Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Real-time phase reconstruction for dielectrophoretic micromanipulation

Bc. Viktor-Adam Koropecký

Supervisor: Ing. Martin Gurtner
Field of study: Cybernetics and Robotics
May 2021

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Koropecký Viktor-Adam**	Personal ID number:	**465906**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Real-time phase reconstruction for dielectrophoretic micromanipulation

Master’s thesis title in Czech:

Fázová rekonstrukce pro dielektroforetickou mikromanipulaci v reálném čase

Guidelines:

Design, implement and test a real-time algorithm for phase reconstruction in lensless digital holographic microscopy used on a platform for dielectrophoretic micromanipulation. The algorithm will clear twin-images from back-propagated holograms and display the resulting pictures on a monitor. The algorithm will run in real time on a GPU on a dedicated computing device and be implemented in CUDA.

1. Review state-of-the-art methods for phase-reconstruction and choose the most promising one for real-time use.
2. Prototype the chosen method in Matlab or Python not taking into account the real-time aspect.
3. Optimize the method by implementing it in CUDA so that it can run in real-time (at least few frames per second).
4. Implement the final method on the used platform for dielectrophoretic micromanipulation.

Bibliography / sources:

[1] M. Gurtner and J. Zemánek, "Twin-beam real-time position estimation of micro-objects in 3D," Meas. Sci. Technol., vol. 27, no. 12, p. 127003, 2016.
[2] Y. Wu and A. Ozcan, "Lensless digital holographic microscopy and its applications in biomedicine and environmental monitoring," Methods, vol. 136, pp. 4-16, Mar. 2018
[3] F. Momey, L. Denis, T. Olivier, and C. Fournier, "From Fienup's phase retrieval techniques to regularized inversion for in-line holography: tutorial," J. Opt. Soc. Am. A, vol. 36, no. 12, p. D62, Nov. 2019

Name and workplace of master’s thesis supervisor:

Ing. Martin Gurtner, Department of Control Engineering, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **28.01.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

Ing. Martin Gurtner	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I owe my supervisor, Martin Gurtner, a great debt of gratitude for all of his help and guidance throughout the very hectic last year. I would also like to thank him for the opportunity to start working on this project last summer.

My thanks also goes to Fabien Momey, who, in the very brief correspondence that we shared, managed to point me in the right direction in my research.

Of course, I have to thank my family, friends and Anička, who have all supported me immensely throughout the past several months that I spent working on this thesis. Thanks to them it was a pleasurable experience.

This thesis is the culmination of almost a year of work and it could not have been finished without the amazing people on this list.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 21. May 2021
Bc. Viktor-Adam Koropecký

Abstract

This diploma thesis focuses on designing, implementing and testing a real-time algorithm for phase reconstruction in lensless digital holographic microscopy. The chosen method approaches phase reconstruction as an inverse problem regularized by sparsity and positivity constraints. It uses a proximal gradient method as its optimization strategy. The algorithm was first prototyped in Matlab and tuned on recorded holograms. It was then reimplemented in CUDA and optimized for real-time use with the help of warm-starting. Finally, it was implemented on a platform intended for feedback dielectrophoretic micromanipulation where it runs on a dedicated GPU computing device, NVIDIA Jetson AGX Xavier. Here it provides real-time hologram reconstructions with well-suppressed twin-images. These reconstructions are displayed on a monitor connected to the platform. The implementation presented in this thesis provides quality hologram reconstructions in speeds that exceed commonly used methods. The resulting CUDA implementation, as well as the Matlab prototype, are published online as open-source software.

Keywords: phase reconstruction, digital holographic microscopy, GPU, CUDA, regularized inversion, FISTA

Supervisor: Ing. Martin Gurtner
Faculty of Electrical Engineering,
Karlovo náměstí 13,
Praha 2

Abstrakt

Tato diplomová práce se zaměřuje na návrh, implementaci a testování algoritmu pro fázovou rekonstrukci v reálném čase pro digitální holografickou mikroskopii bez použití optických prvků. Vybraná metoda přistupuje k problému jako k inverzní úloze regularizované na řídkost a nezápornost řešení. Pro optimalizaci úlohy je využita proximální gradientní metoda. Prototyp algoritmu byl nejprve implementován v Matlabu, kde byl nastaven za pomoci nahraných hologramů. Algoritmus byl poté reimplementován v CUDA a optimalizovaná pro běh v reálném čase za pomoci zrychleného startování. Nakonec byl algoritmus implementován na platformu určenou pro zpětnovazební dielektroforetickou mikromanipulaci, kde běží na dedikovaném grafickém vypočetním zařízení, NVIDIA Jetson AGX Xavier. Na této platformě algoritmus v reálném čase poskytuje rekonstrukce hologramů s dobře potlačenými artefakty (twin-images). Tyto rekonstrukce jsou pak zobrazeny na displeji, který je připojený k platformě. Implementace prezentována v této práci poskytuje kvalitní rekonstrukce hologramů v rychlostech, které přesahují běžně používané metody. Výsledná implementace v CUDA i prototyp v Matlabu jsou dostupné online ve formě otevřeného softwaru.

Klíčová slova: fázová rekontrukce, digitální holografická mikroskopie, GPU, CUDA, regularizovaná inverze, FISTA

Překlad názvu: Fázová rekonstrukce pro dielektroforetickou mikromanipulaci v reálném čase

Contents

Chapter 1

Introduction

This thesis focuses on designing, implementing and then testing a real-time algorithm for phase reconstruction in lensless *digital holographic microscopy* (DHM). This algorithm is supposed to run on a platform designed for feedback dielectrophoretic micromanipulation [2]. The algorithm will be used for estimating the phase of holograms captured by this platform after which the holograms will be reconstructed and the results will be displayed on a monitor connected to the micromanipulation platform.

In DHM, an image sensor is used to capture interference patterns, also called holograms, which are formed when a scattering microscopic sample is illuminated by a source of coherent light. These captured holograms can then be reconstructed to show the actual geometry of observed objects. To do such a reconstruction properly, both the amplitude and phase of the captured interference patterns have to be known. However, since an image sensor can only capture the intensity of the interference patterns, the reconstructions are affected by the so-called twin images which obscure the geometry of the observed objects. This is caused by the lack of information about the phase of the captured holograms. The occurrence of twin-images is visible in figure 1.1, where a reconstruction of a hologram recorded on the micromanipulation platform is compared to an image of the same situation recorded by a regular optical microscope.

Phase reconstruction algorithms, such as the one that will be implemented in this thesis, are procedures of estimating the phase of the recorded holograms. Reconstructions of holograms with properly estimated phase have suppressed twin-images and thus the geometry of the observed objects is more clearly visible.

First, a suitable phase reconstruction algorithm will be selected from a review of appropriate already existing phase reconstruction algorithms. Then a prototype of the algorithm will be implemented in Matlab to test its func-

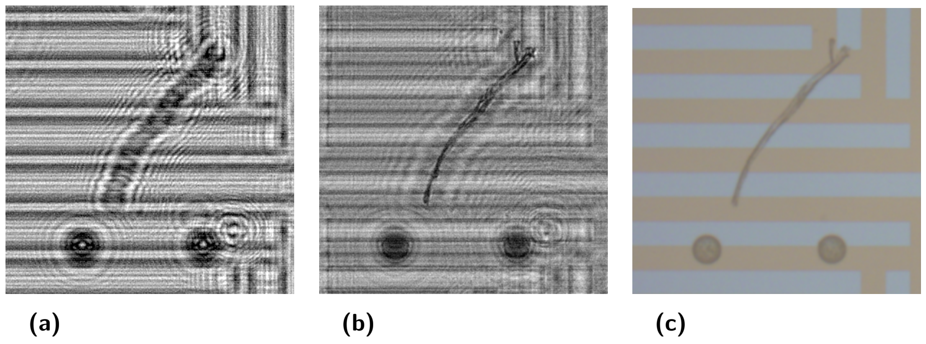


Figure 1.1 Recorded hologram of two polystyrene particles with a diameter of approximately 50 µm next to a dust particle. (a) The original hologram, (b) reconstruction of the hologram affected by twin-images, (c) image captured on the regular optical microscope

tionality. After that, the algorithm will be implemented in CUDA so that it can run on a *graphical processing unit* (GPU) and it will be optimized for real-time use. Finally, the algorithm will be implemented on the embedded computing device connected to the dielectrophoretic micromanipulation platform.

The platform [1, 2] has already been developed and has been in use by the research group *Advanced Algorithms for Control and Communications* (AA4CC) at the Faculty of Electrical Engineering of the Czech Technical University in Prague. Thus, the algorithm has to be integrated into the already existing code implemented on the platform alongside algorithms used for estimating and tracking the positions of the manipulated particles. The integrated algorithm should not restrict in any way the original functionality of the existing code and it should completely suppress twin-images from reconstructions of holograms captured on the platform.

1.1 Structure of the thesis

The author will first provide a brief introductions to in-line digital holography, then he define the most commonly used method of hologram reconstruction called backpropagation, introduce the twin-image problem and briefly describe the micromanipulation platform in Chapter 2. Later, the author will review already existing phase reconstruction methods in in-line digital holography and discuss their suitability for use with the provided platform in Chapter 3. In the following Chapter 4, the selected method will be described in more detail along with possible improvements to the methods for its better real-time implementation. After that, all three implementations of the selected method will be briefly described in Chapter 5. The implementations will then be experimented on and evaluated in Chapter 6. And finally, in Chapter 7 the results achieved in this thesis will be summarized.

Chapter 2

Theoretical background

This chapter provides a short introduction to in-line digital holography, followed by a description of the backpropagation method. Then, the twin-image problem will be introduced and, finally, the provided micromanipulation platform will be described in detail.

2.1 In-line Digital Holography

Digital holography can be defined as a group of methods for recording an interference created when a plane reference wave interferes with a wave that interacted with an observed object. These patterns are recorded on an electrical image sensor, such as a *complementary metal oxide semiconductor* (CMOS) image sensor. Digital holography is a broad field a detailed review of which can be found in [3].

A plethora of hardware arrangements can be used in digital holography, however, in this thesis, the focus is mainly on the simplest arrangement that is the in-line digital holography, which is derived from the original setup invented by Gabor [4].

In in-line digital holography, the image sensor is placed directly beneath the observed sample, which is then illuminated from above by a coherent light source. Considering that a usual observed object is at least partially transparent and small, the planar wave that interacts with the object is modified by a combination of several phenomena: It can be either diffracted, absorbed, reflected and or it can pass through the object itself. The sensor then records the intensity of the generated interference patterns in which both the position and shape of the object are partially — due to the lack of any phase information in the recording — encoded.

The quality of captured interference patterns stems from the coherence of the used light source. The more both spatially and temporally coherent a light source is, the more visible the interference patterns or holograms are.

It should be noted that lensless in-line digital holography setups can be an incredibly cheap alternative to regular optical microscopes when used for observing microscopic objects. The most expensive part of such a setup is usually the image sensor. Additionally, regular optical microscopes tend to be much larger than the in-line digital holography setups.

2.2 Backpropagation

It is very difficult to deduce any information about the shape of the observed object from just the recorded interference patterns, regardless of their quality. For this reason, one can utilize *backpropagation*, which is a method that convolves the hologram with some specific propagator function, otherwise called a backpropagation kernel. This operation simulates what the planar wave captured by the sensor would look like at a different axial distance above the sensor. A result of backpropagation can be seen in figure 1.1. Further in this thesis, *sensor plane* refers to the plane on which the hologram is recorded and *sample plane* is the plane where an observed object is located.

In this thesis, the used propagator function is the Fourier transform of the Rayleigh-Sommerfeld diffraction integral [5], which can be calculated as

$$H_{-z}(f_x, f_y) = \begin{cases} \exp\left(-i2\pi z \frac{n}{\lambda}\sqrt{1 - \left(\frac{\lambda f_x}{n}\right)^2 - \left(\frac{\lambda f_y}{n}\right)^2}\right), & \sqrt{f_x^2 + f_y^2} \leq \frac{n}{\lambda}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

where H_{-z} is the backpropagation kernel, n is the refractive index of the medium between the sensor plane and the sample plane, λ is the wavelength of the said light source, z is the backpropagation distance and $f_y,\ f_x$ are the spatial-frequency coordinates. These coordinates depend on the size of individual pixels dx on the image sensor.

Since the propagator function is acquired in the Fourier domain, the actual operation of backpropagation is done in the form of

$$I_z(x, y) = \mathcal{F}^{-1}\{H_{-z}(f_x, f_y)\mathcal{F}\{I_H(x, y)\}\}, \tag{2.2}$$

where I_z is the backpropagated image, I_H is the recorded intensity, $x,\ y$ are the image coordinates and $f_x,\ f_y$ are the spatial-frequency coordinates. $\mathcal{F}$ and $\mathcal{F}^{-1}$ are spatial Fourier and inverse spatial Fourier transforms, respectively.

An image can also be *propagated*, to simulate how the image would look further from the light source. This means that the only difference between the backpropagation and the propagation operations lies in the direction in which the operation propagates the complex planar wave. This difference between the two kernels will be signified by a negative sign with the distance parameter for the backpropagation kernel H_{-z}, which is omitted in propagation. The sign is also omitted during the calculation of propagation kernels in (2.1), from which it can be easily seen that a propagation kernel is just a complex conjugate of the appropriate backpropagation kernel.

The phase reconstruction method designed later in this thesis utilizes both propagation and backpropagation to alternate projections between the sample plane and the sensor plane.

Further in this text, the notation for both the propagation or backpropagation operation at (2.2) will be simplified by the use of the expression of

$$\texttt{prop}(h(f_x, f_y), a(x, y)) = \mathcal{F}^{-1}\{h(f_x, f_y)\mathcal{F}\{a(x, y)\}\}, \tag{2.3}$$

where the first input h is the input kernel that drives the operation and a is the input complex planar wave that undergoes the operation.

2.3 Twin-Image Problem

Considering that the observed object is located only at the sample plane, the complex amplitude of its interaction with the reference planar wave can be written in vector form as

$$A_z(x, y) = r(x, y) - a_z(x, y), \tag{2.4}$$

where r is the reference planar wave at the sample plane, which is considered to have an amplitude equal to 1, A_z is the complex amplitude at the sample plane and a_z is the disturbance caused by the observed object.

The complex amplitude at the sensor plane is then given as

$$A_H(x, y) = \texttt{prop}\left(H_z(f_y, f_x), [r(x, y) - a_z(x, y)]\right), \tag{2.5}$$

where H_z is the propagation kernel with a distance parameter of z.

The reference planar complex wave is assumed to be constant in all coordinates. When such a wave is propagated to the sensor plane it undergoes a phase shift that can be characterized by the following equation:

$$\texttt{prop}(H_z(f_x, f_y), r(x, y)) = H_z(0, 0)r(x, y). \tag{2.6}$$

In this thesis, an assumption is made that when the reference wave at sample plane is propagated to the sensor plane, its value changes to a real one, more specifically that

$$\mathtt{prop}(H_z(f_x, f_y), r(x, y)) = 1. \tag{2.7}$$

This assumption is made strictly for simplification purposes and should not negatively impact any results computed under its influence. Taking the squared magnitude at each pixel, or intensity, which is what the image sensor records, and taking into account the previous assumption, one gets

$$I_H(x, y) = |1 - \mathtt{prop}(H_z(f_x, f_y), a_z(x, y))|^2, \tag{2.8}$$

and after further expansion [6], one gets

$$I_H(x, y) = 1 - \mathtt{prop}\,(H_{-z}(f_x, f_y), \bar{a}_z(x, y)) - \mathtt{prop}(H_z(f_x, f_y), a_z(x, y)) = \tag{2.9}$$

$$= 1 - 2\mathrm{Re}\{\mathtt{prop}(H_z(f_x, f_y), a_z(x, y))\}, \tag{2.10}$$

where $\bar{a}_z(x, y)$ is the complex conjugate of $a_z(x, y)$ and $\mathrm{Re}\{a_z(x, y)\}$ is the real part of the complex number $a_z(x, y)$.

If this result is backpropagated using a kernel H_{-z}, one gets

$$I_z(x, y) = 1 - a_z(x, y) - \mathtt{prop}(H_{-2z}(f_x, f_y), \bar{a}_z(x, y)), \tag{2.11}$$

where the final term is the mathematical expression of a twin-image.

2.4 Used platform

The experimental platform for feedback micromanipulation used in this thesis was developed in the past years by the AA4CC group [1, 2]. A cross-section of the platform along with its 3D render can be seen in figure 2.1.

The platform consists of a small pool of distilled water in which the observed objects are submerged. In the case of this thesis, these objects are polystyrene particles of approximately 50 µm. An electrode field is situated at the bottom of this pool. Through this electrode field, dielectrophoretic micromanipulation is performed. Finally, there is a small gap below the electrode field with an image sensor placed at the bottom. This apparatus is covered by a lid to protect it from surrounding light sources. The pool is then illuminated by two LEDs attached to the top of the protective lid. Each of these LEDs is

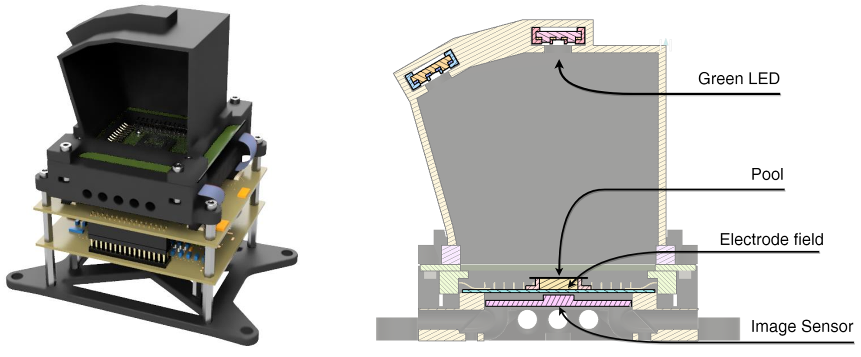


Figure 2.1 The platform for dielectrophoretic micromanipulation without the attached computer and a power supply: (a) 3D render of the platform with opened cover, (b) side cross-section of the platform with labels

situated behind its own 50 µm pinhole, which allows their light to be partially coherent.

One of the light sources is situated some distance directly above the pool and has a green peak wavelength of 515 nm. The other light source is placed slightly off to the side and has a red peak wavelength of 630 nm. This is really important for determining the position of observed objects as the green channel allows precise tracking of lateral position with the red channel showing the axial position.

The image sensor used is the LI-IMX477-MIPI-M12 image sensor with a resolution of 4056×3040 pixels. A single pixel of this sensor is a square with a side of 1.55 µm. It uses three colour channels encoded in the YUV-420 format with two green pixels per one red pixel. Having different colour channels, it is easily possible to separate the interference patterns caused by individual light sources and thus generate two different holograms from one recording. In this thesis, only the green channel is considered, since the geometry of the observed samples is not skewed on this channel. Since the green channel has double the pixels of the red channel, the hologram on the green channel is clearer. A captured hologram of the complete observable area on the green channel can be seen in figure 2.2.

The image sensor is connected via the *camera serial interface* (CSI) to Jetson AGX Xavier, an embedded computer developed by NVIDIA. This computer was designed with image processing in mind, so it is a great fit for the problem of phase reconstruction. All graphical computations needed for the micromanipulation to take place are done on this computer.

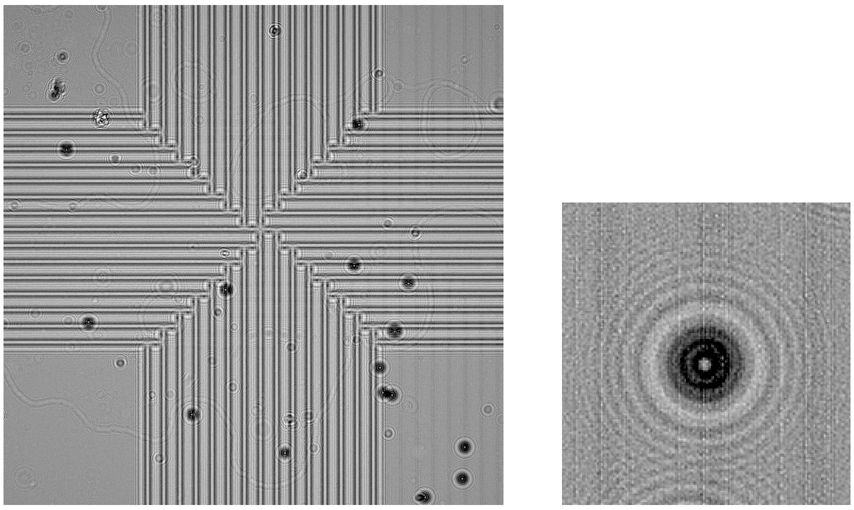


Figure 2.2 Hologram captured on the green channel. (a) Full view of the pool with submerged microparticles above the electrode field. (b) A close up of a single interference pattern caused by the observed polystyrene microparticles.

The complete setup poses a couple of challenges for phase reconstruction: Firstly, the backpropagation operation assumes that the medium between the sample plane and the sensor plane is homogeneous. This is however not the case. The light coming from the light source has to pass through multiple layers with different refractive indices before reaching the image sensor. This means that when a hologram is backpropagated to some distance z using a kernel calculated from one refractive index, the resulting complex planar wave does not actually match the complex planar wave at the distance z but rather a complex planar wave shifted with respect to the refractive indices of the different mediums.

Secondly, the recorded holograms are densely populated and interference patterns generated by one object interfere with interference patterns generated by other objects. This can lead to inaccuracies in the reconstructions. The chosen phase reconstruction method should have no trouble working with this setup.

Chapter 3
Related works

In this chapter, a brief review of several methods used for phase reconstruction is provided.

The methods discussed in this chapter will be compared to one another based on their suitability for work with the provided micromanipulation platform described in section 2.4. That means the optimal method should have no problem eliminating twin-images from both the electrode field and the observed particles. Further, it needs to be designed for work on an in-line digital holography platform without the use of optical lenses and it should work properly without the need for multiple holograms needed per phase reconstruction. It also has to be suitable for real-time implementation.

Overall, three different groups of methods will be discussed in this chapter [7, 8]. The first group focuses on the iterative elimination of twin-images directly. The second group consists of methods that focus on retrieving the phase information by iteratively imposing constraints on the captured hologram on both the sensor plane and on the sample plane. The final group of methods utilizes deep learning for the purpose of phase reconstruction.

3.1 Twin-image elimination methods

The twin-image elimination methods focus primarily on the recovery of the complex amplitude of sample interference $a_z(x, y)$ from either 2.9 or 2.11.

3.1.1 Fourier-quotient methods

For strictly planar samples a fairly straightforward solution is to find a filter in the frequency domain that tries to invert the convolution kernel of the propagation operation in 2.10.

Such a filter has a singularity in its transfer function at each zero of the convolution kernel and methods using this approach differ in the way that they deal with these singularities. Nugent proposed adding a positive constant to the denominator of said transfer function [9]. There are other two algorithms that expand the filter's transfer function into a series and only consider the first term [6, 10].

The main disadvantage of the previous methods is that they only work for samples with real interference a_z, which is not necessarily the case on the provided platform. This problem is fixed in the method proposed by Maleki and Devaney [11], which also solves the singularity problem by setting the transfer function to 1 whenever a singularity is approached.

While all of the algorithms mentioned in this section can be easily implemented for real-time use, they only work for planar objects and return completely wrong results if z is guessed with accuracy worse than 5 %.

3.1.2 Iterative elimination methods

Iterative approaches to twin-image elimination utilize masking to separate the twin-image from the real image. Usually, these methods are only suitable for planar samples [12, 13] and work by alternating between the planes where the twin-image is located and the plane of the real image at the sample plane, while imposing spatial constraints by masking.

The algorithm proposed by Denis et. al. [14], which also belongs to this group, allows twin-image elimination from samples in a volume. This is done by first sampling the volume to as many sample planes as needed and separates the twin-images from real images in each one by masking. The complex amplitude of the twin-images is calculated and subtracted from the hologram for each plane in each iteration. The mask can be updated at the start of each iteration.

The methods mentioned in this section all could be implemented to use warm starting, where they would use the resulting guess of one run of their procedure as the initial guess of the second run so that fewer iterations are needed in this second run. This could potentially speed up the algorithm for all runs after the first one.

The iterative elimination methods seem promising but proper masking of twin-images in reconstructions as densely populated as the ones gathered from the provided platform could be very inaccurate.

3.2 Iterative phase reconstruction methods

A considerable number of iterative phase retrieval algorithms is derived from the Gerchberg-Saxton error reduction algorithm [15, 16]. These procedures are then sometimes referred to as the GS-based algorithms. This section will mainly focus on such methods, but a really thorough description of others that would fit into this section can be found in [17].

In the original procedure, two different intensity recordings are required. One of these is in the sample plane and the other in the sensor plane which corresponds to the hologram. The algorithm then randomly guesses what the phase information will be and enforces it to the sensor plane modulus measurements. In each iteration, the algorithm then applies four different operations:

1. Backpropagate to the sample plane
2. Enforce sample plane modulus
3. Propagate to the sensor plane
4. Enforce sensor plane modulus

Fienup [18] designed a method derived from the original error reduction algorithm, in which he replaced the second step by introducing the object support constraints, which works by assigning zero to places, where the observed objects are not believed to be. Later, Fienup generalized the procedure [19], so that other constraints can be enforced. The need for the predefined object support makes Fienup's algorithm difficult to use in the micromanipulation setting where the object support changes with time.

The original error reduction algorithm as well as the Fienup's algorithm were developed specifically for coherent diffraction imaging, where the sensor plane is the Fourier transform of the sample plane. Latychevskaia and Fink [20] changed the procedure to work in digital lensless in-line holography. The change was already included in the iteration description above.

Other GS-based algorithms for digital holography were developed in recent years, with their main differences stemming from the constraints that they use in the sample plane. Typical examples include positive absorption constraint [20], sparsity constraints [21] and object support constraints [22].

A GS-based algorithm could definitely be used for the provided micromanipulation platform depending on the constraint used.

Regularized inversion [23, 24, 25, 26, 27] is another way to approach phase reconstruction which can be analogous to the GS-based algorithms. The focus is shifted from reconstructing the phase information at the sensor plane to reconstructing the disturbance caused by the observed objects at the sample plane. It works by comparing a current disturbance guess by pushing it as a parameter for the direct model for hologram formation and calculating the optimal scaling factor for the model by comparing the model to the actual hologram. Depending on the chosen optimization strategy, the algorithm then can take a similar form to the alternating projections structure seen in the GS-based algorithms [23]. It is also possible to apply constraints at the sample plane just like with Fienup's algorithm.

The regularized inversion methodology [27] can be adjusted to work with multiple sample planes.

Since the regularized inversion algorithms estimate the disturbance at the sample plane, the hologram reconstruction is achieved without the need for additional backpropagation after the algorithm has ended. This can lead to a significant reduction in computation times when compared with GS-based algorithms.

Warm-starting can be used in the implementation of each of the algorithms mentioned in this section by carrying over the last estimate of one run of the algorithm to be used as the first estimate in the following run.

3.3 Deep learning

A fairly obvious approach to the problem of phase reconstruction would be using deep learning. Several authors [28, 29, 30] have already tested this approach with success using the convolutional neural network (CNN) architecture. However, for use in the scope of this thesis, CNNs have a major disadvantage: They are quite computationally demanding and therefore might be slow compared to some of the iterative algorithms discussed above, especially if the iterative algorithms utilized warm starting.

3.4 Conclusion

As was shown in this chapter, a plethora of methods exist for phase reconstruction in in-line digital holography, with many not even discussed in this thesis. Most of the ones described in this chapter can also be suitable for use with the provided micromanipulation platform such as the iterative twin-image elimination algorithm for samples in a volume [14], a GS-based algorithm with properly chosen constraints [16] or the regularized inversion approach [23].

The problem with these methods stems from their speed as each one requires two computationally demanding propagation operations per sample plane in each iteration. However, if warm starting is utilized it can lower the number of iterations needed after the first run. Additionally, if implemented on a GPU such as the one used in the Jetson AGX Xavier included in the provided platform, the algorithms might speed up enough to run in real-time.

It must also be noted that in the literature only sparsely populated samples were used, usually with complete knowledge about the support of the observed objects and rarely any overlap. It is possible that the results achieved by any of the mentioned methods on holograms recorded from the provided platform will not be comparable to the results achieved in the literature.

For this thesis, a regularized inversion approach will be used, thanks to the speed up gained by omitting the final backpropagation operation required by the GS-based algorithms. The implementation will then utilize the positivity constraint and the sparsity constraint.

Chapter 4
Selected method

In this chapter, the chosen phase reconstruction method [23] will be described in greater detail. The direct model for hologram formation will be explained, from which a cost function will be derived. Possible constraints will be introduced later along with corresponding changes to the cost function. Two different algorithms for minimization of the designed cost function will be then described, the latter of which will be used for implementation. Finally, warm-starting is briefly introduced and its implementation with the selected method is explained.

It is important to note that in this chapter, a vector notation will be used that is different from the notation used in the previous chapters. To easily differentiate between the two notations, a vector $\boldsymbol{p} = (x, y)$ is defined as the column vector of all possible 2D spatial coordinates. As an example, the column vector $\boldsymbol{a}_z$ can be then written as

$$\boldsymbol{a}_z = a_z(\boldsymbol{p}). \tag{4.1}$$

Furthermore, the output of the expression used for propagation and back-propagation operations, $\texttt{prop}(h(f_x, f_y), a(x, y))$, will retain the structure of the second input parameter. If it is a column vector, the output is also a column vector.

4.1 Cost function

In the chosen method, a direct model for hologram formation is given as:

$$\boldsymbol{I}_H = c\boldsymbol{m}(\boldsymbol{o}) + \boldsymbol{\eta}, \quad \boldsymbol{m}(\boldsymbol{o}) = |\mathbf{1} - \texttt{prop}(H_z, \boldsymbol{o})|^2, \tag{4.2}$$

where scalar c is the scaling factor, $\boldsymbol{m}(\boldsymbol{o})$ is the hologram formation model and vector $\boldsymbol{o}$ is the estimated complex disturbance caused by the observed objects at the sample plane. Vector $\boldsymbol{\eta}$ corresponds to the measurement noise. Like in the earlier sections, $\boldsymbol{I}_H$ represents the captured hologram and H_z is a propagation kernel.

From this hologram formation model, it is easy to derive a cost function that checks the quality of the model. That is done with squared l_2 norm,

$$\mathcal{J}_{fid}(c, \boldsymbol{o}, \boldsymbol{I}_H) = ||c\boldsymbol{m}(\boldsymbol{o}) - \boldsymbol{I}_H||^2 = (c\boldsymbol{m}(\boldsymbol{o}) - \boldsymbol{I}_H)^T(c\boldsymbol{m}(\boldsymbol{o}) - \boldsymbol{I}_H), \quad (4.3)$$

where $\boldsymbol{I}_H$ is the captured hologram. In the original article, the l_2 norm is weighted by a positive definite matrix $\boldsymbol{W}$ which allows taking into account the quality of captured data at individual pixels. In this thesis, said matrix $\boldsymbol{W}$ will be considered equal to identity and will therefore be omitted from all equations. The presented l_2 norm will be referred to as the data-fidelity term.

From (4.3) it is simple to get a minimization problem: To find the optimal solution, one must find the best values of both c^* and $\boldsymbol{o}^*$. The optimal scaling factor can be found in closed form from (4.3):

$$c^*(\boldsymbol{o}) = \frac{\boldsymbol{m}(\boldsymbol{o})^T \boldsymbol{I}_H}{\boldsymbol{m}(\boldsymbol{o})^T \boldsymbol{m}(\boldsymbol{o})}, \quad (4.4)$$

Now the minimization problem can be solved with respect to the estimated complex amplitude at the sample plane:

$$\boldsymbol{o}^* = \arg\min_{\boldsymbol{o}} \quad \mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}), \boldsymbol{o}, \boldsymbol{I}_H). \quad (4.5)$$

The data fidelity term is, however, not enough to properly reconstruct the phase, as a plethora of estimated complex amplitudes could fit a noisy hologram without actually being satisfactory. For this reason, it is possible to include some prior knowledge about the observed objects in the form of constraints. Adding constraints to the cost function is done simply by adding a new regularization term to the already established data fidelity. The minimization problem then has the form of

$$\boldsymbol{o}^* = \arg\min_{\boldsymbol{o}} \quad \mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}), \boldsymbol{o}, \boldsymbol{I}_H) + \mathcal{J}_{\text{reg}}(\boldsymbol{o}, \theta), \quad (4.6)$$

where θ represents the constraint parameters.

As for the constraints used, the most obvious choices are the positivity constraint and the sparsity constraint.

The positivity constraint is very simple and works on the idea, that since the observed objects do not emit their own light, so their absorption has to be positive. That is all elements of the estimated matrix $\boldsymbol{o}$ need to have a positive real value.

An additional constraint used in this thesis is the sparsity constraint. Phase reconstruction under the sparsity constraint favours pixels with zero value. The cost portion created by the sparsity constraint takes the form of the term

$$\mathcal{J}_{l_1}(\boldsymbol{o}, \mu) = \mu||\boldsymbol{o}||_1, \tag{4.7}$$

where $||\boldsymbol{o}||_1$ is the l_1 norm of $\boldsymbol{o}$ and μ is the sparsity hyperparameter. Actually, the sparsity constraint corresponds better to minimizing the l_0, or the sum of non-zero elements of $\boldsymbol{o}$, however, the l_1 norm is easier to minimize and proves to be sufficient for enforcing sparsity.

With both the constraints defined, the regularization term can now be written as

$$J_{\text{reg}} = \begin{cases} J_{l_1}, & \text{Re}\{\boldsymbol{o}\} \geq 0 \\ \infty, & \text{otherwise} \end{cases}. \tag{4.8}$$

With this term defined, the minimization problem can be rewritten to the form

$$\boldsymbol{o}^* = \underset{\text{Re}\{\boldsymbol{o}\}\geq 0}{\arg\min} \quad ||c\boldsymbol{m}(\boldsymbol{o}) - \boldsymbol{I}_H||_2^2 + \mu||\boldsymbol{o}||_1. \tag{4.9}$$

4.2 Iterative Shrinkage-Thresholding Algorithm

Now the next problem becomes the question of actually finding the optimal estimated disturbance by observed objects at the sample plane, $\boldsymbol{o}^*$. To solve this minimization problem, a proximal gradient method [31] called Iterative Shrinkage-Thresholding Algorithm (ISTA) can be used.

In the ISTA algorithm, the difficult direct minimization of the cost function is circumvented by instead iteratively minimizing a local majorant approximation of the cost function. In the case of the minimization problem (4.6), the approximation takes form of a proximal operator of the sparsity constraint term $J_{l_1}(\boldsymbol{o}, \mu)$ which can be written as the mapping [32]

$$\boldsymbol{x} \mapsto \underset{\text{Re}\{\boldsymbol{o}\}\geq 0}{\arg\min} \quad \frac{1}{2t}||\boldsymbol{o} - \boldsymbol{x}||_2^2 + J_{l_1}(\boldsymbol{o}, \mu), \tag{4.10}$$

where t is the constant step length, and $\boldsymbol{x}$ is the expression

$$\boldsymbol{x} = \boldsymbol{o}^{(i)} - t\nabla\mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}^{(i)}), \boldsymbol{o}^{(i)}, \boldsymbol{I}_H), \tag{4.11}$$

where $\boldsymbol{o}^{(i)}$ represents to current best estimate of $\boldsymbol{o}$. The parameter t is then the constant gradient descent step length. If this parameter is set to a low enough value, the algorithm can reach convergence.

Calculating the minimal value of the proximal operator gives the value of the next best estimate, $\boldsymbol{o}^{(i+1)}$.

It is useful to notice, that this proximal operator is separable, or that it is minimal only if its value at each pixel is minimal as well. The mapping (4.10) can be then rewritten to the form

$$\boldsymbol{x} \mapsto \underset{\mathrm{Re}\{\boldsymbol{o}\}\geq 0}{\arg\min} \sum_q \frac{1}{2t}\left[(o_q - x_q)^2 + \mu |o_q|\right], \tag{4.12}$$

where o_q and x_q are values of $\boldsymbol{o}$ and $\boldsymbol{x}$, respectively, at pixel pixel coordinates (x_q, y_q).

Minimum of the summed expression in (4.12) is for real-valued o_q found easily with the shrinkage operator:

$$o_q^{(i+1)} = \mathcal{T}_{\mu t}(x_q) = \mathrm{sgn}(x_q)\mathrm{max}(0, |x_q| - \mu t). \tag{4.13}$$

This operator can be applied to (4.10) to give the following ISTA iteration:

$$\boldsymbol{o}^{(i+1)} = \mathcal{T}_{\mu t}\left(\boldsymbol{o}^{(i)} - t\nabla \mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}^{(i)}), \boldsymbol{o}^{(i)}, \boldsymbol{I}_H)\right). \tag{4.14}$$

Now, the gradient of $\mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}), \boldsymbol{o}, \boldsymbol{I}_H)$ can be calculated as

$$\begin{aligned}\nabla \mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}), \boldsymbol{o}, \boldsymbol{I}_H) &= \nabla || c^*(\boldsymbol{o})|1 - \texttt{prop}(H_z, \boldsymbol{o})|^2 - \boldsymbol{I}_H||_2^2 = \\ &= 2tc^*(\boldsymbol{o})\texttt{prop}\left(H_z^T, \left((\mathbf{1} - \texttt{prop}(H_z, \boldsymbol{o}))^T (c^*(\boldsymbol{o})\boldsymbol{m}(\boldsymbol{o}) - \boldsymbol{I}_H)\right)\right),\end{aligned} \tag{4.15}$$

where $H_z^T = H_{-z}$ as can be seen from (2.1). This solved gradient can be then used to get a complete iteration of the ISTA method.

Since the shrinkage operator does not bind the estimate only to positive values, the real part of the estimate has to be bounded to non-negative numbers after applying the shrinkage operation. This can be done as:

$$\mathrm{Re}\{o_q^{(i+1)}\} = \mathcal{B}_{\text{pos}}(\mathrm{Re}\{o_q^{(i+1)}\}) = \max(0, \mathrm{Re}\{o_q^{(i+1)}\}), \tag{4.16}$$

where $\mathrm{Re}\{o_q\}$ is the real part of the complex number o_q.

One problem still has to be addressed: The shrinkage operator from (4.13) only works for real-valued arguments. This is solved easily by separating the real and imaginary parts of its input matrix and running the shrinkage operator on both of these parts separately. The results are then combined back together to form the new complex estimate. In the following sections, it will be assumed that the shrinkage operator undergoes these steps automatically for complex inputs.

4.3 Fast Iterative Shrinkage-Thresholding Algorithm

In this thesis, the minimization algorithm used is actually the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [33]. Instead of calculating the new estimate $\boldsymbol{o}^{(i+1)}$ from just the value of $\boldsymbol{o}^{(i)}$ as was done with ISTA, the new estimate is actually calculated from a specific linear combination of $\boldsymbol{o}^{(i)}$ and $\boldsymbol{o}^{(i-i)}$, corresponding to the accelerated gradient descent method proposed by Nesterov [34]. This is achieved by adding two additional steps to each iteration.

At the start of the algorithm, the original estimate is assigned to $\boldsymbol{o}^{(0)}$, as well as to the initial intermediate value of $\boldsymbol{o}^{(1/2)}$. The most straightforward initial estimate for these two matrices is the recorded hologram for which the phase is to be reconstructed. A variable FISTA scalar factor $s_1 = 1$ is also assigned at the start. Every iteration of this algorithm then has these four steps:

$$\boldsymbol{o}^{(i+1)} = \mathcal{T}_{\mu t}\left(\boldsymbol{o}^{(i+1/2)} - t\nabla\mathcal{J}_{\text{fid}}(c^*(\boldsymbol{o}^{(i+1/2)}), \boldsymbol{o}^{(i+1/2)}, \boldsymbol{I}_H)\right), \tag{4.17}$$

$$\text{Re}\{\boldsymbol{o}^{(i+1)}\} = \mathcal{B}_{\text{pos}}(\text{Re}\{\boldsymbol{o}^{(i+1)}\}) \tag{4.18}$$

$$s_{i+1} = \frac{1+\sqrt{1+4s_i^2}}{2}, \tag{4.19}$$

$$\boldsymbol{o}^{(i+3/2)} = \boldsymbol{o}^{(i+1)} + \left(\frac{s_i - 1}{s_{i+1}}\right)(\boldsymbol{o}^{(i+1)} - \boldsymbol{o}^{(i)}). \tag{4.20}$$

When the algorithm converges, the estimated hologram reconstruction can be calculated by adding the first value of the backpropagation kernel to each value of the final FISTA estimate and calculating the absolute value of the result.

The FISTA method has a major advantage in comparison to the original ISTA method: Whereas the ISTA method has a worst case complexity result of $O(1/k)$ [33], the FISTA method has a worst case complexity result of only $O(1/k^2)$.

4.4 Warm-Starting

During the control process, the platform will record subsequent holograms which should have their phase reconstructed in real-time, if possible. And while the FISTA method has a fast rate of convergence, small improvements can still be made to achieve faster execution times in the implementation.

It can be assumed that between recordings the observed particles will not change their position by a significant amount. This assumption allows the use of information gathered from one run of the algorithm in the subsequent run. In other words, this assumptions allows the use of warm-starting.

Warm-starting can be implemented easily by using the final estimate and the final intermediate estimate from one run as the initial corresponding estimates in the next run. This can be done by reconstructing each hologram after the first one that passes through the algorithm.

Chapter 5

Implementation

In this chapter, the different implementations of the selected method are described in detail. As was mentioned at the start of this thesis, the selected method is to be implemented three times: First, a prototype of the selected method is to be implemented in Matlab to test its functionality and tune its parameters, without any regard for real-time use. Then the method is to be reimplemented in CUDA and optimized for real-time use. Finally, the algorithm is to be integrated into the already existing code running on the micromanipulation platform, in which it is supposed to process all captured holograms and portray them on a display connected to the platform.

Since the selected phase reconstruction algorithm will remain mostly unchanged between the different implementations, its implementation will be briefly described first. This will be followed by descriptions of the three different implementations.

The Matlab and CUDA implementations can be found in an online repository at: `https://github.com/vikroe/phasereconstruction` along with documentation that explains, how each of the implementations can be tested.

The integrated algorithm is available on a repository branch at: `https://github.com/aa4cc/twinbeam-setup/tree/phasereconstruction`. It is possible that this branch will be updated even after this thesis is finished.

5.1 Phase Reconstruction Algorithm

The proposed implementation of FISTA for phase reconstruction is straightforward and mostly follows the process described in the previous chapter. The implementation has one major difference from the previous chapter, in which the direct model is calculated as:

$$\boldsymbol{m}(\boldsymbol{o}) = |\mathbf{1} + \texttt{prop}(H_z, \boldsymbol{o})|^2. \tag{5.1}$$

This means that in the implementation, the estimated disturbance has a negative real part. The positivity constraint, therefore, becomes the negativity constraint.

The input parameters of the algorithm are:

- Propagation kernel
- Hologram
- Sparsity hyperparameter
- Gradient descent step length
- Maximum number of iterations

During initialization, the initial estimates $\boldsymbol{o}^{(0)}$ and $\boldsymbol{o}^{(1/2)}$ are set equal to the input hologram. Then the backpropagation kernel is calculated as the complex conjugate of the propagation kernel. Finally, the variable scalar factor used for the acceleration steps of FISTA is set to its initial value of $s = 1$. After this the main loop of FISTA starts.

The algorithm itself then does not deviate much from the description of FISTA given by (4.17)-(4.20). Actually, the only major difference is that cost at each iteration is calculated after calculating the optimal scaling factor $c^*(\boldsymbol{o}^{(i+1/2)})$. This cost is usually then printed to the console and can be used as a useful metric for evaluating how good are the current estimates at each iteration.

When the maximum number of iterations is reached, the algorithm terminates and the final estimate is post processed so that the desired hologram reconstruction can be obtained. This post-processing is done as follows:

$$\boldsymbol{A}_z = \boldsymbol{o}^{(\max)} + H_{-z}(0,0), \tag{5.2}$$

where $H_{-z}(0,0)$ is the constant value of the reference planar wave at the sample plane. The result $\boldsymbol{A}_z$ is then an estimate of the complex planar wave at the sample plane. In the future sections, the result of this algorithm will be considered to be the absolute value of $\boldsymbol{A}_z$ as it should provide a good enough image of the sample plane for displaying. The phase of $\boldsymbol{A}_z$ can be discarded as it will not be required anymore.

5.2 Matlab Prototype

The Matlab prototype operates on a sequence of a few elementary steps: First, the parameters for the FISTA algorithm, as well as for the calculation of a desirable propagation kernel are loaded into the Matlab workspace from a configuration file with their values. The propagation kernel is calculated immediately after that.

The configuration file also selects whether the input hologram is to be simulated or if it is supposed to be loaded in from an image file that contains hologram recorded from the micromanipulation platform.

If the input hologram comes from an actual recording, it is first normalized so that its average value is equal to one, so that the assumption in (2.7) is kept. That is done by calculating its original average and then dividing each pixel of the recorded hologram by said average. The average is stored for later use.

The hologram is then run passed to FISTA and the result is pictured to provide visual proof of the reconstruction effectiveness. In this implementation, the phase information is also pictured but it serves no purpose. If the hologram was originally scaled by its average value, each element of the calculated modulus is multiplied by this saved average value so that the modulus matches the original hologram in intensity. The results are compared to an ordinary backpropagation of the hologram. Cost calculated at each iteration of FISTA is printed to console during the run of the algorithm.

In the configuration file, it can be specified if the pictured results are to be saved as images.

5.2.1 Hologram simulation

As was already mentioned, the input data for the prototype can come in the form of a simulated hologram. For this reason, a method of generating reasonable holograms has to be designed.

To simulate a hologram, first, a disturbance caused by the simulated observed objects is generated and stored for later use. This true disturbance is then propagated to the simulated sensor plane and each pixel of the propagation result is summed with 1. Finally, according to (2.8), the absolute value of this generated complex amplitude is calculated and squared element-wise. The result of this is a simulated hologram. Of course, real recorded holograms are affected by additional noise, which can be simulated by adding Gaussian noise with zero mean and a small standard deviation to the hologram.

The generated disturbance should have a complex 0 at most pixels with non-positive real values at the places where simulated objects are supposed to be situated on the sample plane. At the same places, the imaginary values are not restricted.

One advantage in using the simulated holograms is that the actual disturbance at sample plane is known and can be used to measure how effective the algorithm is at reconstructing the phase information. One common metric that can be used to quantify this effectiveness is the *signal-to-noise ratio* (SNR), which is calculated by the equation:

$$\mathrm{SNR} = 20\log\left(\frac{||a_z||_2}{||o - a_z||_2}\right), \tag{5.3}$$

where a_z is the ground truth and o is the estimate of the complex disturbance. $||a_z||_2$ is of course the l_2 norm of a_z.

SNR is automatically calculated if the input hologram is a simulation.

5.3 CUDA Implementation

The CUDA implementation deviates from the prototype in a major way: A sequence of holograms can be uploaded from a video recording and sequentially processed by FISTA. Since it is assumed at this point that the algorithm works well, only actual recorded holograms either in picture or video format are used with simulations omitted in this implementation.

In this implementation, first, the parameters are uploaded into memory from a `JSON` configuration file. The parameters that can be specified in this file are, of course, the necessary parameters for generating a propagation kernel listed in section 2.2 and the parameters necessary for running FISTA listed in 5.1. Warm-starting can be enabled in the configuration as well. A different number of iterations can be specified for warm-started reconstructions than for the first reconstruction. Saving of results received from FISTA can also be enabled. Another important parameter is the input file type, which can be either set to `AVI` which signifies that the input is a video recording captured on the provided platform, or it can be set to `PNG` which signifies that the input is, just like in the prototype, only a single hologram.

If the file type parameter is set to `PNG` the algorithm continues more or less in the same fashion as the Matlab prototype: The implementation is initialized by loading parameters from the configuration file. The input hologram is loaded during initialization and is passed along other parameters to FISTA. Both the propagation and the backpropagation kernels are calculated during the initialization of FISTA. After the algorithm ends, the result is displayed in a separate window. If enabled in the configuration file, the displayed result is saved to a new image file.

On the other hand, if the file type is `AVI` the implementation utilizes multi-threading to optimize the phase reconstruction process so that each frame of the recording can be processed by FISTA as fast as possible. Three threads were overall implemented:

- Frame thread
- Image processing thread
- Display thread

The *frame thread* holds the pipeline for uploading individual frames from the video file. A frame is uploaded and then stored into a global array. The thread then waits for a signal from the *image processing thread* before uploading another frame.

The *image processing thread* starts after the first frame is uploaded by the *frame thread*. It copies said frame into its local array and sends a signal to the *frame thread* so that it can upload a new frame. FISTA is then run on the stored frame.

If warm-starting was enabled in the configuration file, FISTA can use the final estimates of the disturbance at the sample plane calculated for one frame as the new initial estimates for the next frame. This is, of course, possible only after the first frame was already processed. Warm-started runs of FISTA should have a greatly reduced cost of initial estimates compared to runs that are not warm-started. For this reason, fewer iterations should be needed in warm-started runs to reach satisfactory results. The configuration file thus also allows setting a different number of iterations for processing the first frame of the recording and for processing all the other frames.

After the algorithm finishes processing the current frame, the *image processing thread* copies the result to a global array and sends a signal to the *display thread*. The *display thread* then copies the result to its local memory and displays it in a separate window. In the configuration file, saving of the results to a new video file can be enabled.

5.3.1 CUDA

As was stated earlier in this thesis, this implementation was programmed in C++ utilizing CUDA. The main advantage of using CUDA for this implementation is that it allows sections of the implemented code to run directly on a CUDA-enabled GPU. This is greatly advantageous for such sections that can be easily parallelizable, which most of the image processing operations used in FISTA are.

5.3.2 Used software

Three external, publicly available, libraries were needed for finishing this implementation excluding CUDA. Those are OpenCV which was used for uploading holograms from both possible input file types and for displaying the results received from FISTA as well as for saving those results.

Another library used was the C++ `JSON` parser created by Niels Lohmann.

The last external library used was cuFFT, which provides parallelized implementations of the spatial Fourier transform and its inverse. Both of these operations are needed each time an image is to be backpropagated or propagated.

5.4 Integration

The code into which FISTA for phase reconstruction needs to be integrated has a similar structure to the CUDA implementation designed in the earlier section for processing video recordings from the provided platform. Most importantly it also shares the same architecture of having a *frame thread* that collects the input holograms — this time directly from the image sensor — which are then sent over to an *image processing thread* where the holograms are processed and the results are, finally, sent over to a *display thread* from where they are displayed on a monitor connected to the micromanipulation platform and saved if so desired.

Normally, the *image processing thread* is used in this code to backpropagate captured holograms on each of the two relevant colour channels of the image sensor. These backpropagations are then used to estimate the 3D positions of polystyrene particles and for their subsequent tracking. The backpropagations of holograms from one of the channels can also be displayed on a monitor connected to the platform or saved in a video format.

The integration kept the backpropagations of both channels for position estimation and tracking of the particles but added a step in the *image processing thread* in which the hologram recorded on the green channel is also passed to FISTA that was set to the same parameters as it was in the CUDA implementation. Instead of displaying the simple backpropagation, the result of FISTA are sent to the *display thread* instead. All other functionality of the original code is otherwise retained.

Chapter 6
Experiments

In this chapter, first, the selected method will be tested against a simulated hologram generated by the process explained in section 5.2.1. After that, the FISTA parameters will be tuned by running the algorithm on an actual hologram captured from the provided platform until satisfactory results are gathered. Then the real-time CUDA implementation will be tested on video recordings captured from the provided platform and the speed per video frame of the algorithm will be measured. Additionally, experiments on the video recordings are supposed to show how effective warm-starting is with the selected method. Finally, the ability of the algorithm to run in real-time while integrated into the micromanipulation code on the platform will be tested at the end.

6.1 Simulated Hologram

Before experimenting with the algorithm on holograms captured on the micromanipulation platform, it would be useful to find a way of testing whether and how well FISTA works for reconstructing phase on holograms at all. For this reason, a hologram can be simulated by the process described in section 5.2.1.

The parameters needed for generating a propagation kernel can be set to any values in this experiment since the propagation kernel will be used for both simulating the hologram and reconstructing its phase. The parameters were therefore selected with values similar to the values which will be probably required for properly reconstructing phase in holograms captured on the platform. Such values can be found in table 6.1.

Parameter	Value
λ	515 nm
dx	1.55 µm
n	1.45
z	2.4 mm

Table 6.1 Table of parameters selected for generating the propagation kernel used for generating the simulated hologram.

The disturbance on the sample plane, or the ground truth, was then generated from 3 filled in circles with a diameter of 32 pixels which comes up to a simulated diameter of almost 50 µm if the pixel size dx in table 6.1 is taken into account. All circles have their real part equal to 0.5 and imaginary part somewhere on the interval [-0.2; 0.2].

Finally, Gaussian noise can be added to the generated hologram to show how FISTA deals with noisy holograms. The value used for standard deviation in the experiment is $\sigma = 0.05$. The algorithm will then run 5 times with the added Gaussian noise for 5 iterations and the cost and SNR will be recorded at each iteration. The same will be done for a single run without Gaussian noise. SNR is calculated according to (5.3).

The calculated values of SNR will be in both cases compared to the SNR of an appropriate backpropagation of the simulated hologram. To calculate this value, the result of the backpropagation has to be normalized first by having the first value in the backpropagation kernel subtracted from each value to suppress the effect of the reference planar wave.

The sparsity hyperparameter was set to the value of $\mu = 0.25$ and the constant gradient step length was set to the value of $t = 0.25$.

The resulting images of the last iteration from one run of each experiment compared to the original hologram, simple backpropagation, and the ground truth can be seen in figure 6.1. The calculated costs and appropriate SNRs can be found in table 6.2. For the runs with added noise, the resulting values are averaged across the 5 runs and their standard deviation is calculated as well. In the table, there is a row for iteration 0, which corresponds to the initial guesses. For the backpropagation, only the SNR was calculated.

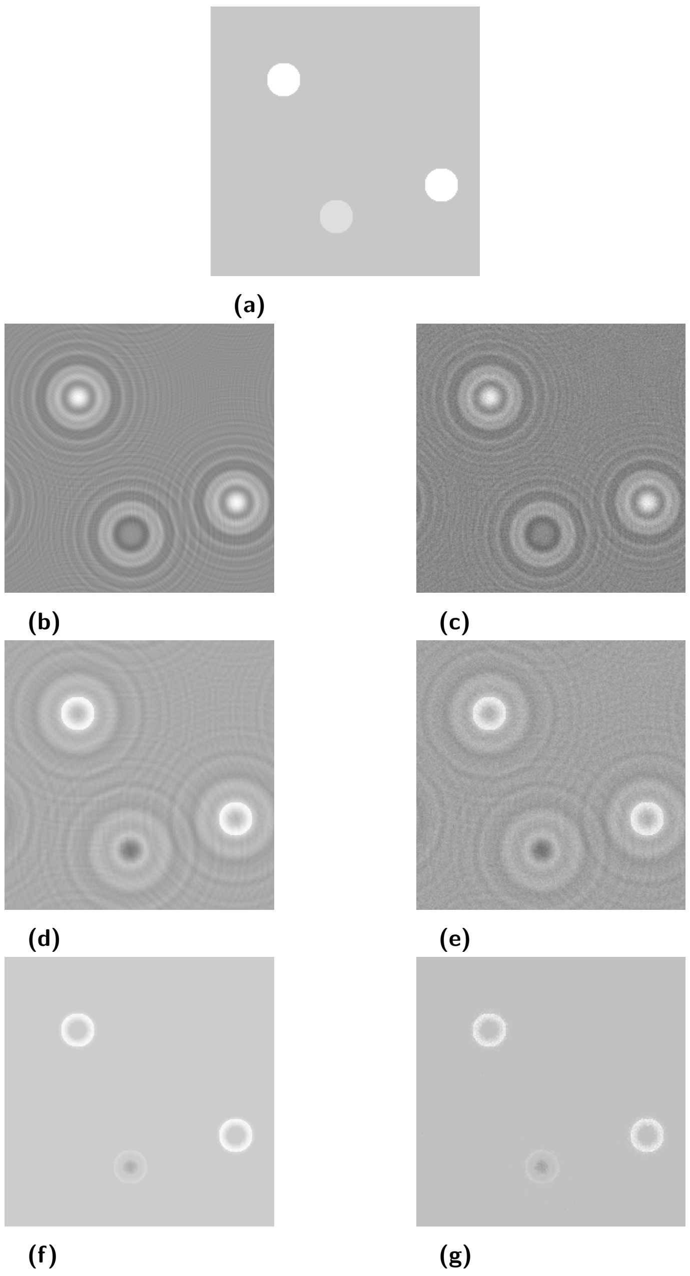


Figure 6.1 Results of the simulated hologram experiments. (a) Ground truth summed with the first element of the backpropagation kernel. (b)-(c) Generated hologram (b) without and (c) with the added noise. (d)-(e) Backpropagations of the holograms (d) without and (e) with the added noise. (f)-(g) Final results of FISTA on holograms (f) without and (g) with the added noise.

Iteration	Noise $\sigma = 0$		Noise $\sigma = 0.05$	
	Cost	**SNR**	**Cost**	**SNR**
0	17138	-20.642	17500 ± 5	-20.652 ± 0.001
1	442	-0.016	606 ± 3	-0.020 ± 0.001
2	223	0.541	372 ± 1	0.535 ± 0.002
3	215	0.813	365 ± 1	0.799 ± 0.004
4	211	0.977	360 ± 1	0.957 ± 0.005
5	212	1.061	361 ± 1	1.0411 ± 0.007
Backpropagation	-	-0.878	-	-1.728 ± 0.008

Table 6.2 Table of recorded costs and SNRs for the run of FISTA without added noise and 5 runs with the added Gaussian noise of $\sigma = 0.05$. The SNRs are compared to the corresponding values gathered from simple backpropagations.

As can be seen from both the figure 6.1 and table 6.2, FISTA does return clearer reconstructions than a simple backpropagation without the interference of twin-images. It also deals well with the Gaussian noise affecting the simulated hologram. The reconstructions are not perfect representations of ground truth but look better than the ones gathered using regular backpropagation.

6.2 Parameter Tuning

In the previous section, FISTA was shown to be able to estimate the complex amplitude at the sample plane of a simulated hologram. This section then serves to provide a detailed explanation of the process used to find the proper values for each of the required parameters. The parameters in question are then the parameters needed for calculation of the propagation kernel, described in section 2.2, and the parameters needed for properly running the FISTA algorithm.

The parameters will be selected in the following order:

1. Wavelength λ and pixel size dx
2. Refractive index n
3. Propagation distance z
4. Constant gradient step length t
5. Sparsity hyperparameter μ

6.2.1 Wavelength λ and pixel size dx

These two parameters are determined from the properties of the used micromanipulation platform, and their values were already mentioned in section 2.4. The pixel size is therefore equal to $dx = 1.55\,\mu\text{m}$ and the wavelength of the light source is then $\lambda = 515\,\text{nm}$, which corresponds to the wavelength of the green LED used in the platform. Since only holograms captured on the green channel are used for phase reconstruction, the red light source can be omitted in this thesis.

6.2.2 Refractive index n

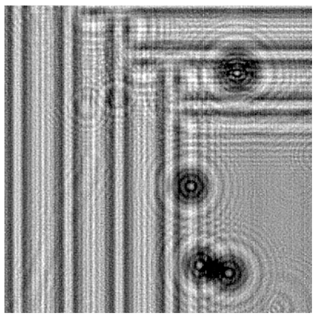
The propagation and backpropagation operations expect the medium between the sample plane and the sensor plane to be homogeneous. That is, however, not true, as was earlier stated in section 2.4 and the medium consists of layers of mediums with differing refractive indices. Nonetheless, this does not pose a significant problem, thanks to the fact that the refractive index only corresponds to the speed at which the planar complex wave produced by the light source propagates through the medium. Therefore, if, for example, a hologram is backpropagated to some axial distance using just one of the refractive indices, the resulting planar complex wave does not match the actual planar complex wave at that axial distance but rather a planar complex wave at a different axial distance shifted with respect to the real refractive indices of the different mediums between the two planes.

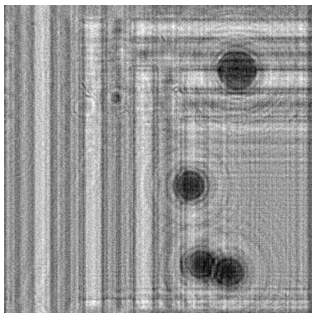
The choice of the refractive index used for backpropagation and propagation kernels is thus free and its error only results in a shift between the input axial distances and the real ones as long as the selected value is sensible $(n \geq 1)$. For all experiments the refractive index was selected to have a value of $n = 1.45$, which is close to the refractive indices of the mediums between the observed objects and the image sensor. The selected value should not result in a significant shift between the used and real axial distances.

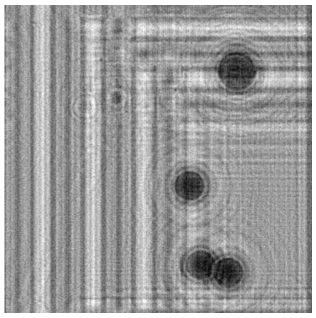
6.2.3 Propagation distance z

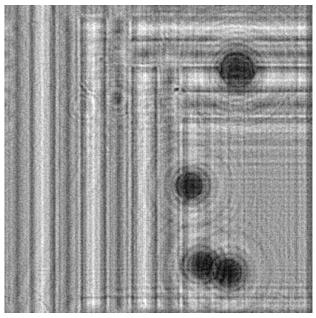
The distance used for propagation z and with a negative sign for backpropagation can be estimated to a satisfactory degree by just visually comparing results of backpropagation operations with different distances on the same hologram. In the case of the used platform, propagation distance will be tuned around the polystyrene particles, which should have a well-defined edge and a diameter of approximately 32 pixels. Since the electrodes are very close to the particles, a result with well-defined particles should have fairly well-defined electrodes as well.

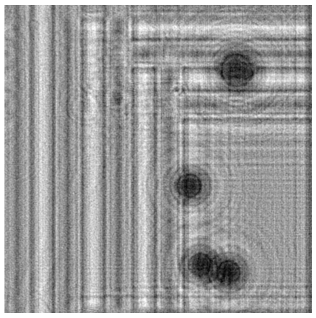
A small section of the hologram pictured in figure 2.2a along with five of its backpropagations at different distances can be seen in figure 6.2. From the figure, it can be seen that satisfactory results are gathered from backpropagations with propagation distances ranging from 2.3 mm to 2.5 mm.


(a) $z = 0\,\mathrm{mm}$


(b) $z = 2.2\,\mathrm{mm}$


(c) $z = 2.3\,\mathrm{mm}$


(d) $z = 2.4\,\mathrm{mm}$


(e) $z = 2.5\,\mathrm{mm}$

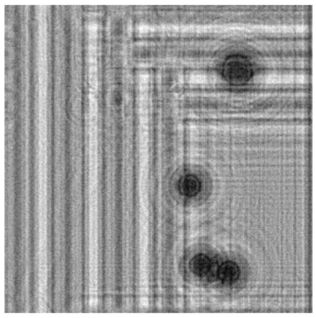

(f) $z = 2.6\,\mathrm{mm}$

Figure 6.2 Results on backpropagation with different propagation distance on a (a) section of a hologram recorded from the provided platform. The propagation distances are (b) 2.2 mm, (c) 2.3 mm, (d) 2.4 mm, (e) 2.5 mm and (f) 2.6 mm.

A surprising behaviour can be observed when running FISTA on recorded holograms with differing values of propagation distance: The absolute value of the final estimate returned from FISTA summed with the first element of the backpropagation kernel changes drastically with even small variations in propagation distance. To show this, sensible values for FISTA parameters have to be assumed as they are not tuned yet, so let $t = 0.2$ and $\mu = 0$, which means that the sparsity constraint is turned off for now. The number of iterations is set to 5. In figure 6.3, it is possible to see the absolute values of the complex planar wave at the sample plane estimated by FISTA from the same hologram section as before with just slightly differing propagation distances.

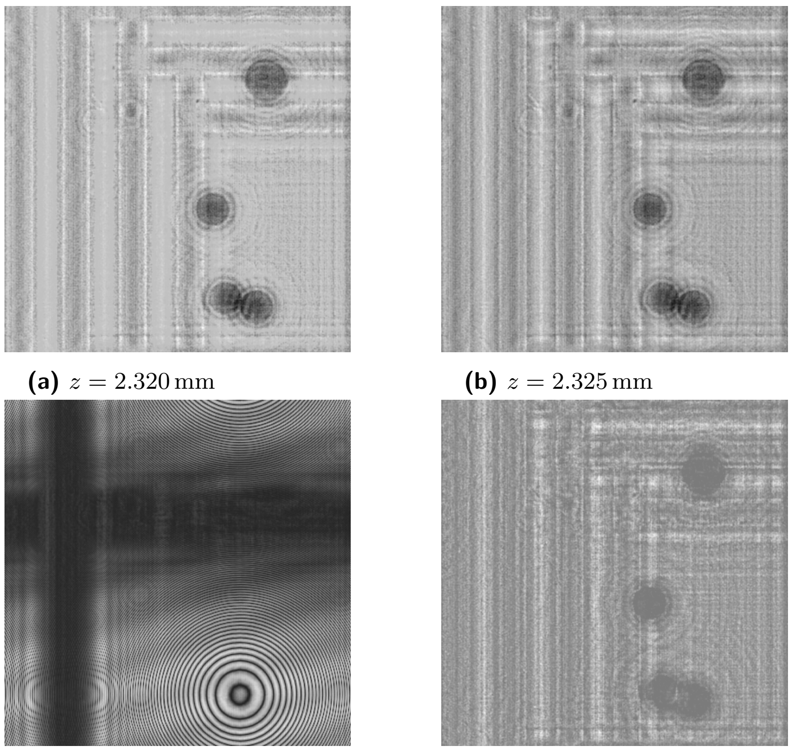

(a) $z = 2.320\,\mathrm{mm}$ **(b)** $z = 2.325\,\mathrm{mm}$ **(c)** $z = 2.339\,\mathrm{mm}$ **(d)** $z = 2.345\,\mathrm{mm}$

Figure 6.3 Resulting complex amplitudes at sample plane estimated by FISTA on a section of a hologram captured on the platform. For propagation distances (a)-(b) the estimation returns satisfactory results while for (c)-(d) the estimation returns bad reconstructions or it outright diverges.

Interestingly, it can also be shown that some periodicity occurs with respect to the propagation distance which leads to having reconstruction results that appear very similar if their associated propagation distance differs by a multiple of 0.027 mm. This behaviour is visualized in figure 6.4. This behaviour stems from the way that the propagation and backpropagation kernels are calculated in (2.1) and from the way by which the ratio of z/λ changes with changing propagation distance z. Just a small change in the propagation distance can lead to a drastic change in the phase of the whole kernel. This does not affect simple backpropagations of holograms in which the phase information is discarded. However, in phase reconstruction, it takes effect each time the new estimate is calculated and then finally when the final estimate is offset by the first element of the backpropagation kernel.

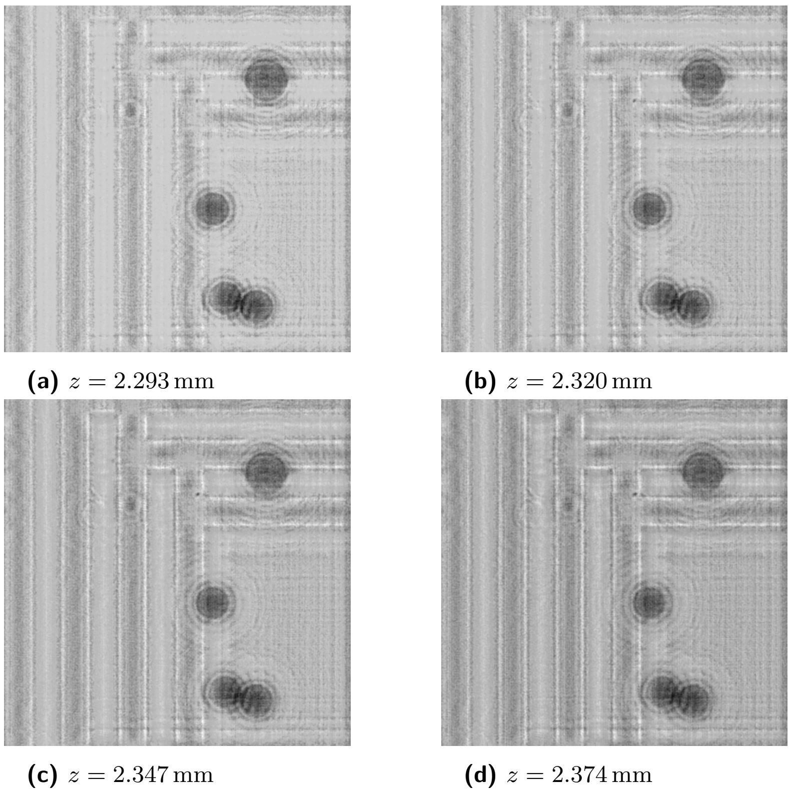

(a) $z = 2.293\,\mathrm{mm}$
(b) $z = 2.320\,\mathrm{mm}$
(c) $z = 2.347\,\mathrm{mm}$
(d) $z = 2.374\,\mathrm{mm}$

Figure 6.4 Resulting complex amplitudes at sample plane estimated by FISTA on a hologram section. The periodical behaviour when the propagation distance differs by a multiple of 0.027 mm can be observed.

From the multiple experiments done with the propagation distance, the chosen value for the propagation distance will be $z = 2.347\,\mathrm{mm}$, the effect of which on the estimated complex amplitude at sample plane can be seen in figure 6.4. The reason for this is that the resulting reconstruction has a large enough contrast between the particles and their surroundings while it also sufficiently reconstructs the electrodes.

6.2.4 Constant gradient step length t

The constant gradient step length is an important parameter for the speed at which FISTA converges. If it is set as too large, the algorithm overshoots and can't get to the minimum value of the cost function. If it, on the other hand, is set as too small the algorithm will converge too slowly.

Setting its value is rather simple: Run FISTA multiple times with $\mu = 0$ and other parameters set to the already found values and only change the value of t between runs. The algorithm should run for a sufficiently large number of iterations with 10 iterations being an adequate amount. Then record the calculated cost at each iteration for each run and compare the results. A graph showing the calculated cost at each iteration of FISTA for multiple runs with $t \in [0.1; 0.3]$ can be seen in figure 6.5. The actual values of the calculated costs at each iteration can then be seen in table 6.3.

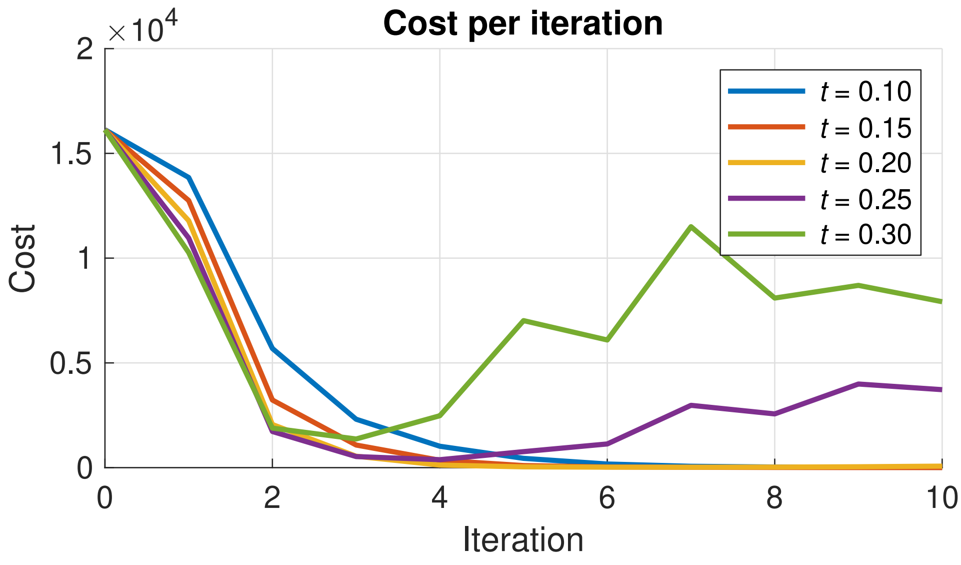


Figure 6.5 A graph showing the calculated cost at each iteration of multiple runs of FISTA with differing values of the constant gradient step length t.

t \Iteration	0	1	2	3	4	5	6	7
0.10	16129	13849	5688	2310	1021	443	173	64
0.15	16129	12752	3227	1079	352	99	33	17
0.20	16129	11788	2058	546	122	38	22	20
0.25	16129	10959	1725	529	380	764	1131	2974
0.30	16129	10265	1887	1368	2478	7017	6093	11501

Table 6.3 Table of calculated costs per iteration for multiple runs of FISTA with differing values of constant gradient step length. The last 3 iterations are cut as they will definitely not be used for actual reconstructions.

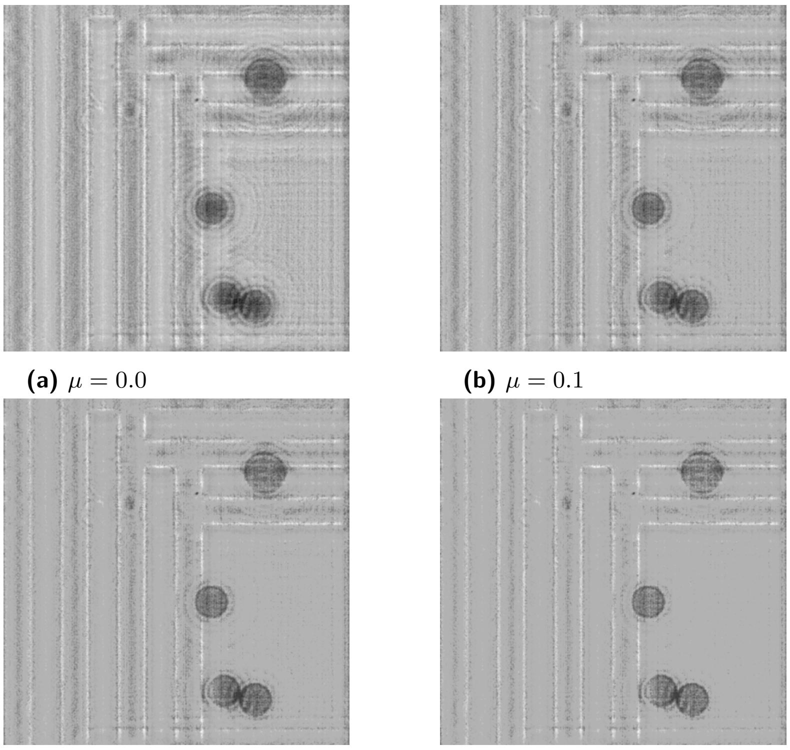
From both the table and the figure, it is fairly easy to select the step length value of $t = 0.2$. At 3 iterations it achieves very low cost and continues to converge. In real-time experiments, FISTA will usually be run to at most 3 iterations, which gets the algorithm sufficiently close to convergence in as few iterations as possible. Additionally, larger values tend to increase the value of the cost function after several iterations, which could have a negative impact on warm-started executions of the algorithm.

6.2.5 Sparsity hyperparameter μ

The final parameter set in this section will be the sparsity hyperparameter μ. When setting this parameter it is useful to recall the definition for the shrinkage operator from (4.13), in which each pixel is set to zero if its absolute value is smaller than the product μt. The hyperparameter is therefore set to such a value that the operator reduces noise and removes twin-images but does not significantly affect the shape of the actual observed objects.

With the other parameters already set, FISTA is run several times for 5 iterations with different values of $\mu \in [0.0; 0.3]$ on the same hologram section as before. The resulting absolute values of complex amplitudes at the sample plane can be seen on figure 6.6.

The chosen value of the hyperparameter will be $\mu = 0.3$ as it removes almost all of the twin-images from the reconstruction while not disrupting too much the geometry of electrodes and the polystyrene particles. The cost evolves the same way with enabled sparsity constraint as it did on figure 6.5, just with an offset caused by the regularization term.



(a) $\mu = 0.0$ **(b)** $\mu = 0.1$

(c) $\mu = 0.2$ **(d)** $\mu = 0.3$

Figure 6.6 Resulting complex amplitudes at sample plane estimated by FISTA on a hologram section from 4 runs with differing values of the sparsity hyperparameter.

6.2.6 Further tests

Now that all parameters have been set to good enough values, the results generated by the tuned algorithm can be compared to a regular optical microscope. Such comparisons can be seen in figure 6.7. From the images, it can be seen that the tuned algorithm fares well when compared to recordings from an optical microscope.

6.3 Real-time Experiment

Now that the algorithm parameters are properly tuned it is time to find out how well suited the algorithm is for real-time use. In this section, this will be tested on the CUDA implementation with the use of NVIDIA GTX 660 as

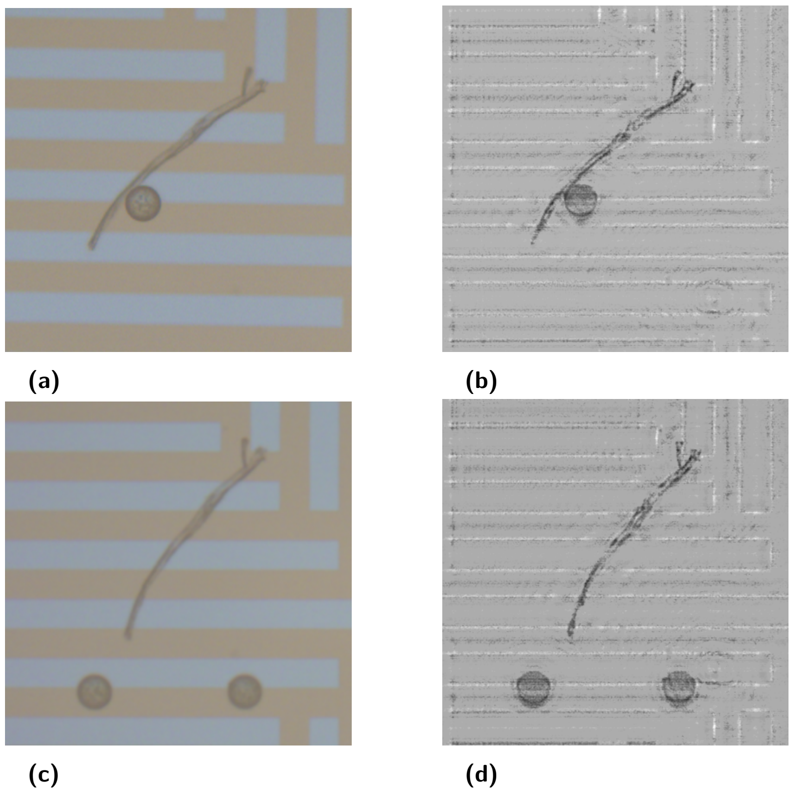


Figure 6.7 Comparisons between the images captured on an optical microscope and results generated by tuned FISTA.

the GPU.

In this experiment, two different video recordings of micromanipulation experiments were used to determine the quality of FISTA as a real-time algorithm for phase reconstruction[1]. One of these videos is in the same 4 *megapixel* (MP) resolution as the hologram found on figure 2.2a, specifically 2048×2048 pixels. The other video is in a reduced resolution of only 1024×1024, or 1 MP. The reduced resolution is achieved by taking adjacent squares of 2×2 pixels and taking their average as a new pixel value in their place. The tuned algorithm can be run on this reduced resolution without any problem, only the pixel size has to be doubled to $dx = 3.1\,\mu\text{m}$.

The first frame of either video will be always processed by FISTA running for 12 iterations since the processing of the first frame can be considered to

[1]The video recordings can be found on a Youtube playlist found at `https://youtube.com/playlist?list=PLMChF1jy7Ihrs6QmfWcMJlV2mtYQRiF9W`

be just another step in the initialization of this implementation.

The first experiment is rather straightforward and depends on running the algorithm on both of the two video recordings for a limited number of iterations for a specified number of frames. The execution time of the algorithm will be measured for each combination of input videos and number of iterations. The maximum number of iterations used in this experiment will be 3 since as can be seen from 6.3, this number of iterations is enough to get very close to convergence with the tuned parameters. The resulting execution times will be averaged for all frames and the standard deviation will be calculated as well.

The results of this experiment can be found in table 6.4. From the table, it is obvious that it is not viable to run FISTA for more than two iterations on a 4 MP video recording. However, from table 6.3 it is clear that running the algorithm for just one iteration will most probably not result in satisfactory reconstructions. Running the algorithm on the 1 MP recordings is suitable for all the tested numbers of iterations.

	1 MP	**4 MP**
#Iterations	t [ms]	t [ms]
1	19.2 ± 1.4	62.4 ± 1.6
2	27.0 ± 1.1	94.4 ± 0.7
3	35.8 ± 0.8	129.0 ± 2.3

Table 6.4 Calculated execution times for combinations of input videos of different resolutions (1 MP and 4 MP) with the maximum number of iterations of FISTA. Averaged over 450 frames of video. Standard deviation calculated for each of the results.

It would be preferable to reach execution times, at least, as low as 33 ms to be able to process 30 frames per second (FPS). That will be especially useful when integrating the algorithm into the existing code running on the Jetson AGX Xavier, which already uses the processing power of its GPU for estimating the positions of individual polystyrene particles. Integrating a slower algorithm could lead to a radical slowdown of the micromanipulation process which is not desirable. Finding a way of reaching satisfactory reconstructions with just 2 iterations of FISTA or even with only 1 iteration on the 1 MP recording would solve this potential problem. This can, of course, be done with warm-starting.

6.3.1 Warm-starting

When warm-starting is enabled, the algorithm uses the final estimate and final intermediate estimate calculated for one frame as the initial estimate and initial intermediate estimate, respectively, for the next frame.

To test how much better the reconstructions get with warm-starting, the algorithm was again run on both of the videos. The number of iterations was again set to 3. The algorithm ran once on either video recording with warm-starting and then once again without warm-starting. The calculated costs for frames 2 to 6 were recorded and averaged for each of the four runs and for each iteration as well as for the initial estimates. Standard deviation was, of course, calculated for each of the values. The results can be found in table 6.5.

Resolution	1 MP	1 MP	4 MP	4 MP
Warm-Start	NO	YES	NO	YES
Iter. 0	100605 ± 1690	24595 ± 328	385536 ± 5397	101722 ± 862
Iter. 1	78636 ± 1062	20688 ± 211	309876 ± 3450	83373 ± 676
Iter. 2	31098 ± 250	19797 ± 203	121031 ± 730	79373 ± 661
Iter. 3	22853 ± 111	19445 ± 171	88618 ± 318	77820 ± 505

Table 6.5 Comparison between the calculated costs for runs of the algorithm with and without warm-starting.

In section 6.2.4 it was stated that the algorithm gets sufficiently close to convergence at 3 iterations. As can be seen from table 6.5 the runs without warm-starting reach higher costs at 3 iterations than the cost warm-started runs reach after just one iteration.

The algorithm was finally run on both video recordings, with and without warm-starting, for a number of iterations equal to 1, 2 and finally 3. The captured video results for the first 450 frames are available in a Youtube playlist at the address `https://youtube.com/playlist?list=PLMChF1jy7IhqkeNyWt_jjZ5-xODGdqU80`. Some of the 1 MP videos in the playlist have significantly shorter lengths than the others which is caused by the fact that the display thread of the CUDA implementation was sometimes slower than the reconstruction thread and therefore skipped a few frames.

To clearly show the comparison between the recorded results, a section taken from each reconstruction of the 1 MP video recording focusing on one moving particle at frame number 50 can be found in figure 6.8. From the figure, it can be seen that the electrodes are really well reconstructed on all results of the algorithm that were warm-started. That is actually true for all non-moving objects in the warm-started reconstructions. On the other hand, executions of the algorithm that were not warm-started struggle with non-moving objects with only becoming satisfactory after 3 iterations. Twin-images were successfully suppressed in each reconstruction with recognizable features.

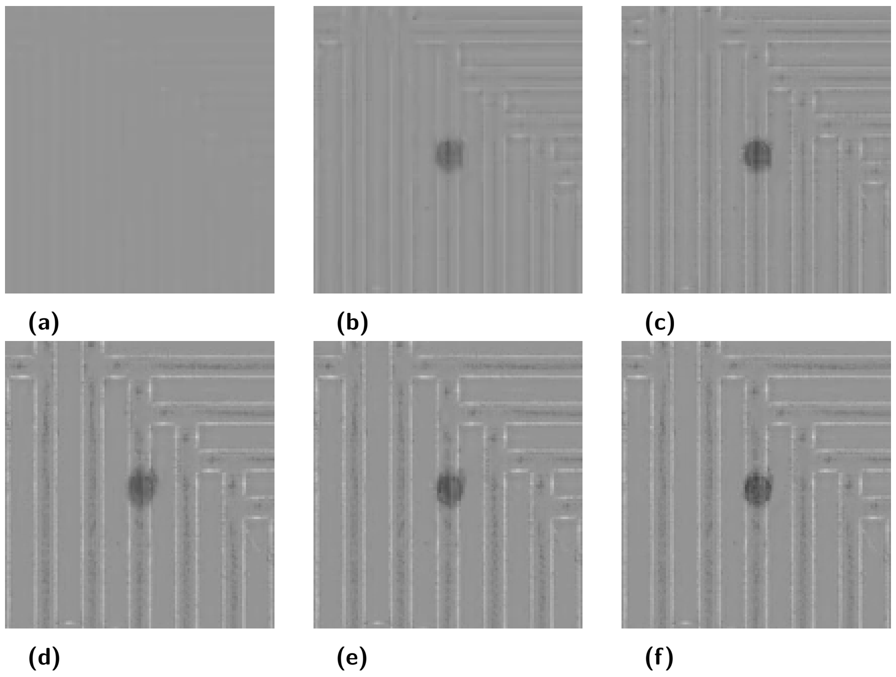


Figure 6.8 Comparisons between the results of different runs of the algorithm on the 1 MP hologram video recording. The comparison is of the same frame and focus is on a currently moving polystyrene particle. Runs (a)-(c) are not warm-started as opposed to runs (d)-(e). Executions (a) and (d) ran for 1 iteration, (b) and (e) ran for 2 iterations and, finally, (c) and (f) ran for 3 iterations.

In figure 6.8, it can be seen that the moving particle looks particularly blurred for warm-started reconstructions. This is most probably caused by the fact that the moving particle retains its position from the previous frame in the initial estimates of a warm-started run. In a low number of iterations, FISTA is unable to forget the previous position. This unfortunate drawback will be overlooked in favour of well reconstructed non-moving objects and low execution times, especially as the distortion on the moving object is not that substantial at 2 and more iterations.

Additionally, in the recorded videos available online, it can be seen that the moving particle gets heavily distorted when passing over individual electrodes in reconstructions of the reduced 1 MP holograms but not in reconstructions of the 4 MP holograms. This is most probably caused by the loss of information caused by reducing the holograms.

On the micromanipulation platform, FISTA will be run on reduced holograms for 2 iterations. The distortion on the moving objects is not significant enough to warrant increasing the computation time.

6.4 Integration Testing

The CUDA implementation of FISTA was successfully integrated into the micromanipulation code running on the provided platform. This integration was successfully tested when a micromanipulation experiment was running with FISTA reconstructing each captured hologram in real-time. The resulting images were then portrayed on a display connected to the platform. Each captured frame which originally had 4 MP was reduced to 1 MP and passed for processing to FISTA, which ran for 2 iterations. The same section taken from several captured frames from this experiment can be seen in figure 6.9. The complete recording of the experiment can be found online at the address `https://youtu.be/2uY98i34kaE`.

This experiment was performed with obsolete parameter tunings with $z = 2.303\,\mathrm{mm}$ and $\mu = 0.08$. Additionally, there was an error in this earlier implementation, where normally the final estimate received from FISTA should be offset at each pixel by the first value of the backpropagation kernel, while in this experiment, the final estimate was offset by a real one. The resulting reconstructions were still satisfactory.

6.5 Additional reconstructions

This section is more of a side note, just to show that the algorithm was also tested on holograms that do not match the usual setup with polystyrene particles over the electrode fields. Two holograms that differ from the usually expected sample were also passed to FISTA to see how well it would fare.

The first of these holograms is of a situation where an opaque microscopic diode was dropped on the electrode field. The hologram and the result of its reconstruction can be seen in figure 6.10.

The second hologram captured the situation where a number of opaque glass balls was dropped on the electrode field. The hologram, the result of its

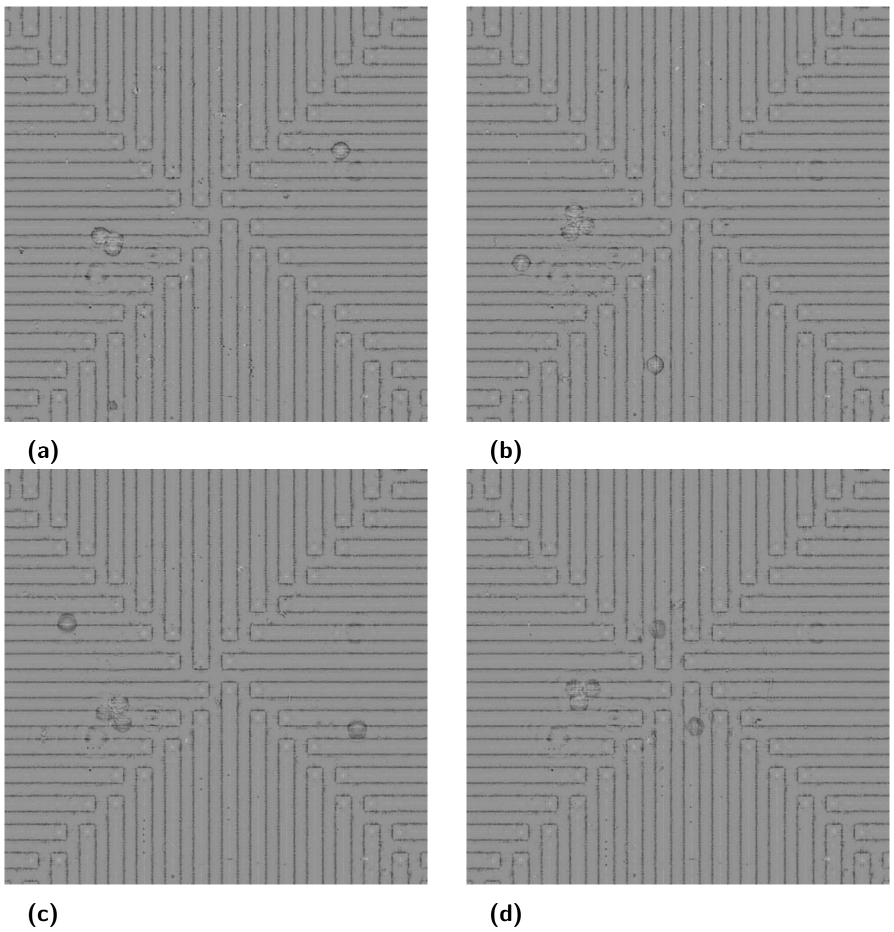


Figure 6.9 Several frames captured during a micromanipulation experiment on the provided platform. The frames captured are frames number (a) 40, (b) 800, (c) 1500, (d) 2000.

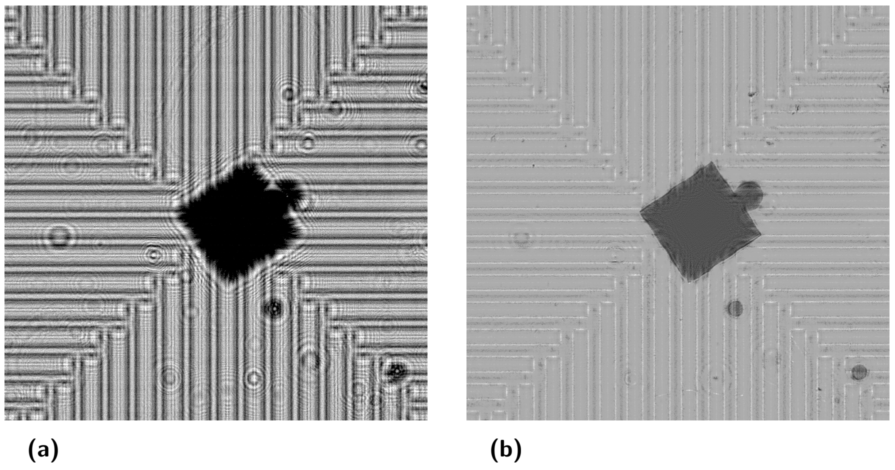


Figure 6.10 Result of FISTA reconstruction for a hologram containing opaque microscopic diode. (a) The original hologram, (b) the result of FISTA reconstruction.

reconstruction and the same situation captured on an optical microscope can be seen in figure 6.11.

6.6 Results and possible improvements

From the previous sections in this chapter, it can be seen that all of the different implementations reached satisfactory results: The selected method can almost completely remove the effect of twin images as well as noise from the reconstructed complex planar wave at the sample plane. More so, if the algorithm is properly tuned, the results of processing holograms with the implemented algorithm are comparable to recordings captured by a regular optical microscope as can be seen on figure 6.7.

With the use of warm-starting and reduction of input holograms from 4 MP to 1 MP, it is also possible to confidently say that the algorithm can process holograms in real-time. This was, actually, successfully confirmed by running a dielectrophoretic micromanipulation experiment with the code in which the phase reconstruction algorithm was integrated without noticing any significant slowdown that could negatively affect the manipulation process.

Therefore, it can be said that all of the tasks presented in the assignment of this thesis were fully completed.

However, despite these positive results, perhaps there is still a matter that can be addressed to perhaps improve the quality of the phase reconstruction

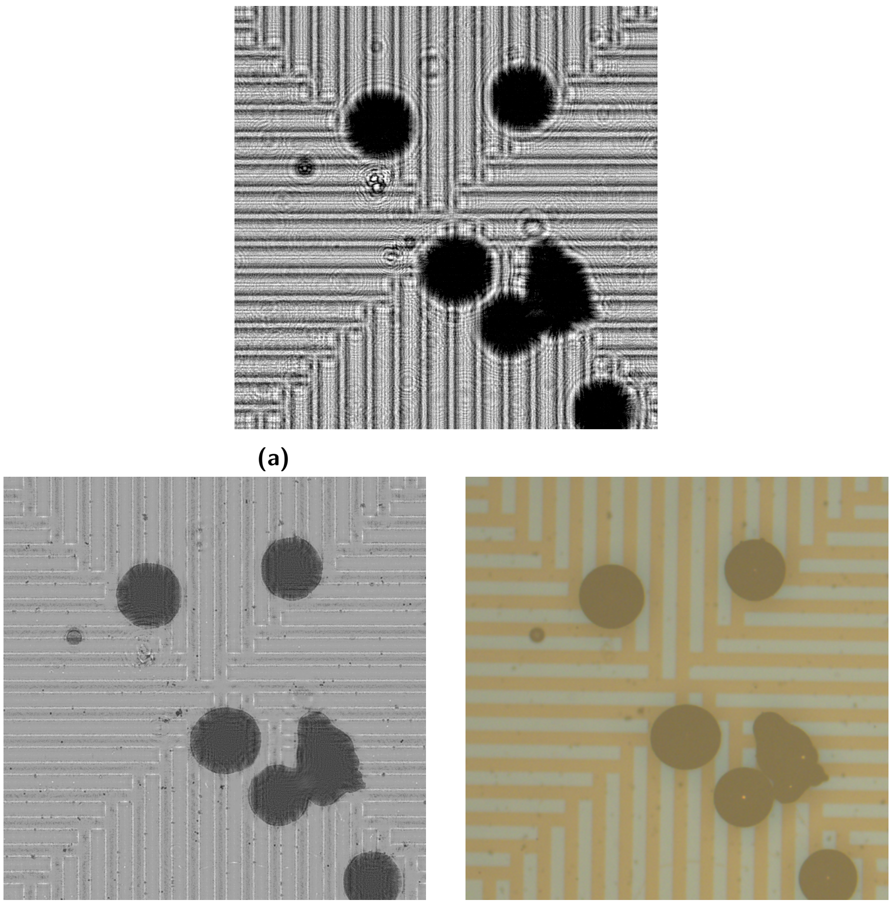


Figure 6.11 Result of FISTA reconstruction for a hologram containing opaque glass balls. (a) The original hologram, (b) the result of FISTA reconstruction and (c) the same situation captured on an optical microscope.

results.

6.6.1 Incorporation of prior knowledge

Since the electrode field remains static throughout all holograms captured on the provided platform, further research could be invested into finding ways of incorporating the knowledge of their support into the phase reconstruction algorithm. This could improve the way they appear in the phase reconstruction results. Such incorporation has to be done with special care as not to negatively affect the way that the polystyrene particles appear in the same reconstruction results.

Chapter 7

Conclusion

The main goal of this thesis was to design, implement and test a phase reconstruction algorithm that can be used in real-time alongside an already existing code on a dielectrophoretic micromanipulation platform. On this platform, the algorithm should provide reconstructions of captured holograms with completely suppressed twin-images that could be used for displaying. This goal was fully achieved with just a small drawback.

First, a suitable method was selected from already existing methods for phase reconstruction in in-line digital holography. The selected method approaches phase reconstruction as an inverse problem regularized by sparsity and positivity constraints. The goal of this problem is to estimate the disturbance caused by the observed objects at the sample plane. The method uses FISTA as the optimization strategy.

A prototype of the selected phase reconstruction algorithm was then implemented in Matlab, where it was first tested against simulated holograms. These tests showed that the algorithm is really appropriate for the given problem. The algorithm in the prototype was then tuned against holograms captured on the micromanipulation platform. The tuned algorithm was then able to significantly suppress the effect of twin-images in reconstructions of holograms captured on the platform after just 5 iterations.

The algorithm was then implemented in CUDA with warm-starting to test its capability of running in real-time. It was determined that if warm-starting is enabled and the processed holograms are reduced to 1 MP, the algorithm can run in real-time and result in good reconstructions after just 2 iterations. The drawback here is that the moving particles are, unfortunately, affected by noticeable blurring in their reconstructions.

The algorithm was finally integrated into the existing code on the dielectrophoretic micromanipulation platform. This integration was then successfully tested by running a feedback dielectrophoretic micromanipulation experiment on the platform. During this experiment, reconstructions estimated by the implemented algorithm were portrayed for each captured hologram on a display connected to the platform. The integrated algorithm did not negatively affect the process of micromanipulation.

With the quality of hologram reconstructions and the speed at which they are calculated, the implemented algorithm can be considered an improvement when compared to common phase reconstruction algorithms for in-line holographic microscopy.

Bibliography

[1] M. Gurtner and J. Zemánek, “Twin-beam real-time position estimation of micro-objects in 3d,” *Measurement Science and Technology*, vol. 27, p. 127003, Nov. 2016.

[2] M. Gurtner, *Real-time Optimization-based Control and Estimation for Dielectrophoretic Micromanipulation.* Master’s thesis, Czech Technical University in Prague, Prague, Feb. 2016.

[3] U. Schnars, C. Falldorf, J. Watson, and W. Jüptner, *Digital Holography and Wavefront Sensing.* Springer Berlin Heidelberg, 2015.

[4] D. Gabor, “A new microscopic principle,” *Nature*, vol. 161, pp. 777–778, May 1948.

[5] J. Goodman, *Introduction to Fourier optics.* New York: McGraw-Hill, 1996.

[6] L. Onural, “Digital decoding of in-line holograms,” *Optical Engineering*, vol. 26, p. 261124, Nov. 1987.

[7] L. Allen and M. Oxley, “Phase retrieval from series of images obtained by defocus variation,” *Optics Communications*, vol. 199, pp. 65–75, Nov. 2001.

[8] A. Greenbaum, W. Luo, T.-W. Su, Z. Göröcs, L. Xue, S. O. Isikman, A. F. Coskun, O. Mudanyali, and A. Ozcan, “Imaging without lenses: achievements and remaining challenges of wide-field on-chip microscopy,” *Nature Methods*, vol. 9, pp. 889–895, Aug. 2012.

[9] K. Nugent, “Twin-image elimination in gabor holography,” *Optics Communications*, vol. 78, pp. 293–299, Sept. 1990.

[10] S. Yang, X. Xie, Y. Zhao, and C. Jia, “Reconstruction of near-field in-line hologram,” *Optics Communications*, vol. 159, pp. 29–31, Jan. 1999.

[11] A. J. Devaney, “Noniterative reconstruction of complex-valued objects from two intensity measurements,” *Optical Engineering*, vol. 33, p. 3243, Oct. 1994.

[12] G. Koren, F. Polack, and D. Joyeux, “Twin-image elimination in in-line holography of finite-support complex objects,” *Optics Letters*, vol. 16, p. 1979, Dec. 1991.

[13] A. Lannes, “Correction des effets de l'image jumelle et du terme quadratique en holographie de gabor,” *Optics Communications*, vol. 20, pp. 356–359, Mar. 1977.

[14] L. Denis, C. Fournier, T. Fournel, and C. Ducottet, “Numerical suppression of the twin image in in-line holography of a volume of micro-objects,” *Measurement Science and Technology*, vol. 19, p. 074004, May 2008.

[15] R. W. Gerchberg, “A practical algorithm for the determination of phase from image and diffraction plane pictures,” *Optik*, vol. 35, pp. 237–246, 1972.

[16] T. Latychevskaia, “Iterative phase retrieval for digital holography: tutorial,” *Journal of the Optical Society of America A*, vol. 36, p. D31, Nov. 2019.

[17] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, “Phase retrieval with application to optical imaging: A contemporary overview,” *IEEE Signal Processing Magazine*, vol. 32, pp. 87–109, May 2015.

[18] J. R. Fienup, “Reconstruction of an object from the modulus of its fourier transform,” *Optics Letters*, vol. 3, p. 27, July 1978.

[19] J. R. Fienup, “Phase retrieval algorithms: a comparison,” *Applied Optics*, vol. 21, p. 2758, Aug. 1982.

[20] T. Latychevskaia and H.-W. Fink, “Solution to the twin image problem in holography,” *Physical Review Letters*, vol. 98, June 2007.

[21] S. Mukherjee and C. S. Seelamantula, “An iterative algorithm for phase retrieval with sparsity constraints: application to frequency domain optical coherence tomography,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Mar. 2012.

[22] M. L. Moravec, J. K. Romberg, and R. G. Baraniuk, "Compressive phase retrieval," in *Wavelets XII* (D. V. D. Ville, V. K. Goyal, and M. Papadakis, eds.), SPIE, Sept. 2007.

[23] F. Momey, L. Denis, T. Olivier, and C. Fournier, "From fienup's phase retrieval techniques to regularized inversion for in-line holography: tutorial," *Journal of the Optical Society of America A*, vol. 36, p. D62, Nov. 2019.

[24] F. Jolivet, F. Momey, L. Denis, L. Méès, N. Faure, N. Grosjean, F. Pinston, J.-L. Marié, and C. Fournier, "Regularized reconstruction of absorbing and phase objects from a single in-line hologram, application to fluid mechanics and micro-biology," *Optics Express*, vol. 26, p. 8923, Mar. 2018.

[25] A. Berdeu, O. Flasseur, L. Méès, L. Denis, F. Momey, T. Olivier, N. Grosjean, and C. Fournier, "Reconstruction of in-line holograms: combining model-based and regularized inversion," *Optics Express*, vol. 27, p. 14951, May 2019.

[26] A. Berdeu, T. Olivier, F. Momey, L. Denis, F. Pinston, N. Faure, and C. Fournier, "Joint reconstruction of an in-focus image and of the background signal in in-line holographic microscopy," 2020.

[27] A. Berdeu, O. Flasseur, L. Denis, F. Momey, M. Loïc, N. Grosjean, and C. Fournier, "Joint reconstruction in in-line holography combining parametric and non-parametric inverse approaches: Application to fluid mechanics," in *Holophi5 : 5ème rencontre francophone d'holographie numérique appliquée à la métrologie des fluides*, (Montpellier, France), Nov. 2018.

[28] Y. Rivenson, Y. Zhang, H. Günaydın, D. Teng, and A. Ozcan, "Phase recovery and holographic image reconstruction using deep learning in neural networks," *Light: Science & Applications*, vol. 7, pp. 17141–17141, Oct. 2017.

[29] Y. Wu, Y. Rivenson, Y. Zhang, Z. Wei, H. Günaydin, X. Lin, and A. Ozcan, "Extended depth-of-field in holographic imaging using deep-learning-based autofocusing and phase recovery," *Optica*, vol. 5, p. 704, May 2018.

[30] A. Sinha, J. Lee, S. Li, and G. Barbastathis, "Lensless computational imaging through deep learning," *Optica*, vol. 4, p. 1117, Sept. 2017.

[31] N. Parikh and S. Boyd, “Proximal algorithms,” *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[32] A. Beck and M. Teboulle, “Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems,” *IEEE Transactions on Image Processing*, vol. 18, pp. 2419–2434, Nov. 2009.

[33] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183–202, Jan. 2009.

[34] Y. E. Nesterov, “A method of solving a convex programming problem with convergence rate $\mathrm{o}\left(\frac{1}{k^2}\right)$,” *Dokl. Akad. Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.