels. The ROI size was deliberately chosen asymmetrical as the granulary in $x$- is more coarse than in the $y$-direction. This mimics the extension of a rectangular readout region with the size of $30 \times 30$ pixels with a padding of one kernel in each direction. The performance degradation by the larger size in $x$ is much smaller than in $y$, costing nearly no performance, thus this asymmetrical arrangement has no further disadvantages.
4.1.2 Processing unit

The processing unit is the Zynq system on a chip (SOC), the chosen SOC is the XC7Z020CLG400-1 (Xilinx Inc.) mounted on the MicroZed development board (Avnet Inc.). It is paired with 1 GB of DDR3-1066 memory with 32 bit bus width. All programs including the PL bitstream are stored on a microSD card (128 GB). The microZed includes a USB 2.0 PHY for the USB controller of the SOC and a Gigabit Ethernet SGMII PHY for the Ethernet controller in the SOC. These two interfaces are used to control the sensor and store the obtained data on permanent storage (e.g. a USB hard drive or a network drive).

The SOC itself contains a processing system (PS) mostly consisting of two ARM Cortex-A9 cores running at 667 MHz executing the Linux kernel and programmable logic (PL). The PL has 220 DSP units (25 bit pre-adder, 25x18 multiplier and 48 bit adder), 140 full dual ported SRAMs á 36864 bit, called block RAM (BRAM), 106400 flip flops, and 53200 6-input look up tables (LUTs) of which 17400 can be used as 64 bit RAM blocks. The PL can reasonably run at frequencies up to 400 MHz, but a good sweetspot to allow the synthesis tools to reach the required timing margins is 100 MHz. The theoretical performance of the PL-DSPs at 400 MHz is 88 GMAC/s. Communication of the PL and the PS can be done via some general purpose IO (GPIO) pins, 4 High performance AXI interfaces (PL master) to the RAM, 2 general purpose AXI interfaces (PL master) to the central interconnect of the PS, and two general purpose AXI interfaces (PL slave). Additioanlly, most of the peripheral controllers can be mapped to PS-PL pins. In this design, it is done with two I²C controllers to communicate with the monitor over HDMI (EDID data) and to control a real-time clock (RTC).

4.1.3 Custom interfacing board

The microZed brings out most of the SOC programmable pins on two 50 pin high-speed ports. On this a custom developed PCB is mounted, containing the power supply for the microZed (5 V, 2 A) and the IO banks (2.5 V, 100 mA for the LVDS, 3.3 V 100 mA for all other signals). The layout of the custom board is shown in figure 4.1.

The custom PCB contains the following components:
4.1. Hardware

Figure 4.1: Labeled PCB layout of the custom designed board. The socket for the sensor is placed slightly off-center as the active sensor area is also not centered, making the sensor area centered. The GPIO port can be programmed freely and can be used to implement further inputs or outputs to the sensor allowing to control the experiment with the sensor. Currently it is used to trigger a pulsed LED for very short, very bright illumination. The PCB was designed by and printed with permission from Dr. Daniel Geiger.

- A power control IC, allowing to shut down the sensor by software, starting it with a button and force turning off by pressing the button for 3 seconds.

- Power supplies:
  - 5 V direct pass through to the microZed
  - 2.5 V buck converter for Bank 34 of the SOC
  - 3.3 V buck converter for the sensor
  - 3.3 V buck converter for Bank 35 of the SOC
  - 3.3 V LDO supply for the Vpix power of the sensor
  - 1.8 V LDO from the 3.3 V buck converter for the LVDS power supply of the sensor

- An HDMI port, the TMDS lines are connected to the SOC by a HDMI mux, having a high ohmic behavior when shut down. Otherwise, the monitor will backpower the SOC with the 50Ω pullup termination.

- An RTC to allow for timestamping of measurement files.
• A DIN-6 plug for the control of additional devices, e.g. a pulsed illumination for reduced light induced cell damage of the samples

• 4 status LEDs

• The sensor

The PCB was designed by Dr. Daniel Geiger and the components were placed by hand and reflow soldered. The complete hardware is placed in a custom enclosure provided by Sensific GmbH. This ensures that no stray light leaks onto the sensor and also provides cooling of the SOC.

The sensor hardware can also be used for different measurement applications by reconfiguring the PL. An example for a microfluidic sorting setup is given in [Fre+20].

4.2 Software

The algorithm presented in chapter 3.3.3 needs to be executed at least as fast as one complete convergence per frame. The fastest frame-rate that can be achieved is 10.016 kHz, leaving only 99.8\,\mu s time to run the complete algorithm. This requires a fast solution, which is possible by using the optimizations laid out in section 4.3 and a hardware design realized in the PL.

The complete user interface is implemented with Ubuntu Linux (Ubuntu 18.04 bionic beaver, Canonical) and Linux 4.19 (https://github.com/xilinx/linux-xlnx, Linus Torvalds). A screenshot of the interface with some of the functions labeled is shown in figure 4.2.

The software consists of two parts. The first part is the user interface. The user can use a mouse, a keyboard and a full HD HDMI screen to configure every aspect of the device. The second part is the particle tracking system running completely in the PL to avoid any unforeseen latencies and timing uncertainties. Also the data paths to the PS are too slow to fit into the small timing window.

The software running on the ARM controls a MicroBlaze softcore running in the PL. This is done by a BRAM block shared by both the data LMB port and an AXI BRAM Controller connected to the GP0 port of the PS-PL interface. Various other functions are controlled by PL mapped pins of the GPIO controller. Some of them are directly routed
Figure 4.2: Screenshot of the running sensor. The live view with 6 dots on notes tracked is on the left. The controls are on the right. The rightmost in blue tracked particle is slightly deformed. The sensor is detecting this deformation and drawing the tracking circle as an ellipse. The two particles on the upper left and the two on the lower left overlap with the readout region. The sensor and the software takes care that this is possible and works as expected. As long as the particles do not touch each other, the ROIs will be tracking their separate particles even when their areas overlap.
to the PL output pins to control the power regulators. Some of them are wired internally to various functions. In addition, there is a scatter-gather direct memory access (DMA) implemented in the PL to facilitate the storage of the computer parameters. After initial configuration the DMA is controlled by lists of control blocks which are written by the software. The PL also contains a serial peripheral interface (SPI) interface which controls the sensor and is used to upload the firmware to the sensor.

### 4.2.1 User interface

The user interface consists of a Gtkmm application running under the X11 windowing system with lightdm as display manager and LXDE as the compositor. The interface is best used with a mouse but can also be used with a keyboard. Particles are marked initially on the sensor image by either entering the coordinates and the size manually or by clicking the “Mark particle in picture” button followed by clicking and holding to the center of the particle to drag and release the mouse at the rim of the particle. A click on the “track” check box will now track the particle. If the particle is expected to move outside of the active window region marked by the rectangle, the user should also click the “follow” check box to advise the PL subsystem to follow the particle. When the button “fullscreen” is checked, the full sensor area is read out, giving the operator a good overview of the experiment but this setting limits the speed of the sensor to 220 Hz. Unchecking this check box while tracking at least one particle will limit the sensor readout to the areas around these particles. The frame rate of the sensor can be increased by sliding the “frame length” slider to the left and therefore decreasing the frame length. The brightness and the illumination time can be adjusted using the “illumination time” slider and the “gain” slider. Increasing the gain will also increase the noise of the image.

The “zoom” slider can be used to change the size of the preview to facilitate the particle marking and the observation and control of the experiment. A debug interface can also be used to simulate data presented to the sensor and write the registers of the sensor directly. A pulsed illumination is possible with a 3.3 V TTL output at pin 2 of the DIN plug at the sensor. The histogram shows the statistics of the pixel brightness of the regions currently read out. Scale bars can be displayed in the picture. After entering the magnification of the optical setup or by measuring a distance in the image (by clicking
twice in the image after clicking the “Mark in image” button) the bars will be drawn. To enhance the contrast, the view range of the preview can be limited to a range of gray values that are stretched to fill the complete black-white range. Darker pixels are colored in dark red, brighter pixels are colored in bright red.

The user can take a screenshot with the “Make screenshot” button. As the preview is rendered as an overlay, taking a screenshot with normal tools will only yield a green area in which the sensor image is displayed. With this button the user can take a screenshot of this area.

The output values $x, y$ and $r$ of the algorithm can be logged to a CSV file (comma separated, decimal point is ‘.’). If “log fitting parameters” is checked, the 15 fitting parameters $p$ are also logged to the file. If the check box “binary” is checked, all data after the first line is stored as binary little endian IEEE 754 single precision floating point. Saving the data in binary reduces the file size, the needed bandwidth and processing power.

### 4.2.2 Particle tracking implementation

The particle tracking algorithm used in the sensor is implemented as described in chapter 3.3.3 and optimized in chapter 4.3. To achieve the necessary performance, the bilinear interpolation is implemented as a fixed point interpolation with 16 bit subpixel precision and 15 bit result depth. The matrix multiplication with the precomputed matrix was transferred to a fixed point solution with 25 bit signed two-complement integers for the matrix coefficients. The maximum and relative errors including the required rescaling factors for each of the 15 parameters are shown in table 4.1.

The symmetries of the multiplication coefficients described and discussed in chapter 4.3 are used in the $x$- and $y$-direction to decrease the computing time of this step. The latency is then approximately 300 clock cycles. The clock frequency was set to 250 MHz, resulting in a latency of 1.2 $\mu$s. This speed is sufficient to be in the same latency range as the floating point computation of the new positions and the new radius $x_n, y_n, r_n$, thus further timing and frequency optimizations are not necessary.

To achieve a good resource utilization, four particles are processed together. This enables the fitting by matrix multiplication of four particles to compute $p$ while the other four are running through the computation of $x, y, r$ of the parameters $p$ part. Image acquisition
Table 4.1: Errors and scaling factors for the fixed-point matrix multiplication.
The maximum error is the largest relative rounding error for any pixel in the image. The mean error is the mean relative rounding error. The multiplication factor needs to be multiplied with the result of the matrix multiplication to get the equivalent floating point result.
Note: The values for $p_{ij}$ are identical to those of $p_{ji}$ due to symmetry.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Maximum error in %</th>
<th>Mean error in %</th>
<th>Multiplication factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{00}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td>$2.6 \times 10^{-4}$</td>
<td>$1.2 \times 10^{-09}$</td>
</tr>
<tr>
<td>$p_{10}, p_{01}$</td>
<td>$1.9 \times 10^{-2}$</td>
<td>$2.2 \times 10^{-4}$</td>
<td>$1.3 \times 10^{-10}$</td>
</tr>
<tr>
<td>$p_{20}, p_{02}$</td>
<td>$1.2 \times 10^{-2}$</td>
<td>$2.4 \times 10^{-4}$</td>
<td>$2.8 \times 10^{-11}$</td>
</tr>
<tr>
<td>$p_{11}$</td>
<td>$3.0 \times 10^{-2}$</td>
<td>$3.1 \times 10^{-4}$</td>
<td>$2.0 \times 10^{-11}$</td>
</tr>
<tr>
<td>$p_{30}, p_{03}$</td>
<td>$1.2 \times 10^{-2}$</td>
<td>$1.7 \times 10^{-4}$</td>
<td>$8.8 \times 10^{-13}$</td>
</tr>
<tr>
<td>$p_{21}, p_{21}$</td>
<td>$7.3 \times 10^{-4}$</td>
<td>$3.1 \times 10^{-5}$</td>
<td>$9.8 \times 10^{-13}$</td>
</tr>
<tr>
<td>$p_{40}, p_{04}$</td>
<td>$1.2 \times 10^{-2}$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$1.2 \times 10^{-13}$</td>
</tr>
<tr>
<td>$p_{31}, p_{13}$</td>
<td>$2.8 \times 10^{-3}$</td>
<td>$5.5 \times 10^{-5}$</td>
<td>$1.1 \times 10^{-13}$</td>
</tr>
<tr>
<td>$p_{22}$</td>
<td>$2.7 \times 10^{-4}$</td>
<td>$2.9 \times 10^{-5}$</td>
<td>$1.7 \times 10^{-13}$</td>
</tr>
</tbody>
</table>

and smoothing can be performed while the particles are tracked by the algorithm. To achieve this, the BRAM storage of the images is doubled, allowing for a ping-pong buffer and pipelined execution.

The image smoothing step is performed simultaneously in parallel and takes roughly 9600 clock cycles at 100 MHz, giving slightly over 10 kHz as the maximum frame rate. The smoothing step allows for a maximum kernel size of $13 \times 13$ but requires a mirror symmetric and separable kernel. The smoothing is written in HLS and uses 20 DSPs and 4 BRAM blocks plus 8 BRAM blocks for an additional image ping-pong buffer. The smoothing increases the latency of the tracking by one frame, making it lag 4 frames behind the acquisition. A comparison of the filter module with the Matlab `imfilter` function is shown in figure 4.3. The FPGA module yields the bitexact same values as the Matlab function, validating its functionality.

As the fitting part is very resource intensive, it is written in VHDL using many features of the DSP and BRAM that are not directly available to the higher level synthesis tools (e.g. HLS). This includes dynamic function selection and cascaded in- and outputs for the DSP and asymmetric in- and output widths and asynchronous clocking of the BRAM. It runs at 250 MHz to achieve the performance goals.
4.2. Software

Figure 4.3: Comparison of the filter functions with a random noise image, Matlab and FPGA result.

The image on the left ($32 \times 64$ px) shows random noise. In the middle is the filtered result by the `imfilter` function of Matlab with a Gaussian kernel (`fspecial('gaussian',[13 1],5)`) and the filter result of the FPGA image smoothing module is shown on the right. Both filters yield the same result.
Chapter 4. Sensor

The computation of \( x_n, y_n, r_n \) on the other hand is written in HLS, as the compiler is very good in scheduling the operations to the floating point computing units. The non-pipelined design operates on 4 particles at the same time and computes the output parameters \( x_n, y_n, r_n \) from the unscaled integer parameters \( p \) in 118 clock cycles at 100 MHz, therefore taking 1.18 \( \mu s \). To achieve this computing speed, the optimizations outlined in chapter 4.3 like optimized algorithms, a hand crafted \( \sqrt{x} \) function and manual floating point operations are implemented. A small third block is controlling and configuring the fitting and the \( x_n, y_n, r_n \) computation blocks, which takes additional 22 cycles at 100 MHz. Thus, a complete iteration of the algorithm takes 2.6 \( \mu s \).

The aim was to perform 20 iterations at 10 kHz or 100 \( \mu s \). With scheduling delays (2 \( \mu s \)) and initial copying (2.1 \( \mu s \)), the total runtime is approximately 56.1 \( \mu s \), leaving enough headroom for higher tracking speeds or more iterations. This might get limited by the overhead for the timing synchronization, which is significant and reduces the available time slot length. As the smoothing algorithm is currently limiting the speed to slightly over 10 kHz, only the iteration count can be bumped up if needed.

All timing relevant, but not performance critical processing is done by a MicroBlaze soft-core (Xilinx Inc.). This includes the mechanism described in section 4.3.4 to follow the particles with the sensor readout regions. It also controls the basic configuration and initialization of the sensor pipeline by the user.

The estimated power consumption by the synthesis tools amounts to 4 W for the complete processing system excluding memory, the sensor and further peripherals. The complete system is powered by a 5 V, 3 A barrel plug power brick. The power consumption is high enough to let the Zynq SOC reach over 90 °C. Therefore additional cooling is required. This cooling is supplied by the body, distributing the heat over the whole surface and keeping the Zynq SOC below 70 °C. The temperature difference of the body to the surrounding air is below 20 K, making it compliant with workplace safety rules.

The combined computing performance equals roughly 70 GOPS for the fitting part and 700 MFLOPS for the computation part. The smoothing requires roughly 7.8 GOPS. This does not consider small operations and memory fetches. The realized design allows 20 rounds of the tracking algorithm with up to 8 particles with a frame rate of 10 kHz, thus achieving its design goals.
4.3 Tracking algorithm optimizations

The algorithm presented in 3.3.3 is a powerful but computationally very demanding fitting based algorithm to detect particles and their positions with high accuracy. This algorithm alone is not suitable for real-time tracking as the computing demand exceeds the available power of commonly used computers and graphics cards. Therefore, if this algorithm should be applied in real-time particle tracking it has to be optimized to reduce the computing load required to a level that is achievable by modern computers.

The algorithm can be roughly grouped into two main steps:

1. Fitting the polynomial \( P \) to the image \( I \) using estimated positions \( x, y \) and radius \( r \).
   This can be formulated as a system of equations. Solving this system is discussed with potential optimizations in chapter 4.3.1.

2. Use the polynomial coefficients to compute new \( x, y \) and \( r \). This can be achieved by computing some formulas. Those have to be optimized, as shown in chapter 4.3.2. The results can be again plugged in as new estimates in the first step.

This iterative approach is executed until the estimation for position and radius converges to stable values. This needs to be executed sequentially – the dependency chain is built by \( x, y, r \) – limiting the speedup possible by parallelization.

4.3.1 Optimize fitting the polynomial to the image

Fitting a function to a dataset in a least squared error sense is generally a non trivial task, as there are no closed solutions for all type of functions. For polynomials there exists a closed form as described in the following.

Vandermonde matrix and polynomial fitting

Given a dataset of \( \bar{y} \) containing the \( y \) values of a function \( y = f(x) = p_0 + p_1 x + p_2 x^2 + \ldots \) with the supporting points in \( \bar{x} \) and the polynomial factors \( \bar{p} \), the evaluation of \( f \) can be written as
Chapter 4. Sensor

\[ y = f(x) \]  
(4.1)

\[ = \begin{pmatrix} 1 & x & x^2 & \ldots \end{pmatrix} \cdot \vec{p} \]  
(4.2)

\[ = V \cdot \vec{p} \]  
(4.3)

with "\cdot" denoting the matrix product and \( \vec{p} \) being a column vector. If multiple points are known, this can be written as

\[ \vec{y} = V \cdot \vec{p} \]  
(4.4)

\[ = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots \\ 1 & x_2 & x_2^2 & \ldots \\ \vdots & \vdots & \ddots & \ddots \end{pmatrix} \cdot \vec{p} \]  
(4.5)

where \( V \) is called a Vandermonde matrix [Kal84].

This is a matrix-vector problem of the form \( A\vec{x} = \vec{b} \) for which multiple solutions exist to solve this equation for \( \vec{x} \). If the dimension of \( \vec{p} \) equals the dimension of \( \vec{y} \) and the values in \( \vec{x} \) are pairwise different, then the matrix \( V \) is regular and it exists a single exact solution as the determinant is not zero [Kal84]. If the dimensions are different, it is either an over- or an under-defined system of equations which needs to be solved. For images, this system of equations is typically over-defined.

The solvers for the over-defined case normally search for a least squared error solution, thus minimizing

\[ e = \sum_i (f(x_i) - y_i)^2. \]  
(4.6)

If certain supporting points and values should be prioritized, e.g. in particle tracking the points around the particle center. This can be achieved with a weighting vector \( \vec{w} \).
containing the weight for each point value pair. \( w_i \) is multiplied to the function

\[
w_i y_i = w_i f(x_i) \quad (4.7)
\]

\[
e = \sum_i \left( w_i f(x_i) - w_i y_i \right)^2 \quad (4.8)
\]

and thus the error function prioritizes the values with higher weight factors. It should be noted, that in this case the weight factor will be squared in the error function. If linear weighting is needed, \( w_i \) needs to be substituted with \( w'_i = \sqrt{w_i} \). Now, the Vandermonde matrix also needs to incorporate the weights

\[
\vec{w} \odot V \cdot \vec{p} = \vec{w} \odot \vec{y} \quad (4.9)
\]

\[
\begin{pmatrix}
  w_1 \\
  w_2 \\
  w_3 \\
  \vdots
\end{pmatrix} \odot
\begin{pmatrix}
  1 & x_1 & x_1^2 & \cdots \\
  1 & x_2 & x_2^2 & \cdots \\
  \vdots & \vdots & \vdots & \ddots
\end{pmatrix} \cdot \vec{p} = \vec{w} \odot \vec{y} \quad (4.10)
\]

\[
\begin{pmatrix}
  w_1 & w_1 x_2 & w_1 x_2^2 & \cdots \\
  w_2 & w_2 x_2 & w_2 x_2^2 & \cdots \\
  \vdots & \vdots & \vdots & \ddots
\end{pmatrix} \cdot
\begin{pmatrix}
  p_0 \\
  p_1 \\
  \vdots
\end{pmatrix} =
\begin{pmatrix}
  w_1 y_1 \\
  w_2 y_2 \\
  \vdots
\end{pmatrix} \quad (4.11)
\]

with \( \odot \) symbolizing the element-wise multiplication with dimensional singleton expansion. With known weights, this can be formulated as \( A \vec{x} = \vec{y} \) for an unknown \( \vec{x} \) which can be solved. This weighted \( V' \) is still regular when \( V \) is regular and \( w_i \neq 0 \) for all \( i \) as the following holds for a general matrix \( A \) when multiplying the \( j \)-th row with \( w \)

\[
w \det A = \det
\begin{pmatrix}
  \vdots \\
  wa_{j,1} & wa_{j,2} & \cdots \\
  \vdots
\end{pmatrix} \quad (4.12)
\]
as Laplace expanding in the $j$-th row yields with the minor matrixes $M_{j,i}$, where the $j$-th row and $i$-th column has been removed from $A$

\[
\det \begin{pmatrix}
:\ 
wa_{j,1} & wa_{j,2} & \ldots \\
\vdots
\end{pmatrix} = \sum_i (-1)^{(i+j)} wa_{j,i}M_{j,i}
\]

\[
= w \sum_i (-1)^{(i+j)} a_{j,i}M_{j,i}
\]

\[
= w \det A
\]

\[
\Rightarrow \det V' = \det V \prod_i w_i.
\]

Therefore, as long as $w_i \neq 0$ for all $i$ and $V$ is invertible, $V'$ is invertible.

**Modified Vandermonde matrices for vector → scalar functions**

Abstracting this problem to vector to scalar polynomial functions $f(\vec{x}) = p_0 + p_1x_1 + p_2x_2 + p_3x_1x_2 + p_4x_1^2 + \cdots = y$ it is possible to incorporate the additional linear combinations by extending the Vandermonde matrix $V$. Defining a power-factor $\vec{\alpha}$ with $\alpha_i$ being the sum of powers of the $x_i$ factor contributing to $p_i$, this modified Vandermonde matrix (without weights) is defined as

\[
V_{ij} = \prod_j x_{i,j}^{\alpha_j}
\]

\[
V = \begin{pmatrix}
1 & x_{1,1} & x_{1,1}x_{1,2} & x_{1,1}^2 & \ldots \\
1 & x_{2,1} & x_{2,1}x_{2,2} & x_{2,1}^2 & \\
\vdots & & & & \\
\end{pmatrix}
\]

with $x_{i,j}$ being the $i$-th supporting point in the $j$-th dimension.

If this modified Vandermonde matrix of size $m \times n, m \geq n$ is not under-defined is not easy to determine. $V$ is not under-defined if its rank is $n$. In the typical case of $m \gg n$ and grid like supporting point patterns this is normally easily satisfied.

With this framework it is now possible to solve the equations required for the particle tracking algorithm.
Fast approximate fitting of a polynomial to the image

Assuming the polynomial $P$ and the image $I$ are defined as described in 3.3.3. As the weight of the pixels in the image decreases with $w$ it is valid to use a cutoff value below which the weights are set to 0. This limit for a particle with the radius $r$ to a radius of $r_L = kr$ in the $x$- and $y$-direction. $k$ is a chosen model parameter. This creates a square cutout of the image to which the polynomial has to be fitted to. The fit of a function $f_p$ with the fitting parameters $p$ to a dataset in a least squared error sense can be achieved by using each data point $(x, y)$ as a supporting point. The equation of whose the error has to be minimized is $f_p(x) = y$. With multiple data points this gives rise to a set of equations, in case of a polynomial a set of linear equations, which has to be solved.

Each of the $n$ points in the image used for the fitting is described by

$$x \in [x_0 - r_L, x_0 + r_L], y \in [y_0 - r_L, y_0 + r_L], x, y \in \mathbb{N}.$$  \hfill (4.19)

An index $i, 1 \leq i \leq N$ can be assigned to each point, which will be used when referring to an individual point in the image.

Each point has a coordinate $x_i, y_i$, an associated weight $w_i = w(x_i - x_0, y_i - y_0, r)$ and an image intensity $I_i = I(x_i - x_0, y_i - y_0)$. The polynomial can be defined at each point as $P_i = P(x_i - x_0, y_i - y_0)$. $P$ has the form

$$P = p_{00} + p_{10} x + p_{01} y + p_{20} x^2 + p_{11} xy + p_{02} y^2 + p_{30} x^3 + p_{21} x^2 y + p_{12} xy^2 + p_{03} y^3 + p_{40} x^4 + p_{31} x^3 y + p_{22} x^2 y^2 + p_{13} xy^3 + p_{04} y^4.$$  \hfill (4.20)

For these points, a weighted modified Vandermonde matrix $V$ can be constructed, where every column matches one of the coefficients of $P$

$$V = \begin{pmatrix} w_1 & w_1 x_1 & w_1 y_1 & w_1 x_1^2 & w_1 x_1 y_1 & \cdots & w_1 y_1^4 \\ w_2 & w_2 x_2 & w_2 y_2 & w_2 x_2^2 & w_2 x_2 y_2 & \cdots & w_2 y_2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & w_n x_n & w_n y_n & w_n x_n^2 & w_n x_n y_n & \cdots & w_n y_n^4 \end{pmatrix}. \hfill (4.21)$$
The polynomial coefficients \( p \) are then obtained by solving the equation

\[
W_i I_i = pV 
\]

\[
p = V \backslash (WI) 
\]

with \( A \backslash B \) defined as solving \( Ax = B \) for \( x \). \( w \) and \( I \) are the column matrices of all \( w_i \) and \( I_i \).

Solving this typically over-defined system of equations is possible using a QR-factorization \( V = QR \) with \( Q \) being a square and \( R \) being a upper triangular matrix [BS01, p. 917]

\[
Ax = b 
\]

\[
A = QR 
\]

\[
z = Q^T b 
\]

\[
Rx = z. 
\]

The last equation is trivially solvable since \( R \) is a diagonal matrix and can be solved by back-substitution.

An alternative method to use is using the inverse \( A^{-1} \) of \( A \) yielding \( x = A^{-1} b \). This works for rectangular problems, but not for over-defined systems. A least square solution is given using the Moore-Penrose pseudoinverse \( A^+ \) [Eld82] by using the singular value decomposition (SVD) \( A = U \Sigma V^* \) giving \( A^+ = V \Sigma^+ U^* \) and computing the pseudo inverse \( \Sigma^+ \) from \( \Sigma \) as a rectangular diagonal matrix by

\[
\Sigma^+_{i,j} = \begin{cases} 
0, & \text{if } \Sigma_{i,j} = 0 \\
\Sigma^{-1}_{i,j}, & \text{otherwise.}
\end{cases} 
\]

To get a realistic estimate of the computing power required, a typical use case is assumed consisting of a particle with \( r = 10 \) using \( k = 1.6 \) and an iteration count of 20 iterations with 8 particles tracked simultaneously with a frame rate of 10 kHz. The value of \( k \) is taken from the program written by [Rog+07], \( r \) is taken from some measurements as typical values and also yields a power of two sized image cutout, easing implementation. The iteration count is within the range that is observed when fitting high-speed camera videos. The particle count is chosen to match the maximum the
4.3. Tracking algorithm optimizations

used sensor can track. These parameter choices imply a rectangle of $32 \times 32$ pixels used by the algorithm to fit the polynomial to the image for every particle.

The need to fit $32 \times 32 = 1024$ pixels to 15 parameters leads to a set of equations of the size $m \times n = 1024 \times 15$. The computational cost for this step depends on the values of $m$ and $n$. When utilizing the QR decomposition, the computational cost scales with $2mn^2 + 2n^3$ [DO12, p. 7]. $m$ denotes the bigger dimension of the matrix, $n$ the smaller dimension. Inserting the values of the assumed sample case yields a cost of 468k operations.

This has to be done for $10^4$ frames per second, 8 particles and 20 iterations per particle leading to

$$468 \text{ k operations} \cdot 10^4 \text{ kHz} \cdot 8 \cdot 20 \simeq 750 \times 10^9 \text{ FLOPS} = 750 \text{ GFLOPS}. \quad (4.29)$$

This is close to 1 TFLOPS which is theoretically achievable using modern GPUs, but actual implementations of the QR-factorization results in much lower values [SA05] and need about 5 ms for a matrix with 16 rows, being too slow by multiple magnitudes for the required speed of 5µs. Modern CPUs also are not fast enough, reaching computation times in the range of 1.2 ms [SA05]. Thus they are also too slow by two orders of magnitude. Therefore, this calculation needs to be simplified to be computable with current technology in real time.

To accelerate the computation, equation 4.23 needs to be solved faster. By writing the equation as

$$p = V \backslash (W \cdot I) = (V \backslash W) \cdot I = M \cdot I \quad (4.30)$$

the computation time can be reduced if $M$ is precomputed. As $I$ is a column vector and $M$ has the size of $m \times 15$ this reduces the computation down to $2 \cdot m \cdot 15$ operations. With the assumed sample case this equals 31 k operations per particle and is roughly fifteen times faster than the QR method.

$M$ cannot be precomputed in the general case as $M$ depends on the Vandermonde matrix $V$ which depends on $x_0, y_0, r$ and $w(x_0, y_0, r)$. Notably $M$ does not depend on $I$ at all, making it only dependent on position and radius.
Chapter 4. Sensor

The novel idea is to transform the image by \( x_0, y_0 \) and \( r \) to a standard problem with a precomputed matrix \( M \) and to reverse transform the coefficients \( p \). A sketch of this transformation is shown in figure 4.4. Basically, the coordinates \( x, y \) are transformed into a new coordinate system \( x', y' \) using

\[
x' = (x - x_0) \cdot \frac{r_R}{r}, \tag{4.31}
y' = (y - y_0) \cdot \frac{r_R}{r}. \tag{4.32}
\]

This is equivalent to shifting the assumed image center \( x_0, y_0 \) to zero and to scale it to a standard radius \( r_R \). This new image is then fitted and the parameters \( p' \) are computed. After computation they have to be transformed back into the original coordinate system via

\[
p'_{ij} = \frac{r^{i+j}p_{ij}}{r_R}. \tag{4.33}
\]

This transformation can be either done by transforming the complete image into the new coordinate system and letting the fitting algorithm run for that image or it can dynamically generate the image gray values by sampling the image on demand when accessing the image brightness with the new coordinates. The on-demand method has the advantage that it does not need any additional memory and it introduces no additional computing overhead. The image transformation method has the advantage to be able to use more sophisticated image scaling techniques. As the computing demand for the sophisticated scaling is beyond the power of the hardware used (see section 4.1), this approach was not realized and the direct sampling was used instead.

The standard case is set to \( r_R = 10, k = 1.6 \) as chosen above as a typical case. \( M \) is a weight matrix for each coefficient and each point in the image. These weights are shown as false color image in figure 4.5. All weight matrix show either point or mirror symmetry along the \( x \)- and \( y \)-axes. These symmetries reduce the storage requirement for the coefficients by a factor of four. Also by grouping sampling points with their symmetric partners, the multiplication of the sampled value with the coefficient can be factored out, allowing a reduction of the multiplications to a quarter of the coefficients.
4.3. Tracking algorithm optimizations

Figure 4.4: Sketch of the transformation to the standard case. By using an affine projection a particle from an recorded image shown on the left with arbitrary size and position is projected into a standard coordinate system shown on the right. The overlaid dots are the sampling points used by the original and the novel algorithm. The fixed sample density on the left side is given by the pixel pitch of the camera. On the right side the number of sampling points is always constant, thus the apparent sampling density is scaled with the particle radius.

The factors $p'$ computed after this transformation need to be re-scaled to match the dimensions of the original radius of the image using

$$p_{ij} = \frac{p'_{ij} r \cdot R}{r^2 + J}.$$  \hspace{1cm} (4.34)

In the original algorithm the absolute position is relative and therefore does not need to be considered in the corrections.

This novel approach is not perfect. Most importantly, it is not an exact equivalent substitution of the original formula and therefore does not yield the exact same result. Mathematically speaking, the difference lies in the affine transform and the altered sampling points. This is equal to selecting different support points from the original data set. If the sampling grid would be infinitely dense and the image completely described by a surface, then the results would be identical. As we are bound by noisy data, finite sampling point density and finite resolution, this is only an approximation. The precision of this approximation and how it influences the tracking performance is discussed in detail in section 4.4. As shown in figure 4.4 the image data is transformed onto a new coordinate...
Figure 4.5: The weights matrix in false color.

The original size of the matrix $M$ is $15 \times 1024$ (15 parameters, $32 \times 32 = 1024$ sample points). Since each row of $M$ represents a polynomial factor $p$ sampled in a grid, they are shown as an image. The numbers on the top and the side are the powers of $x$ and $y$ associated with the factors $p$. Visible is the mirror symmetry in the even rows per direction and the point symmetry for the odd rows.
4.3. Tracking algorithm optimizations

Table 4.2: Different sampling/interpolation methods and their smoothness, flatness, and computational complexity. SM is the order of the spatial derivative when the sampling function is no longer smooth. LU/sample is the number of data points needed to fetch from the original image to sample one point. OP/sample is the number of arithmetic operations to compute one point. OV shows if the algorithm can produce data outside the input range (overshoot). Costs are approximated after fetching the samples and by counting arithmetic operations.

<table>
<thead>
<tr>
<th>Name</th>
<th>SM</th>
<th>LU/sample</th>
<th>OP/sample</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest neighbor</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>Bilinear</td>
<td>1</td>
<td>4</td>
<td>9 (3 for each dimension)</td>
<td>no</td>
</tr>
<tr>
<td>Bicubic</td>
<td>2</td>
<td>16</td>
<td>72 (18 for each dimension)</td>
<td>yes</td>
</tr>
</tbody>
</table>

system and therefore needs to approximate the data between the mathematically infinite small original data points. For this step, texture sampling can be employed. There are different sampling methods with distinct advantages and disadvantages. In general, it is a trade-off between computational complexity and accuracy.

The sampling algorithm that is used needs to fulfill some basic requirements. Most importantly, it needs to be smooth between two points of different values. Violating this requirement leads to aliasing problems as shown in figure 4.6. The computational cost of the interpolation should be kept in mind when selecting a method as the sampled point gets only multiplied with 15 values. Thus the sampling itself can take a significant part of processing power compared to the other parts of the algorithm and might offset the speed gains by the transformation if it is too complex. Moreover, the sampling algorithm should not have a variable runtime, as this would make it impossible to give performance guarantees.

A list of commonly used sampling schemes is shown in table 4.2. The nearest neighbor does not suffice the minimum requirement for smoothness and is therefore excluded. The bilinear and bicubic methods yield good results. The bicubic method has theoretically better properties as it is smooth up to the second derivative, but the arithmetic calculation cost of approximately 72 operations is much higher compared to the bilinear method and would dominate the computation requirements for the whole algorithm while not giving significantly better results. Therefore, bilinear interpolation was chosen, because it has the lowest computational cost while satisfying the requirements.
Figure 4.6: Comparison of convergence with different interpolation methods.

The image a) is fitted with the algorithm with slightly varying starting values in $x = 32 + [-0.5, 0.5]$, $y = 16$, $r = 10$. The three methods nearest-neighbor, bilinear, and bicubic interpolation are used. As depicted in the plots in b) for the final radius $r$, c) for the final $x$ and d) for the final $y$ position, nearest neighbor interpolation introduces artifacts with up to $\pm0.5$ px divergence, whereas the bilinear and bicubic interpolations converge to the exact same values for $x$ and $y$ and nearly the same value for $r$. 
4.3. Tracking algorithm optimizations

When using the bilinear interpolation, the arithmetic cost to compute the coefficients \( p \) is therefore

\[
1024 \cdot (9 + 2 \cdot 15) = 39936 \text{ operations,} \tag{4.35}
\]

which is independent of the size of the particle and thus allows for deterministic run times and timings. Compared to the cost of the non-optimized algorithm using the QR factorization, the arithmetic cost reduces from 468 k operations to 40 k operations, making the novel approach more than eleven times faster to calculate than the original one.

The generation of \( V \) and \( W \) is not attributed for in the computation cost of the original algorithm, making it even slower in reality than calculated, increasing the speed advantage of the novel method further.

4.3.2 Optimized radius determination

The algorithms to compute the position estimates given in equation 3.30 and 3.31 are relatively straightforward and cannot be optimized much. However, the calculation of the radius \( r_n \) is more complex and can be optimized. One of the computational most costly parts are the computation of the angle \( \theta \) and the trigonometric operations in 3.32, 3.33, 3.34 and 3.35. As the angle \( \theta \) is only needed for the trigonometric operations, it is not necessary to compute it explicitly for every iteration. The angle \( \theta \), eccentricity \( e \) and skewness \( s \) can be completely computed offline or after the algorithm has converged, reducing the computational demands.

To optimize computation of the radius \( r \), plug the equation 3.38 for \( \theta \) into a sine and a cosine. The aim is then to simplify the calculation of

\[
\sin \theta = \sin \left( \frac{1}{2} \cot^{-1} \frac{p_{20} - p_{02}}{p_{11}} \right), \tag{4.36}
\]

\[
\cos \theta = \cos \left( \frac{1}{2} \cot^{-1} \frac{p_{20} - p_{02}}{p_{11}} \right). \tag{4.37}
\]
By using the identities

\[
\sin^2 \alpha + \cos^2 \alpha = 1, \quad (4.38)
\]

\[
\cos^2 \left( \frac{1}{2} \cot^{-1} a \right) = \frac{1 + \cos \cot^{-1} a}{2}, \quad (4.39) \quad [\text{AS13, p. 72}]
\]

\[
\tan \alpha = (\cot \alpha)^{-1}, \quad (4.40)
\]

\[
\Rightarrow \tan^{-1} x = \cot^{-1} \frac{1}{x}, \quad (4.41)
\]

\[
\cos \tan^{-1} x = \frac{1}{\sqrt{1 + x^2}}, \quad (4.42)
\]

and defining

\[
a = p_{20} - p_{02}, \quad (4.43)
\]

\[
b = p_{11}, \quad (4.44)
\]

cos \theta can be computed as follows

\[
\cos \theta = \cos \left( \frac{1}{2} \cot^{-1} \frac{a}{b} \right) \quad (4.45)
\]

\[
\cos^2 \theta = \cos^2 \left( \frac{1}{2} \cot^{-1} \frac{a}{b} \right) \quad (4.46)
\]

\[
= \frac{1 + \cos \cot^{-1} \frac{a}{b}}{2} \quad (4.47)
\]

\[
\Rightarrow \cos^{-1} \frac{a}{b} = \cos \tan^{-1} \frac{b}{a} \quad (4.48)
\]

\[
= \frac{1}{\sqrt{1 + \frac{b^2}{a^2}}}
\]

\[
= \sqrt{\frac{a^2}{a^2 + b^2}} \quad (4.50)
\]

\[
\Rightarrow \cos^2 \theta = \frac{1 + \sqrt{\frac{a^2}{a^2 + b^2}}}{2} \quad (4.51)
\]

\[
\Rightarrow \cos \theta = \sqrt{\frac{1 + \sqrt{\frac{a^2}{a^2 + b^2}}}{2}}. \quad (4.52)
\]
4.3. Tracking algorithm optimizations

\( \sin \theta \) can be simplified in the same manner:

\[
\sin \theta = \sin \left( \frac{1}{2} \cot^{-1} \frac{a}{b} \right)
\]

\[
\sin^2 \theta = \sin^2 \left( \frac{1}{2} \cot^{-1} \frac{a}{b} \right)
\]

\[
= 1 - \cos^2 \left( \frac{1}{2} \cot^{-1} \frac{a}{b} \right)
\]

\[
= 1 - \frac{1 + \sqrt{a^2 + b^2}}{2}
\]

\[
\Rightarrow \sin \theta = \sqrt{1 - \frac{1 + \sqrt{a^2 + b^2}}{2}}
\]

\[
= \sqrt{1 - \frac{\sqrt{a^2 + b^2}}{2}}.
\]

Further optimizations can be done using only the combined products of \( \sin \theta \) and \( \cos \theta \), since always two or four terms are multiplied together in the equations to compute the factors \( A, B, C, D \) as described in section 3.3.3, saving another square root:

\[
\cos^2 \theta = \frac{1 + \sqrt{a^2 + b^2}}{2},
\]

\[
\sin^2 \theta = \frac{1 - \sqrt{a^2 + b^2}}{2},
\]

\[
\cos \theta \sin \theta = \sqrt{1 - \frac{\sqrt{a^2 + b^2}}{2}} \cdot \frac{1 + \sqrt{a^2 + b^2}}{2}
\]

\[
= \frac{1}{2} \sqrt{1 - \frac{a^2}{a^2 + b^2}}.
\]

The optimizations might seem small, but they form a critical dependency path within the algorithm and therefore need to be as fast as possible. Additionally, by eliminating the need for the trigonometric functions, a hardware implementation requires much less area and is therefore more compact which saves power, resources and allows higher speeds.
Chapter 4. Sensor

4.3.3 Optimized computation of $\sqrt[4]{x}$

The last step of the radius computation contains the fourth root. This root takes around 20 cycles to compute and therefore dominates the latency time of the radius computation with roughly 15% of the computation time. To speed up this computation, a floating point approximative solution was developed.

An IEEE-754 floating point number consists of a sign bit $s$ (1 bit), an exponent $e$ (8 bit), and a mantissa $m$ (23 bit) [IEEE, p. 19, p. 23]. The true value $v$ is

$$v = (-1)^s 2^{e-127} (1 + m \cdot 2^{-23}).$$  \hfill (4.63)

The values 0 and infinity and various error values (e.g. NaN) are encoded with $e = 0$ and distinct values in $m$. To compute the fourth root, the cases for $v \leq 0$ and the error conditions can be handled separately, the normal case simplifies to

$$\sqrt[4]{v} = 2^{(e-127)/4} \sqrt{1 + m \cdot 2^{-23}}.$$ \hfill (4.64)

$$= 2^{\left\lfloor \frac{e-127}{4} \right\rfloor} \left( (e-127) \mod 4 + 1 \right) \sqrt{1 + m \cdot 2^{-23}}.$$ \hfill (4.65)

This shows clearly, that the exponent $e$ is divided by four and the remainder is multiplied to the mantissa part, giving it a range in the root of $[1, 16)$. This part has to be approximated. A standard way is to use multiple iterations of approximations, until enough bits of precision have been accumulated. This implies a high latency albeit low complexity hardware. To achieve a faster computation speed at even lower complexity, a lookup table (LUT) is computed containing 2048 approximate roots in the range of $[1, 16)$, which are then linear interpolated. The error of this function is in the worst case 1 unit in the last place (ULP), the mean relative error is $2.6 \times 10^{-8}$, the maximum relative error is $1.2 \times 10^{-7}$. These deviations are good enough for the computation of the radius. The latency of this function is 4 clock cycles at 100 MHz, being roughly five times faster than the trivial approach with two stacked square roots. The resource footprint of configurable logic block (CLB) resources is much smaller with the custom root function. In contrast, the new function consumes one DSP and two BRAM blocks.
4.3. Tracking algorithm optimizations

4.3.4 Adaptive active sensor region adjustment

One of the main hurdles when building high-speed video systems is the amount of data generated by the frame acquisition. As the resolution and the active sensor area determine the field of view and therefore the number of particles which can be tracked at the same time, it also increases the bandwidth that is needed to process the images. However, in microrheology experiments typically more than 90% of the active sensor area is never used for the particle tracking and can be considered therefore as irrelevant data (see figure 4.7). Some sensors using the CMOS technology provide the possibility to only read out arbitrarily defined regions of the sensor’s pixel array. This allows for a significant reduction of the number of active pixels and therefore decreases the needed bandwidth.

For highly confined particles, it is sufficient to define small ROIs around the particles before the experiment as the tracked particles never move far in the field of view. This might not be the case for more dynamic or very long experiments, in which for example

---

Figure 4.7: Visualization of the dead region while tracking particles.
The area outside of the particles and their direct surrounding needs to be read out with a typical high-speed camera setup, as there is no a priori knowledge where a particle is. This forces the storage of much image data that is never used. The developed sensor does not need to read out the red shaded region as it has a priori knowledge of the approximate particle position. Image a) is adapted from [Pfe+12] and shows the reduction of the active sensor area. Image b) is a sample measurement of a NIH/3T3 fibroblast where the unused regions are shaded red. The marked circles are the tracked particles as of the program of [Rog+07].
Chapter 4. Sensor

A cell might start to migrate and drag the incorporated particles with it. To overcome this problem and to make the particle position available in real time with very short latency, it is possible to move the ROI with the particle. This approach of course needs a very tight coupling of sensor and tracking logic, as a direct feedback path between these two is needed. Additionally, invalid sensor configurations or glitches by not reconfiguring the sensor fast enough should be avoided. Schematics showing the latency and reconfiguration problems as well as the strict timing margins are shown in figure 4.8.

To hit the right reconfiguration window it has been proven by trial and error that it is sufficient to wait until the first row of image data is received from the sensor and then to kick off the reconfiguration by scheduling the reconfiguration changes as tightly as possible to the configuration interface of the sensor. Using this tightly coupled approach no frames are lost when reconfiguring the sensor. The actual implementation does this reconfiguration every frame for all particles, as it is easier to simply reconfigure the sensor instead of calculating the differences to the previous frame and reconfigure only with those differences.

The principles of adjusting the sensor readout region to the particles being tracked has been submitted as a patent by the author [Pfe+12].

A disadvantage of the reduced sensor readout region is a limited monitoring of the experiment by the user. This can be compensated if the needed frame rate and particle count allow for at least one additional ROI. This ROI can be moved over the whole sensor region without tracking any particle for this ROI and thus soft-refresh the image presented to the experimenter. This is much slower and needs 100 ms until the complete sensor area has been refreshed once. However, it is fast enough and allows to monitor the experiment in more depth.

4.4 Validation

Since the newly developed computation method for the polynomial fitting is only an approximation of the original algorithm, it has to be validated. Therefore, a bit-exact model of the tracking algorithm used in the sensor was written in Matlab (Matlab R2019b, Mathworks Inc.). This serves two purposes, first it allows the verification of the correct function of the hardware; and second it allows simulating the algorithm with computer
Figure 4.8: Tracking process and timing table for a single particle with adaptive ROI configuration.

For an adaptive reconfiguration of the ROIs there are two challenges. Firstly, the position of the particle is only known several frames later (in this sketch 3 frames later), and secondly the reconfiguration window might be very small and therefore needs to be hit precisely. Missing the window would lead to readout corruption or wrong pixels being read out without any indication, leading to a bogus tracking. Image adapted from [Pfe+12]
generated sample images. To validate the developed approach, synthetic and real images of experiments are fed into the model. The results are compared with the exact input values for the synthetic images and with the original polyparticletracker software as provided by [Rog+07].

4.4.1 Synthetic images

To evaluate the performance of the novel algorithm, first the converging behavior is analyzed. Then a simulated random walk with boundary effects is tested. Finally, the shot pixel noise margin is tested to evaluate the robustness.

Convergence of particle detection

To get a deeper understand of the algorithm’s behavior, a single Gaussian peak is tracked by the algorithm. To asses the performance, the converging behavior of multiple rounds of fitting and position/radius computation is observed.

Figure 4.9 shows the Matlab model converging on a simple Gaussian peak. The differences between the original peak and width parameters put into the model and the results can be attributed to the precision of the interpolation and the sampling of edge pixels. As the semi-logarithmic plot shows, the position estimation and radius estimation converges after roughly 13 rounds. The small positional error of approx. 0.017 px in \( y \) can be attributed to border effects, as the sampled regions extends beyond the image boundaries (the sample region is \([x_0 - 1.6r, x_0 + 1.6r] \times [y_0 - 1.6r, y_0 + 1.6r]\)). The used boundary algorithm is coordinate clamping. The coordinates of the sample to compute are clamped to the image size before given to the memory gather and interpolation routine. This is equal to replicating the image border values to infinity.

Simulated random walk

To get a deeper understanding of the boundary effects and the overall tracking performance, a random walk of a particle is simulated. The particle size is deliberately chosen to be fairly large to force the algorithm to observe boundary effects on the window borders.
Figure 4.9: Parameter evolution while tracking a Gaussian peak. Image a) shows the synthetically generated Gaussian peak with a center position of $x = 30, y = 15$ pixel and a radius of $r = \sigma = 10$ pixel. The tracking algorithm has an initial guess $x = 35, y = 16$ pixel and a radius of $r = 10$ pixel. Plot b) shows 20 rounds of the tracking algorithm. The line for $x$ is not continually drawn as the algorithm computes 0 change for some rounds. The final result of the algorithm is $x = 30, y = 15.0165, r = 10.1524$.

Figure 4.10 shows the positional differences between a generated random walk of a Gaussian peak and the tracking algorithm. The errors are small again. The random walk is generated by integrating Gaussian weighted random numbers with a width of $\sigma = 0.5$ pixels in both dimensions as random walk steps for each successive image, starting from the center of the image at $x = 32, y = 16$ pixel. This is also chosen as the initial position for the algorithm. The size of the peak is randomized by sampling from a Gaussian peak with the center of 10 and a width of 2 for each image. As shown in figure 4.10, the algorithm generates an almost perfect results as long as the sampling area does not move outside the picture boundaries. If the area exceeds the boundaries, a systematic drift is observed, which is expected due to the finite image size and coordinate clamping border interpolation. As the used sampling window is twice as big in the $x$-direction as in the $y$-direction, the tracking results in the $y$-direction are much more likely to be influenced by the borders of the image. This can be seen as a worse tracking result in the $y$-direction.

**Tracking with increased noise level**

An important metric for an algorithm is the noise rejection. Therefore, the tracking was analyzed with increasingly noisy images.
Figure 4.10: Deviation from the ideal result with a simulated random walk. The walk is shown in image a). It starts at $x = 32, y = 16$. In image b) the deviations from the perfect result are shown. It is nearly zero at the beginning, but rises when the particle moves to the left side. As the algorithm samples a region $1.6r$ to both sides, moving with an average size of 10 px near the border will introduce sampling artifacts, thus worsening the fitting result.

In c) the deviation of the computed radius from the true radius is shown. The algorithm seems to underestimate the radius of a Gaussian peak most of the time. This seems to get worse when approaching the border.

The graph d) shows the distribution of the positional errors, showing a better conformance for the $x$- than the $y$-direction. This is expected as the $x$-direction is much less influenced by the border effects compared to the $y$-direction.
Figure 4.11 shows the results of the position and radius tracking of a Gaussian peak with different degrees of noise applied. As shown in the boxplots, the noise level increases the position determination only slightly as long as the noise is moderately high. When the noise level exceeds a higher level, tracking gets very imprecise and even might not converge any more. The minimal signal-to-noise ratio needed to achieve reliable tracking is in the range of $\frac{128}{20} \simeq 8.1 \text{ dB}$ with a Gaussian noise background.

In a real measurement, an image smoothing step is typically done before tracking the particles, giving a better pixel noise rejection by averaging over multiple pixels. This increases the signal-to-noise ratio, but makes the tracking of small particles more challenging.
To a Gaussian peak \( I(x, y) = 64 + 128 \exp\left( -(x - x_0)^2(y - y_0)^2/(2\sigma)^2 \right) \) with \( x_0 = 32, y_0 = 16, \sigma = 10 \) random Gaussian pixel noise is added. The noise-level (NL) corresponds to the width of the noise distribution. For every NL 128 images are generated and the algorithm fits a peak on these images (start position \( x = 32, y = 16, r = 10 \)). In image a) sample images for the corresponding NL are shown. Image b), c) and d) show the absolute deviation distributions for the corresponding algorithm outputs. The boxes mark the [25,75] intervals and the red line is the median. Up to an NL of 20 the results are still usable. With an NL higher than 20 the fitting quality deteriorates very fast.

Figure 4.11: Influence of noise on algorithm output.
4.4 Validation

4.4.2 Real measurements

To assess the performance of the new algorithm in comparison to the classical algorithm polyparticletracker, a real measurement is analyzed with the Matlab model and the polyparticletracker.

Figure 4.12 shows the simulated real-time tracking of a real recorded video. A NIH/3T3 fibroblast with incorporated 1 μm polystyrene beads can be seen. The data is captured using an EcoSens Cube7 (Mikrotron GmbH) and using the setup described in [Nec18]. For comparison the tracking result of the polyparticletracker from [Rog+07] is shown. The images are smoothed with a Gaussian convolution kernel with a width of $\sigma = 1$, which improves the tracking performance. Additionally, the polyparticletracker would take very long to find the particles in the starting particle search when using the non-smoothed images.

The polyparticletracker uses an algorithm to find particles without further input (see chapter 3.3.3). This is used to select initial locations and radii of the tracking particles for both the novel algorithm and the polyparticletracker. For the selected dataset, 29 particles are found and tracked. The novel algorithm is able to track 28 particles and loses 1 particle about halfway through the video. This is not a deficit of the new method as the polyparticletracker loses the particle at frame 9'500 and the new algorithm loses it at frame 37'035, giving a small advantage to the new method. This particle is excluded for the further analysis.

Figure 4.13 shows the differences between both tracking algorithms in more detail. $x$ and $y$ deviations are within a 0.01 pixel range relative deviation for a single particle. This is a good result and shows that the positional error of the new method is in the same range as in the original method. The deviation in $r$ is slightly shifted to the negative, suggesting that the new algorithm tends to underestimate the size compared to the original algorithm. As shown in figure 4.13, the distributions of the deviation vary per particle, suggesting particles which both algorithms track with similar results and other particles where the deviations are more pronounced.

The median radius of all tracked particles for this video is 3 px. The smallest particle has a mean apparent size of 2.3 px and the biggest particle is 4.8 px in apparent size. These sizes are relatively small and introduce a heavy influence of the grid by the selection
Chapter 4. Sensor

Figure 4.12: Tracking of beads in and on a NIH/3T3 fibroblast.
Using the setup described in [Nec18], the cell shown in image a) is a NIH/3T3 fibroblast. The tracks are drawn ten times larger than observed for better visibility. The mean deviation for the tracking components $x, y, r$ with respect to the polyparticletracker from [Rog+07] are shown in b). $x$ and $y$ show a good conformation with less than 0.01 px deviation and the error in $r$ has a larger bandwidth of 0.04 px and shows a small underestimation of $r$.

of the sampling points for the original algorithm. As the average sampled grid has a size of only $10 \times 10$ pixels, including a row in the ROI for fitting influences the result of the tracking algorithm – especially the radius computation. This is partially corrected by the new algorithm by not using pixel grid boundaries as cutoff points, but instead using interpolation to ease in the bordering pixels. This reduces aliasing effects through the underlying grid structure.

A systematic underestimation of the radius by the new algorithm can also be shown. This seems to be introduced by the linear sampling. Assuming a bright particle without loss of generality. As the particle has a negative second derivative to be an extremum, the interpolation always underestimates the brightness around the center, inducing a systematic darkening of the image around the particle center. This effect lowers the predicted radius, which explains the underestimation. This could be mitigated by using cubic or spline interpolation, but then a real-time tracking would no longer be possible with the given hardware. The underestimation is not a problem, as the error is small compared to the absolute value of the radius.

In summary, the tracking performance of the new algorithm is as good as the original polyparticletracker for real measurement data and validates the performance and versatility to be an usable approach for tracking.
Figure 4.13: Detailed plots of the differences between the polyparticletracker and the novel algorithm.

The particles are tracked as described in figure 4.12. Some particles can be tracked better than others and keep their deviation from the reference tracker very small. On the left side for each histogram a Gaussian distribution (form $\frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x-\mu)^2}{2\sigma^2} \right)$) has been fitted. The deviations for the $x$ and $y$ position are in the range of less than 0.01 px while the deviation of the radius $r$ is in the range of 0.04 px and slightly shifted to smaller values.
4.4.3 Discussion of the simulated tracking performance

The novel algorithm is tested with synthetic pictures generated using Gaussian peaks. As long as the particle does not approach the border of the readout region, the deviation from the analytically correct result is small and the resulting position estimate is very precise — even for sub-pixel locations. If the particle is near the border, border effects deteriorate the signal, introducing errors in the sub-pixel range. Even if noise is added, the novel algorithm produces good results up to a signal-to-noise ratio of around 8.1 dB. Higher noise seems to impact the quality of the tracking disproportionately large.

Using the novel approach on real captured movies shows a good confirmation with the old algorithm width differences in the range of 0.01 px. This is confirming the capability of the novel algorithm to be used as a sub-pixel precise tracker. The radius estimation seems to be systematically underestimated by around 0.04 px. This is no problem as the deviation distribution is small between different particles and shows a narrow band of less than 0.04 px for individual particles, which allows for an systematic size compensation if needed.

The Matlab simulation is used as the gold standard which the VHDL models are tested against. The tests are conducted by the means of testbenches and by debugging the in- and outputs of the sensor and shown to match up to the precision of the floating point numbers in the testbench and the implemented hardware. Therefore, the sensor system FPGA is validated to work correctly with high precision.

In conclusion, the novel algorithm achieves comparable tracking precision to the method developed by [Rog+07] while being able to run at 10 kHz on an FPGA by reducing the complexity of the used equations substantially. Therefore, this new method can be used to implement a real-time high-speed tracking of particles.
5 Experiments

To validate the sensors performance, first NIH/3T3 cells were cultivated and incubated with 1 µm polystyrene beads. These were afterwards tracked for up to 1.5 hours. To show the effect of blebbistatin, beads incorporated by a single cell were tracked for 15 minutes, then blebbistatin was added and the same beads were tracked again for additional 45 minutes.

In the beginning, a short overview of the used setup and hardware and the sample preparation is given, then the results of the experiments are shown and discussed.

5.1 Experimental setup

A sketch of the setup is shown in figure 5.1. The arrow thickness in the sketch corresponds to the data rate. The thin arrows represent rates below 10 MB/s, the medium thick arrows are only used for image data and have a data rate of 288 MB/s. The single very thick arrow symbolizes the reading of the image buffer, which happens at 16 GB/s.

The data flow is as follows: After the sensor has triggered the pulsed LED to illuminate the pixel array, it is read out and directly sorted to the correct ROI image buffers by the ROI control. One frame later the smoothing block smooths those raw images and transfers them into the image buffer. Another frame later, the interpolation reads those images 20 times to perform the polynomial fitting and iteratively refine the position and radius estimations for this image. The processor finally transmits this information to the main memory for storage. Again one frame later the image sensor is instructed with the new particle positions to adjust the readout region, which then gets into effect one frame later. In summary, it takes 4 frames until the new position is actively read out, thus having a minimum latency of 400 µs when tracking particles.
5.1.1 Microscope

The used microscope is a Axiovert microscope (Zeiss). The sensor was mounted using a C-mount adapter provided by Zeiss at the right camera port. The illumination is a custom build green LED with a pulsed custom-developed LED driver (Sensific GmbH). Using a pulsed LED reduces motion blur and limits the light amount the cell receives to the minimum necessary. This is done to avoid photo-toxicity. The used objective is a 40x apochromatic (Zeiss EC Plan-neofluar) and phase contrast was used.

The microscope itself is positioned on a damped marble stone plate to reduce oscillations from the building.

5.1.2 Sample preparation

Glass bottom dishes (Ibidi, 81218-200) were coated with fibronectin (Sigma-Aldrich, F2006-2MG) for 2 h. The cultured NIH/3T3 cells (ATCC, CRL1658) are washed with phosphate buffered saline (PBS) (Gibco by life technologies, 14190-144) and detached with Trypsin (Biowest, L0910-100, 0.25 %) for 3 minutes (5 % CO₂, 37 °C). The trypsi-
nation is stopped with Dulbecco's modified eagle medium (DMEM, Gibco by life technologies, 41965-039). 100'000 cells per 2 ml are seeded on the glass bottom dishes and incubated over night (5 % CO₂, 37 °C). The medium contains 10% FBS (fetal bovine serum, Bio&Sell, FBS.GP.0500) and 1% antibiotic and antimycotic solution (GE-Healthcare Hyclone, SV30079.01).

The next day the cells are washed 3 times with PBS and 4 μl beads (Microbead NIST traceable particle size standard 1 μm, Polysciences 64030-15) are added to the cells in 2000 μl medium. The cells are incubated again for 6 h (5 % CO₂, 37 °C) and then washed with PBS 3 times. Then 2000 μl Live cell imaging solution (ThermoFisher, A13291DJ) is added.

The cells which received blebbistatin treatment (Sigma-Aldrich, B0560-1MG) are treated with 50 μM blebbistatin in the 2 ml Live cell imaging solution. When adding the blebbistatin late, as in the second half of the experiment, the concentration of the blebbistatin in the later added solution is doubled to reach the same concentration after mixing.

The lid of the glass bottom dishes is kept on the sample for all measurements to avoid drying and cooling effects trough evaporation. The measurements of the second half, in which the cells get additional imaging solution (with or without blebbistatin) are done with the lid placed reversed on the dish. This is done to ease lifting off and placing the lid on the dish to minimize the movement of the dish when adding the imaging solution.

Blebbistatin is chosen as a drug as it inhibits the correct function of myosin in the cell [DAA07; Str+03], thus block an important method of the cell to do active movement [WS87]. With these experiments the speed of onset and the extend of the effect on the MSD of embedded particles and thus the cell stiffness is probed. Expected is a decrease of the MSD as the cell is limited in the active transport of particles within itself. Especially a decrease of the MSD for longer time delays is expected, as active mechanisms influence those preferably.

### 5.2 Experimental results

To compute the MSD for very long timetraces, it is no longer feasible to compute the MSD for every possible time delay and data point. As the number of points $n$ increases, the point combinations rises with $n^2$, giving a runtime complexity of $O(n^2)$. To illustrate
this, let's assume a 1.5 hour trace which contains roughly $2^{24}$ time points at 3000 Hz. These give rise to $2^{48}$ time point combinations to permute, saving it for one dimension and particle as 32-bit float. Trying only half the combinations – as the other half are the same – requires reading $2^{50}$ bytes or 1 PiB of data. Doing this for 8 particles and 2 dimensions this equals 16 PiB. Computing this costs 32 PiOP operations including checks for NaN. This would take a long time in the order of hours, even when utilizing GPUs. Therefore, the sample points on the time delays were reduced so that each sample point $s_i$ obeys the equation

$$s_i > 1.001s_{i-1}, \ i \geq 2.$$  

(5.1)

The value 1.001 is chosen and gives a relative minimum resolution of the time steps of 0.2%. This reduces the complexity of the algorithm to $O(n \log n)$ and the computation time to some minutes.

First, beads in a blebbistatin threatred cell are tracked for 1.5 hours to demonstrate the correct function of the sensor. Then two measurements are conducted where blebbistatin is added after 15 minutes into the measurement.

### 5.2.1 Very long time tracking of a cell treated with blebbistatin

3T3 cells are incubated and grown as described in section 5.1.2 and are treated with blebbistatin and then directly put under the microscope. The measurement is started approximately 10 minutes after applying blebbistatin.

The particles are then tracked for 5300 s with approximately 3010 Hz, yielding 15’951’072 data points. 8 particles are initially tracked, 1 particle is lost after 9’961’663 frames. All data is recorded using phase contrast microscopy.

An image of the cell at the start and the end of the experiment is shown in figure 5.2. The setup exhibits a slow drift with respect to the focus height. Nevertheless, the sensor has no problem tracking 7 of 8 particles. Those 7 particles are corrected by a center of mass correction according to [Kle18, p. 24][RS13]. As shown in figure 5.3, the center of mass noise of the data increases the first 15 minutes and then reaches a steady state. This might correlate to the drift out of focus, which might have stopped after approximately 1000 s.
5.2. Experimental results

Figure 5.2: Phase contrast images of the blebbistatin treated cell.

The tracked particles are marked by the sensor. Image a) is taken at the beginning of the measurement, image b) at the end of the measurement. As visible in b) image, the orange labeled particle is lost while tracking. Also b) shows the loss of focus by a drift of the microscope. The width of one rectangle is 7.68 \( \mu \text{m} \).

The apparent radius of the particles as shown in figure 5.4 increases over the approximately first 1000 s and then stays approximately constant for the rest of the experiment. This is a strong indicator that the drift of the focus height stops after approximately 1000 s. The apparent radius between particles differs by up to approx. 50 \%. This can be attributed to different heights of the particles in comparison to the focus height and slightly different surroundings with different refractive indices which leads to bigger apparent sizes. An 1 \( \mu \text{m} \) bead should normally have a radius of 0.5 \( \mu \text{m} \). The observed radius is smaller than that. This can be explained by the phase contrast microscopy, where the apparent radius of particles is the size of the focus spot and is not directly correlated to the actual size of the particles, thus the smaller values are expected.

The MSD as shown in figure 5.5 has an increase in the MSD in the lowest step intervals with a slope of 0.17 in the double logarithmic, hinting a sub-diffusive behavior. At time steps of 0.01 s or 100 Hz the MSD enters a long phase of a very shallow slope, indicating that the particles are indeed fixed or embedded in a material, which seems to show viscous behavior at time scales over 10’s of seconds and elastic behavior at shorter time scales. This is in accordance to literature [Lee+10]. The increase of the MSD over longer timescales is also described in literature [Gal+09].

The observed absolute MSD values are in the same range as written in the literature [Nec+16, p. 129] [Mar16, p. 95], proving a working and functional setup.
Figure 5.3: Center of mass movement for a very long measurement. 
a) shows the per frame movement of the center of mass of the particles.
b) shows cumulative total movement. A slow increase of the noise is visible in the first 1000 s, then the noise does not increase any more.
Figure 5.4: Apparent particle radius of 1 μm polystyrene beads in phase contrast. The beads in the cell shown in figure 5.2 are tracked for 1.5 hours. The apparent radius increases for the first 1000 s and then stabilized. The orange labeled particle is lost after 3300 s. The colors of the graphs match the colors on the image. The shown data is smoothed with a window of 1024 measurement points, corresponding to 0.34 s.
Figure 5.5: MSD of 1 μm polystyrene beads in a blebbistatin treated cell. The beads in the cell shown in figure 5.2 are tracked for 1.5 hours and the tracks are center of mass corrected as visible in 5.3. Image a) shows the MSD for individual particles. The colors match the color of cell image. The MSD in image b) is the mean MSD for these particles. An exponential slope is fitted to the first part and the middle part of the curve with the slopes of 0.17 and 0.03. The particles have a diameter of 1 μm.
As this measurement contains a very long time span of continuous observation, it can be used to see changes in the MSD. This is accomplished by plotting a special form of a spectrogram. Using the MSD over the whole timespan as basis, the MSD is again calculated for time slices at different positions. For this measurement, a time slice for every 60 s is calculated. The slice width is 180 s, thus there is an overlap of neighboring slices, but an increased noise immunity. As the slices cannot show the very long step sizes, as they incorporate a maximum step size of 3 min, the spectrogram was limited to begin at the smallest and going to the 60 s step size. The spectrogram false color is calculated by dividing the MSD of the time slice through the MSD of the whole measurement. Thus higher values indicate a relative higher MSD at this time span compared to the overall measurement.

When looking at a spectrogram of the MSD as shown in figure 5.6, there is no clear time evolution of the MSD detectable. This might hint that the cell might have been already in a blebbistatin concentration equilibrium, no longer changing its behavior and stiffness due to the exposure. Therefore, the measurement protocol was slightly modified as described in the next paragraph.

5.2.2 Very long-time tracking of cells before and after treatment with blebbistatin

To make it possible to see the effect of blebbistatin on a single cell, the measurement protocol is adapted to track particles in a cell not treated with blebbistatin for 15 minutes and then adding blebbistatin directly while the cells are under the microscope and track the same cell and the same particles again for additional 45 minutes. The asymmetry of the measurement duration is chosen deliberately as the cell should be in a stable state before the treatment and might not be in a steady state afterwards, making longer measurement times necessary.

It is important to track the same particles twice. This allows for a measurement with comparable data. Therefore screenshots are taken to document the tracked particles before treatment to select the same particles again afterwards.

The measurement consists of several steps:
Figure 5.6: Spectrogram of the MSD of 1 µm polystyrene beads in a blebbistatin treated cell.

The MSD as shown in 5.5 is used as reference for the MSD computed for a 3 min long interval for each minute. The quotient of the so computed MSD curves and the complete MSD is plotted as a false color image. There seems to be no clear change in the behavior over time.
1. The glass bottom dishes containing the cells and beads are placed under the microscope with 1 ml imaging solution without blebbistatin. The lid is placed bottom up on the dish to allow easy removal without disturbing and shifting the dish significantly. This is necessary to find the cell again in step 6.

2. The beads are tracked for 15 minutes. At the beginning and at the end a screenshot is taken to see the exact cell shape and the tracked beads.

3. The lid is lifted off with caution to avoid shifting the dish around.

4. 1 ml imaging solution with blebbistatin is added to the cells in the dish with a pipette. The imaging solution is stored in an Eppendorf tube at room temperature in the same room to minimize changes in the imaging solution temperature.

5. The lid is again placed bottom up on top of the dish. Again caution is taken to avoid moving the dish.

6. The microscope is refocused and the cell is searched. The operator uses the screenshots to find the exact same cell and to select the exact same beads.

7. The beads are tracked for 45 minutes. At the beginning and the end a screenshot is taken to correlate the ROI of the beads with the ROI used in step 2.

While adding blebbistatin, the tracks of the particles are lost due to vibrations and optical disturbances. Therefore, a manual search to find the same cell again and refocusing is necessary. This induced a small delay of approximately two minutes until the measurement can start again, but gives the possibility to compare the behavior of the cell before and after treatment in the same surrounding.

It needs to be validated that the change in the measurement environment has no effect on the measured MSD. A null measurement is performed by adding a placebo instead of blebbistatin to ensure that the changes in the MSD are not influenced by temperature or other effects. For this a measurement was carried out but instead of adding the imaging solution with blebbistatin, only pure imaging solution is added after 15 minutes. Also an additional dummy measurement without cells and blebbistatin is done with an added temperature sensor to quantify the changes in temperature over the measurement.

When analyzing the results, the center of mass correction [Kle18, p. 24][RS13] is done with all tracked particles, irrespectively of being tracked in the other measurement part. This increases the quality of the correction. The MSD on the other hand is only com-
puted with the particles which are tracked in both measurement parts to allow a direct comparison.

To quantify the change in the MSD before and after the treatment, the MSD of each particle is compared using the Wilcoxon rank sum test for every time step\(^1\). This is done for every MSD time delay point where data is available. Those comparisons are shown in figure 5.8 for the measurement with blebbistatin treatment and figure 5.9 for the placebo treatment. When a cell is treated with blebbistatin, the MSD changes significantly, as indicated by more than three-times significance rejection of the null hypothesis, especially in the medium to high time delays. In contrast, for the null measurement the MSD changes not at all for the medium to high time delays as the null hypothesis cannot be rejected. The conclusion is therefore that treatment with blebbistatin indeed shows a reducing effect on the MSD.

Spectrograms over the measurements are used to quantify the changes over time. They are shown in figure 5.10 for the blebbistatin treated cell and figure 5.11 for the placebo measurement. The spectrograms are computed using 90 s windows with a 30 s window spacing. To make the spectrograms easier to compare, the same color bar scaling and \(x\)- and \(y\)-axis limits are used in both figures. Also the color bar is scaled logarithmically to make the smaller values more distinguishable.

The measurements show a decrease in the MSD when applying blebbistatin, especially for the longer time delays. This effect cannot be observed for the placebo experiment. The reduced MSD indicates a reduction of active cell movement, which happens at lower frequencies or longer time delays. The time range until the effect stabilizes can be estimated to be roughly some minutes. In the high frequency range there is less change in the MSD, indicating for less impact on the viscoelastic properties at this frequencies. Significant change of the MSD happens very fast and is not directly observable as there is a short gap in the recordings after adding the blebbistatin. The first measurements after adding blebbistatin show an already reduced MSD, which only reduces slightly in the following time windows. The crucial first moments after adding blebbistatin are not directly observable using the current setup as the microscope needs refocusing and the experimenter needs to find the cell and mark the same particles again.

\(^1\)Matlab `ranksum(a,b,'method','exact')`
5.2. Experimental results

Figure 5.7: Screenshots showing the cell and the tracked beads.

a) shows the beginning of the first measurement period of 15 minutes, b) the end of the first period, c) the begin of the second period of 45 minutes and d) the end of the second period. This cell is treated with blebbistatin.

As visible the rightmost particle in a) and b) tracked with light blue is not tracked in c) and d). The green and yellow tracked particles are swapped between a), b) and c) d). Red is swapped with orange and the right blue particle is tracked by light blue in c) d). This one-to-one match is done after recording the data.
Figure 5.8: Comparison of the tracked beads before and after blebbistatin treatment. Shown is the mean MSD curve before the blebbistatin treatment in blue and after the treatment in orange. 7 particles are tracked. To quantify the change, the Wilcoxon rank sum test is applied for every time delay for the 7 particles before and after treatment. In the middle and high time delays the MSD differs strongly, while on the lower time delays there is only a small and not significant change. The drawn in significance levels are $p < 0.05$ for *, $p < 0.01$ for ** and $p < 0.001$ for ***.
Figure 5.9: Comparison of the tracked beads before and after blebbistatin treatment without adding blebbistatin. Shown is the mean MSD curve before the blebbistatin treatment in blue and after the treatment in orange. 6 particles are tracked. To quantify the change, the Wilcoxon rank sum test is done for every time delay for the 6 particles before and after treatment. The treatment without any additional chemicals does not change the MSD in the middle to high time delays. Adding imaging solution increases the MSD for small time delays slightly. The drawn in significance levels are $p < 0.05$ for *, $p < 0.01$ for ** and $p < 0.001$ for ***.
Figure 5.10: Spectrogram of the MSD when adding blebbistatin.
The spectrogram is computed using a 30 s window. The reference MSD is the MSD before adding the blebbistatin and shown as the blue curve in figure 5.8. Visible is a strong reduction in the MSD after adding blebbistatin, which seems to further decrease over time, especially in the higher time delays. The lower time delays show a small reduction and no further decrease over time.
Figure 5.11: Spectrogram of the MSD when adding placebo imaging solution. The spectrogram is computed using a 30 s window. The reference MSD is the MSD before adding the placebo and shown as the blue curve in figure 5.9. There is no clear change in the MSD over the whole measurement excluding the short time delays at the end. The measurement was stopped eventually shortly after multiple particles could no longer be tracked due to the focus drift. This might explain the visible increase of the MSD at the very end.
Analyzing the change of the MSD after adding blebbistatin or placebo with an exponential decay is shown in figure 5.12 for the measurement with and without blebbistatin. An exponential decay mimics a reaction of a low-pass filter system to a step change. The step change is the addition of the blebbistatin and the system is represented by the concentration of working myosin in the cell as the output signal.

The measurement with added blebbistatin shows a clear decrease of the MSD for the long delay times and nearly no change over time for the shorter delay times below approx. 1 s. As the MSD for delay times below 1 s is reduced after treatment (figure 5.10), but exponential decay is no longer observable, the blebbistatin onset speed might have been too fast to be visible after single-digit minutes and only the steady state could be observed. To mitigate this, the measurement needs to be restarted faster after adding the active substance to show the onset, which is not a trivial task.

The placebo measurement shows poor fitting to the exponential and therefore the exponential decay time is small compared to the noise. The positive values of $b$ for the null measurement is correlated to the sharp increase in the MSD in the last minutes of the experiment, which is most likely a technical artifact. To partially mitigate this effect, the measurement was cropped as indicated in the figure 5.12. The higher RMSE values for the null measurement also indicate higher myosin activity and therefore more out-of-equilibrium processes which give rise to higher variations in the MSD values which would in turn increase the RMSE on their own. In summary, the measurement with added blebbistatin shows an exponential reduction for delay times over 1 s while no conclusive image can be drawn when only adding imaging solution without blebbistatin. The exponential decay model does not describe this case very well, suggesting no direct time-dependent change in the MSD.

5.2.3 Temperature of the imaging solution while measuring

To exclude temperature effects on the MSD when conducting the experiments a temperature sensor is added to a glass bottom dish. The experiment is conducted under the same conditions as with blebbistatin and cells. The measurements are taken using a Keythley 195A digital multimeter and a Pt100 temperature resistor with a resolution of 0.01 K. The measurement is done using a two point setup and compensating the temperature difference to the room thermometer after acclimatization of two hours in air.
5.2. Experimental results

Figure 5.12: Exponential fit of the time varying MSD for a measurement with and without blebbistatin.
The second part of the measurement as shown in figures 5.10 and 5.11 to the right of the white bars is fitted to an exponential horizontally for each MSD. The exponent $b$ of the fit $y = a \exp bx$ and the RMSE is plotted.
a) shows a measurement with and b) without blebbistatin treated cells.
For the calculation shown here, the time series of the MSD for b) is cropped at minute 48 (30 minutes after giving the placebo).
The room temperature is read off the room thermostat with a resolution of 0.5 K and linearly inter- and extrapolated. The temperature of the imaging solution is shown in figure 5.13. The temperature is slightly higher than the surrounding directly when starting the experiment and when adding extra solution, but this effect is small and below 1 K temperature increase. Otherwise, the temperature of the imaging solution is close to the room temperature and slightly lower. As the resolution of the room thermometer is only ± 0.5 K, no clear conclusion can be drawn from this comparison.

The absolute temperature influences the MSD linearly as the MSD is correlated to the temperature $T$ via

$$\text{MSD} = 2dD\tau = 2d \frac{k_B T}{6\pi \eta a} \tau$$ (5.2)

$$\text{MSD} \propto T$$ (5.3)

as laid out in chapter 3.2.2. Therefore, a temperature change of 1 K from 299 K to 298 K should only introduce a relative change of the MSD of -0.34 %. This is much smaller than the intra-measurement fluctuations and the difference when adding blebbistatin.

In conclusion, no big temperature changes can be observed in the imaging solution. Therefore, temperature variations can be excluded as a cause of the changes in the MSD with high certainty.

### 5.3 Discussion

A measurement of a blebbistatin treated cell to validate the sensor performance and cells receiving blebbistatin and placebo treatment while undergoing measurement were analyzed.

Adding blebbistatin to a cell introduces a reduction in the MSD at middle to long timescales. This effect has an onset time faster than single-digit minutes. The measurements show no significant change when using only imaging solution without blebbistatin as a negative control. This is in accordance with the literature, as blebbistatin is a myosin inhibitor [DAA07; Str+03; WS87]. A temperature measurement shows no rel-
Figure 5.13: Temperature of the imaging solution while measuring. The temperature is slightly increased directly when starting the experiment and adding solution, otherwise the difference of the temperature to room temperature is well below $<1 \text{ K}$. 
evant temperature changes which could shadow the measured effects. Therefore, the
temperature has no significant effect on the measurement.

For further studies, it seems to be crucial to make sure to track as many particles as
possible and avoid losing them due to focus problems which makes it difficult to con-
tinuously track the particles. In the raw data there is also noise visible in the form of
oscillations, which are most likely introduced by the environment. A good and stable
decoupled microscope eases the tracking and makes it more precise. This is of course
true for all tracking-based microrheology experiments. Various precursor experiments
showed that the microscope should at least be mounted on a stone plate to prevent
oscillations from the desk.

It is not directly possible to resolve time-depended effects of blebbistatin, as the onset
seems to be faster than single digit minutes. Only a slow creep in the high time delay
values could be observed. The reduction of the MSD was nevertheless visible and a
decrease of the MSD over the time was measurable. With an improved experimental
setup where substances can be introduced without inducing vibrations and with minimal
disturbance of the experiment allowing a continuous tracking of the particles, it should
be possible to study this time dependence much easier and more in depth. A possible
implementation would be the use of a flow chamber with a constant flow of solution,
which can then be primed with the substance after the initial baseline measurement.
Some considerations are needed to not alter the flow pressure and speed to not disturb
the flow profile in direct vicinity of the tracked particles. A possible implementation of
such an experiment is shown in figure 5.14.

This experiment is of course not exhaustive. As a hint for further research blebbistatin
treated cells show a time-dependent effect in the MSD and thus in the cell mechanics.
This effect and its time behavior might be explained by diffusive processes as the bleb-
bistatin needs to reach the cell and the cell interior to function. Additionally, inhibition of
myosin might not directly change the viscoelastic properties but has a small time delay
until it fully sets in.

A complete analysis of those effects of blebbistatin and an extension to other substances
and cell types would exceed this work and should be investigated in further studies. The
prime goal of this work has been shown as these measurements show that the novel
algorithm is able to track particles with high speed and high precision over long time-
Figure 5.14: Possible layout of a non-disturbing drug injection. The most simple form of a flow chamber in which the sample cells are seeded and adhering to the walls. A syringe is pumping imaging solution through the chamber. The Pump is connected to a long tubing with a small diameter. This tubing is filled with drug-free solution and the syringe is filled with drug laden solution. Clean solution is pumped in the beginning, after the tubing has been completely filled with drug laden solution, this solution reaches the flow chamber. This measurement introduces no changes of pressure and flow or other disturbances when drug reaches the cell. A separate study about the diffusive gradient of the drug in the solution when first reaching the flow chamber needs to be conducted.

spans and that this enables new insights into biomechanics and biophysics. The last paragraphs proof, that this is indeed possible, therefore validating the sensor and the design goals.
6 Summary

This work shows that it is possible to track particles with high precision and speed in real time using a modified image fitting algorithm. The speed increase is significant and reaches a theoretical speedup of roughly 1000 % or being 11 times faster than the original algorithm. The true speedup depends on the input data as the original algorithm has a speed dependence on the size of the tracked particles. This modifications achieve on-par precision as state-of-the-art tracking programs [Rog+07].

This new tool allows for tracking particles over much longer time spans with higher speed than previously possible [Tas+12]. This extends the simultaneously measurable frequency spectrum to a range from $10^{-3.7}$ Hz to $10^{3.4}$ Hz, spanning over seven decades, as shown in experiments. If needed the range can be extended to $10^{4}$ on the high frequency end and basically arbitrary low on the low frequency end. With this novel sensor new insights into the mechanical behavior of cells are possible.

The developed algorithms allow the tracking of particles in optical images with very high speed and very high precision and simultaneously use very little resources and allow very long recording lengths. The length of the recording is only limited by the size of the data storage and can exceed multiple hours without any problem. This solves the need for sampling schemes as in [Phi+16]. Computing the MSD over a very long range of available frequencies as done in [Tas+12] is now possible with more than one particle. The complete range of frequencies is still not available as the very high frequencies over 10 kHz are not accessible with the current sensor. This is an inherent limitation of the image sensor.

As the tracking is done online, the speed of measurements as well as the analysis time is increased. This allows much faster measurement and development cycles in science and industry, thus increasing the amount of science and work which can be done in a given time frame. By giving the user direct control over the tracking and an immediate
feedback of a setting change, repetition of experiments due to wrong settings can be avoided in most cases, thus saving resources.

With this novel tool, several experiments could be expanded, as an example an experiment like described in [Wu+20] could expand the MSD range from $[10^{-1} \text{ Hz}, 10^1 \text{ Hz}]$ to bigger ranges. This rate might be potentially limited by the fluorescent labeled particles and therefore low light intensities.

The preliminary measurements on cells with blebbistatin indicate a very fast onset of the effect of the substance on the cell stiffness, as the change in MSD of embedded beads is visible when comparing before- and after-treatment measurements. A time evolving effect of the MSD when treating cells with blebbistatin can be observed, suggesting a non-instantaneous effect on cell mechanics. As blebbistatin directly binds to the myosin [Kov+04], a fast onset is expected. The observed decrease of the MSD is most pronounced in the high time delay values or in the low frequency regime. A possible explanation is a higher influence of active movement on the MSD in the higher time delays compared to the lower time delays.
7 Outlook

The techniques presented in this work allow complicated algorithms to be adapted to real-time and direct point-of-measurement analysis, which enables the experimenter to directly react to changes in the experiment. Direct applications of this feature include feedback loops to control the experiment. This can include moving the field of view with the particles or refocussing by using a piezoelectric stage. As the focus drift over long time spans can be a problem in the experiments shown here, this would have an direct impact on the ease of doing research.

When computing the eccentricity and the orientation angle of the tracked particles on the sensor, it should also be possible to adapt the tracking technique to use rotated sampling grids to track very elongated shapes with high precision and speed.

By using this sensor in cell biology it is possible to monitor the health and biological status of a cell without disturbing it. By computing the MSD after a short sampling interval it should also be possible to react to changes in cell mechanics and apply treatment to get or keep the cell in a desired state.

The analysis of blebbistatin impact on cells is not finally analyzed. No assessment of different concentrations and cell types has been done. As blebbistatin binds very selectively to different variants of myosin [Lim+04], different cell types might with a high likelihood yield different results. Also the experiments can be conducted with a variety of other drugs. As the experiments can be done in a single day if needed, a large number of drugs can be screened in this way. One can think of applying drugs to healthy and cancer cells of the same patient to screen for applicable drugs with minimal impact on the patient while simultaneously increase the therapeutic effect.
7.1 Further improvements

The algorithm can be used to compute the angle $\theta$ and the eccentricity $e$ of the particle being tracked. This information is currently not used in the tracking algorithms. Those two values can be used in the novel algorithm, as it is not limited to a simple scaled and translated sampling of the image and can theoretically use nonuniform stretched grids. This would only require to adjust the rescaling of the factors $p'$ to $p$ by using the scaling factors for both dimensions. By using $\theta$ and $e$ the sampling grid could rotated and scaled with the apparent orientation of the particle to ease the tracking of elongated particles. Whether this improves the tracking results has to be evaluated. In this work, this is not realized as it would require the usage of trigonometric functions to the main tracking part and therefore create additional high hardware requirements. However, with careful tuning and a larger FPGA this should be achievable. This could improve the tracking performance of stretched images or for very elongated particles.

If the need for the tracking of more than 8 particles arises in the future, a bigger version of the sensor with up to 32 ROIs is already commercially available and can be used. To adapt more ROIs on the FPGA a larger one might be needed. Faster tracking speeds of the tracking algorithm can be achieved using newer FPGAs, allowing for higher design speeds.

To achieve image rates over 10 kHz, a different image sensor is needed. This could either be a reconfigurable sensor with an equivalent flexibility as the currently used sensor or a fullframe sensor with a significant higher speed. The higher speed needs to be handled by the FPGA, but does not need to be handled by the systems downstream as still only the particle positions are saved. This approach would also limit the particle count only by the available FPGA resources. A modern FPGA can have up to 20 time the resources than the FPGA used in this work, thus linear scaled this solution could then track up to 160 particles. On the other hand this would incur much higher hardware costs.
7.2 Further applicable fields

Besides the possible applications of a particle tracking sensor in the laboratory in general and specifically the microrheology field, there are various other fields where the sensor can be used. It might be useful in tracking macroscopic objects, for example airplanes in the sky. As the precision of tracking dot-like objects is sub-pixel accurate, the sensor acts like a camera with a hundred times higher resolution. Thus, it is a cost effective solution for a high precision and at the same time wide field-of-view tracking device. This allows the usage of a very small number of sensors with suitable lenses to observe the complete sky. This can be useful for collision tracking in autonomous flying drones and airspace controlling for airports and restricted airspaces.

It also can be used as an eye-tracking device with very high speed and low input delay. Further applications could be employed in sports, like a goal camera in soccer or the hawk-eye in tennis. It could also be used to control a robot as a training partner in sports like tennis or badminton.
List of Figures

3.1 Fluorescent labeled cell cytoskeleton and nucleus .......................... 6
4.1 Labeled PCB layout of the custom designed board .......................... 23
4.2 Screenshot of the running sensor ............................................. 25
4.3 Comparison of the filter functions with a random noise image, Matlab and FPGA result ................................................. 29
4.4 Sketch of the transformation to the standard case ......................... 39
4.5 The weights matrix in false color ............................................ 40
4.6 Comparison of convergence with different interpolation methods .... 42
4.7 Visualization of the dead region while tracking particles ................ 47
4.8 Tracking process and timing table for a single particle with adaptive ROI configuration .................................................. 49
4.9 Parameter evolution while tracking a gaussian peak ..................... 51
4.10 Deviation from the ideal result with a simulated random walk ........ 52
4.11 Influence of noise on algorithm output .................................... 54
4.12 Tracking of beads in and on a NIH/3T3 fibroblast ....................... 56
4.13 Detailed plots of the differences between the polyparticletracker and the novel algorithm ............................................... 57
5.1 Sketch of the experimental setup ............................................. 60
5.2 Phase contrast images of the blebbistatin treated cell ..................... 63
5.3 Center of mass movement for a very long measurement ............... 64
5.4 Apparent particle radius of 1 μm polystyrene beads in phase contrast . 65
5.5 MSD of 1 μm polystyrene beads in a blebbistatin treated cell ........ 66
5.6 Spectrogram of the MSD of 1 μm polystyrene beads in a blebbistatin treated cell ......................................................... 68
5.7 Screenshots showing the cell and the tracked beads ....................... 71
List of Figures

5.8 Comparison of the tracked beads before and after blebbistatin treatment . 72
5.9 Comparison of the tracked beads before and after blebbistatin treatment without adding blebbistatin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
5.10 Spectrogram of the MSD when adding blebbistatin . . . . . . . . . . . . 74
5.11 Spectrogram of the MSD when adding placebo imaging solution . . . . 75
5.12 Exponential fit of the time varying MSD for a measurement with and without blebbistatin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
5.13 Temperature of the imaging solution while measuring . . . . . . . . . . 79
5.14 Possible layout of a non-disturbing drug injection . . . . . . . . . . . . . 81
List of Tables

4.1 Errors and scaling factors for the fixed-point matrix multiplication . . . . . 28
4.2 Different sampling/interpolation methods and their smoothness, flatness,
    and computational complexity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
Bibliography


[Lim+04] J. Limouze et al., “Specificity of blebbistatin, an inhibitor of myosin II”, eng, 
*Journal of muscle research and cell motility* 25 (4-5) (2004), Comparative 


[Mü+04] N. Mücke et al., “Assessing the flexibility of intermediate filaments by atomic 
force microscopy”, eng, *Journal of molecular biology* 335 (5) (2004), Comparative 

[Mar+15] I. Martin et al., “Model-based analysis of keratin intermediate filament as-
ISSN: 0022-3727, DOI: 10.1088/0022-3727/48/37/375401.


[Mas+97] T. G. Mason et al., “Particle Tracking Microrheology of Complex Fluids”, 
*Physical review letters* 79 (17) (1997), pp. 3282–3285, ISSN: 0031-9007, 
DOI: 10.1103/PhysRevLett.79.3282.

[Miz+08] D. Mizuno et al., “Active and Passive Microrheology in Equilibrium and Nonequi-

to Fluidlike Transition and Long Time Superdiffusion in an Evolving Cell 

[MS99] F. C. MacKintosh and C. F. Schmidt, “Microrheology”, *Current Opinion in 
Colloid & Interface Science* 4 (4) (1999), PII: S135902949900109, pp. 300– 


[SA05] I. Schofield and A. Alimohammad, “Parallelized QR decomposition using GPUs”, 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE) (Edmonton, AB, Canada), IEEE, 5.05.2019 -


Bibliography


Glossary

**AXI** Advanced eXtensible Bus Interface, A bus standard for interfacing different peripherals with each other, uses 5 AXI stream interfaces and a master-slave architecture. 22, 24

**buck converter** switching power supply voltage down converter. 23

**HLS** high level synthesis. Uses C and C++ dialects to generate hardware. 28, 30

**IO bank** IO pins grouped together by the same power supply. 22

**ping-pong buffer** A buffer which is twice as large as needed, where one side simultaneous fill one half where the other side read the other half, when both sides are finished, the halves are simply flipped. 28

**softcore** A software defined processor that is synthesised and implemented in an FPGA. 30
Acronyms

**ADC** analog-digital-converter. 15

**AFM** atomic force microscope. 8, 10

**ARM** Advanced RISC machines. 22, 24

**BRAM** block memory. 22, 24, 28, 46

**CLB** configurable logic block. 46

**CMOS** complementary metal-oxide-semiconductor. 1, 2, 20, 47

**DDR** double data rate. 20

**DMA** direct memory access. 26

**DNA** deoxyribonucleic acid. 4, 5

**DSP** digital signal processors. 22, 28, 46

**FF** flip flop. 22

**FPGA** field programmable gate array. 1, 2, 20, 28, 29, 58, 85, 87

**GPIO** general purpose IO. 22–24

**GPU** graphical processing unit. 37, 62

**I^2C** inter integrated circuit. 22

**LDO** Low dropout linear regulator. 23

**LMB** local memory block. 24
**LUT** lookup table. 22, 46

**LVDS** low voltage differential signalling. 20, 23

**MSD** mean squared displacement. 11–13, 61, 63, 66–70, 72–78, 80, 82–84, 87, 88

**NaN** not-a-number. 46, 62

**PBS** phosphate buffered saline. 60, 61

**PCB** printed circuit board. 22–24, 87

**PHY** physical interface chip. 22

**PL** programmable logic. 22, 24

**PLL** phase locked loop. 20

**PS** processing system. 22

**RMSE** root mean squared error $\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2}$. 76, 77

**RNA** ribonucleic acid. 4, 5

**ROI** region of interest. 21, 25, 47–49, 85

**RTC** real-time clock. 22

**SGMII** serial gigabit media independent interface. 22

**SOC** system on a chip. 22–24, 30

**SPI** serial peripheral interface. 20, 26

**SRAM** synchronous RAM. 21, 22

**TMDS** transition minimized differential signalling. 23

**TTL** transistor-transistor logic. 20, 26

**ULP** unit in the last place. 46

**VHDL** very high speed integrated circuit hardware description language. 28, 58
Acknowledgment

The author acknowledges funding by the graduate college Pulmosens of the Deutsche Forschungsgesellschaft (GRK2203).

I want to thank numerous people who helped me while working on this project. I want to thank especially the following people

- Prof. Dr. ETH Zürich Othmar Marti
- Prof. apl. Dr. Berndt Koslowski
- Dr. Tobias Neckernuss
- Dr. Daniel Geiger
- Carolin Grandy
- Anngrit Lutz

I also want to thank the nasty virus SARS-CoV-2 for forcing everybody to their homes and therefore giving me time to write this thesis.
Publications

1 Paper


2 Patents


3 Posters

Chapter 7. Publications


4 Talks

All talks have been held at the seminar of the Institute of Experiment Physics at Ulm university.

• 2020/06: Very long time continuous microrheology. Tracking particles over very long timespans

• 2020/01: Very long time continuous microrheology. First results

• 2019/06: Very long time continuous microrheology. In detail DSPs

• 2019/01: Very long time continuous microrheology. In detail about particle tracking

• 2018/06: Very long time continuous microrheology. Algorithms and used hardware

• 2018/01: Very long time continuous microrheology. Idea and conceptional design
Erklärung des Verfassers:

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe und alle wort- oder sinngemäßen Übernahmen aus anderen Werken unter Angabe der Quelle kenntlich gemacht worden sind.

Ulm, den ...............................................................

Jonas Pfeil