Diplomarbeit

A Fourier Ptychographic Approach to Super-Resolution Microscopy

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Physik

eingereicht von

Moritz Siegel

Matrikelnummer 0913499

ausgeführt am

Austrian Institute of Technology (Ait)

in Kooperation mit der

Fakultät für Angewandte Physik (Iap)

an der

Technischen Universität (Tu) Wien

Betreuer: Univ. Prof. Dipl.-Ing. Dr.techn. Gerhard Schütz
Mitwirkung: FH-Prof. DI Dr. Lukas Traxler, BSc

Wien, February 7, 2023 ....
(Unterschrift Verfasser) (Unterschrift Betreuer)

a

Diploma Thesis

A Fourier Ptychographic Approach to Super-Resolution Microscopy

concluded at the

Austrian Institute of Technology (Ait)

instructed by

Lukas Traxler

in cooperation with the

Institute of Applied Physics (Iap)

at the

Technische Universität (Tu) Wien

advised by

Gerhard Schütz

of

Moritz Siegel

Vienna
February 7, 2023

moritz.siegel@tuwien.ac.at
https://github.com/imrahilias/fourierptychography

Abstract

In most optic microcopy systems, images are captured using a CCD/CMOS sensor, where the phases of the converted photons are inevitably lost. Fourier ptychographic microscopy (FPM) circumvents this issue by capturing microscopy images illuminated from different angles, and Fourier transforming them computationally (hence the name). Reconstructing the complex object not only yields amplitude but also phase information, enhanced up to super-resolution.

Yet one disadvantage remains unsolved: FPM is a very ill-posed problem, the algorithm is not guaranteed to converge to the correct solution, if it converges at all. In practice this means that there is reasonable doubt if the recovered image actually represents the object under the microscope.

This work inquires the quality of FPM reconstruction under variation of important system parameters in simulation and experiment. It shows that the alignment of the illumination source is quite critical: Even 0.2 degrees off renders reconstruction useless.

This thesis further shows that brightness variations of the individual LEDs are not that important, even assuming unrealistically high deviations, the impact on reconstruction is still negligible. This paper thus furthers the cost-benefit analysis of which amount of computation time should be spent on digital post-correction.

The preceding construction of a FPM prototype, on both hardware and software level is extensively covered in my Projektarbeit / Student Project at the TU Wien [Siegel, 2021b]. A digital version of all the source code is available in my git repository [Siegel, 2021a].

Keywords

fourier ptychographic imaging,
super-resolution microscopy,
phase retrieval,
coherent imaging,
numerical robustness,
synthetic aperture

Kurzfassung

Die meisten Optischen Mikroskope verwenden CCD/CMOS Sensoren für die Aufnahmen, dabei geht allerdings die Information über die Phase der einzelnen Photonen unwiederbringlich verloren. Fourier Ptychographie (Fourier ptychographic microscopy, FPM) umgeht dieses Problem indem Aufnahmen unter verschiedenen Beleuchtungswinkeln im Fourier-Raum kombiniert werden. Diese digitale Rekonstruktion beinhaltet nun nicht nur die Information über die Amplituden, sondern auch die der Phasen, und beides mit einer höheren Auflösung als eine einzelne Aufnahme ermöglicht (super-resolution).

Ein gravierender Nachteil bleibt jedoch: Numerisch betrachtet ist FPM ein sehr schlecht gestelltes Problem, der Algorithmus konvergiert nicht zwingend zur korrekten Lösung, falls er überhaupt konvergiert. In der Praxis wirft das Zweifel auf, inwiefern die Rekonstruktionen tatsächlich das untersuchten Objekt darstellen.

Diese Diplomarbeit untersucht die qualitative Einschätzbarkeit der FPM Rekonstruktionen unter Variation verschiedener Systemparameter, sowohl in Simulationen als auch experimentell. Der Ausrichtung der Beleuchtung kommt hier eine tragende Rolle zu, es zeigt ich, dass schon eine Abweichung um 0.2 Grad die Rekonstruktion vollkommen verfälscht.

Im Gegenteil dazu ist die Lichtstärke der einzelnen LEDs nicht von überragender Bedeutung, sogar unter für die Praxis unplausibel großen Schwankungen. Diese Arbeit vertieft so das Verständnis, welche Korrekturen in welchem Umfang im Rahmen der Nachbearbeitung Sinn machen.

Die vorangegangene Konzeption und Konstruktion eines FPM Prototyps inklusive Software ist im Detail in meiner Projektarbeit an der TU Wien [Siegel, 2021b] beschrieben. Eine digitale Sammlung allen verwendeten Quellcodes ist öffentlich zugänglich in meinem Git Repository unter [Siegel, 2021a].

Acknowledgments

First and foremost I want to thank Lukas Traxler for his wonderful way of advising my work throughout the past year. His patience, respectful and constructive criticism made all this a great adventure. Thanks for guiding me through the process of research; from designing an experiment all the way up to taking part at a conference.

I am grateful for having had the opportunity to present part of this thesis at the Electronic Imaging Conference 2021 (EI21), held by the Society for Imaging Science & Technology (IS&T); including a publication [Siegel et al., 2021].

A big thanks to all my colleges at the AIT, for the welcoming atmosphere and for sharing their expertise—from string formatting in C to insights in the art of writing papers. It was actually a lot of fun. Specifically, I want to thank Laurin Ginner for his comments on my paper for the conference proceedings and my talk at the Electronic Imaging Conference.

Thanks to the AIT for proposing the topic of Fourier Ptychography in the first place—I probably still would not know its very existence otherwise. Their Laboratories and funding provided the means for all the involved experiments.

Im am most appreciative of Gerhard Schütz, who's precise analysis of my work so often sparked the right questions to deepen my understanding in optics.

Last but not least I want to thank my parents Gabriele and Günter Siegel. From my earliest memories on, I remember both of them encouraging me to follow my interests—patiently enduring the results. Without their support, none of this would have been possible.

15000	*images captured*
5800	*lines octave/c/python code*
5300	*lines latex code*
6500	*words total in text*
2500	*different words*
1600	*hours of work*
400	*simulations*
100	*pages*
1	*project*
1	*thesis*
1	*paper*

Contents

Chapter 1

Introduction

Fourier ptychographic microscopy (FPM) is a computational imaging technique trading multiple images coherently illuminated from different angles and computation time for a high resolution complex image. measuring the phase of a light-field is non-trivial and usually quite complicated; comprising of lasers for coherent light creating diffraction patterns, which need special detectors with large dynamical range. FPM circumvents this problem by capturing generic microscopic images (in real space), and transforming them computationally into Fourier space—hence the name [Ou et al., 2013]. now one can utilise standard phase retrieval techniques dating back to the eighties [Fienup, 1982], to combine the images to a synthetic aperture. which yields a recovered image, not only of a resolution orders of magnitude higher than the images on their own; it even overcomes the resolution limit of the optical system, obtaining super-resolution [Ou et al., 2013]. Additionally, the final image is a complex object, it not only contains amplitude-, but phase information as well. This enables FPM to work on translucent objects, like biological samples, without the need of staining. Various work has been done on using FPM on translucent objects (e.g. cells), multi-colour-reconstruction [Ou et al., 2013], digital refocusing [Ou et al., 2013], high-speed high-throughput video [Sun et al., 2018]. Yet one disadvantage remains: The Gerchberg-Saxton (GS) algorithm [Gerchberg and Saxton, 1972], which is a type of gradient descent (the formulation is often non-convex), is not guaranteed to converge to the correct solution (the global minimum) [Yeh et al., 2015]. Relying solely on low resolution amplitude images as ground truth—the phase generally being unknown—assigning reconstruction quality is non-trivial. In practice this means that there is reasonable doubt if the recovered image actually represents the object under the microscope.

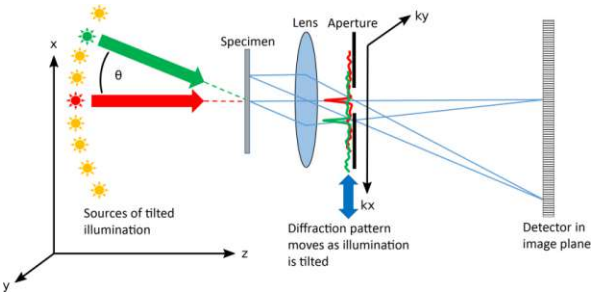


Figure 1.1: Schematic concept of the microscope for FPM, 3D illumination source position $\mathbf{r}$ *and 2D Fourier space locations* $\mathbf{k}$*, adapted from [22sm22, 2018] under CC-BY-SA license.*

A schematic illustration of a typical FPM setup is shown in Figure 1.1; θ angle-varied illumination source (left) illuminates a target, which results in a diffraction pattern at the aperture. The detector is at focus, and so captures generic images.

1.1 Fourier Optics

1.1.1 Fourier Transform

In many fields of optics—like astronomy, electron microscopy or crystallography—one is interested in the phase, yet only able to measure intensity $|O|^2$ of an object O; which consisting of both amplitude A and phase ϕ,

$$\begin{aligned} O &= A\ \mathrm{e}^{\mathrm{i}\phi} \\ O &\in \mathbb{C}^M, A \in \mathbb{R}^M, \phi \in (0, 2\pi) \end{aligned} \tag{1.1}$$

is related to its Fourier transform (FT) $\hat{O}$ by:

$$\begin{aligned} \hat{O} &= \mathcal{F}(O) = \int_{-\infty}^{\infty} O\ \mathrm{e}^{-\mathrm{i}2\pi\mathbf{kr}}\mathrm{d}\mathbf{r} \\ O &= \mathcal{F}^{-1}(\hat{O}) = \int_{-\infty}^{\infty} \hat{O}\ \mathrm{e}^{\mathrm{i}2\pi\mathbf{kr}}\mathrm{d}\mathbf{k} \end{aligned} \tag{1.2}$$

Where $\mathbf{r}$ is an M-dimensional spatial coordinate, and the hat ($\hat{}$) indicates spectral nature; the variable in question (in this case $\hat{O}$) belongs to the Fourier space. This nomenclature is continued throughout this thesis.

1.1.2 Discrete Fourier Transform

In practice one deals with discretely sampled data, so $\mathbf{r}$ and $\mathbf{k}$ are discrete arrays, hence the FT is replaced with the multidimensional discrete Fourier transform (DFT), where $\mathbf{r} = (r_1, r_2, \ldots, r_M)$ and $\mathbf{k} = (k_1, k_2, \ldots, k_M)$ are M-dimensional vectors of indices from 0 to $N-1$, which we define as $\mathbf{N\text{-}1} = (N_1 - 1, N_2 - 1, \ldots, N_M - 1)$:

$$\hat{O} = \mathcal{F}(O) = \sum_{\mathbf{r}=\mathbf{0}}^{\mathbf{N\text{-}1}} O\ \mathrm{e}^{-\mathrm{i}2\pi\mathbf{k}(\mathbf{r}/\mathbf{N})} \tag{1.3}$$

$$O = \mathcal{F}^{-1}(\hat{O}) = \frac{1}{\prod_{\ell=1}^{M} N_\ell} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N\text{-}1}} \hat{O}\ \mathrm{e}^{\mathrm{i}2\pi\mathbf{r}(\mathbf{k}/\mathbf{N})}$$

where the elements $\mathbf{r/N} = (r_1/N_1, r_2/N_2, \ldots, r_M/N_M)$, respective, $\mathbf{k/N} = (k_1/N_1, k_2/N_2, \ldots, k_M/N_M)$, are to be performed element-wise. The sum denotes a set of M nested sums.

Similar to the one-dimensional DFT asserting the input as a superposition of sinusoids, the multidimensional DFT expresses the input O as a superposition of plane waves. The direction of oscillation in space is $\mathbf{k/N}$, their amplitudes are $\hat{O}$.

For objects O consisting of real numbers—as images obtained from CCD/CMOS sensors do—their FTs have a conjugate even symmetry:

$$\hat{O} = \hat{O}_{k_1,k_2,\ldots,k_M} = \hat{O}^*_{\mathbf{-k}}\ , \quad O \in \mathbb{R}^M \tag{1.4}$$

where the *star exponent* (*) again denotes complex conjugation and $\mathbf{-k}$ is interpreted periodically (modulo, mod):

$$\mathbf{-k} = (-k_1 \bmod N_1, \ldots, -k_M \bmod N_M) \tag{1.5}$$

depending on the fact that $\omega_N = \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}}$ is a primitive root of unity:

$$\omega_N = \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}} \rightarrow \omega_N^N = 1 \tag{1.6}$$

Throughout this work—as in the majority of cases—one is dealing with images, so the dimension $M = 2$. For simplicity we define $\mathbf{r} = (x, y)$ and $\mathbf{k} = (k_x, k_y)$, both consisting of (N_x, N_y) values, leading to:

$$\begin{aligned} &\hat{O}_{k_x,k_y} = \mathcal{F}(O) = \\ &= \sum_{x=0}^{N_x-1} \mathrm{e}^{-\mathrm{i}2\pi k_x x/N_x} \sum_{y=0}^{N_y-1} \mathrm{e}^{-\mathrm{i}2\pi k_y y/N_y}\ O_{x,y} \\ &O_{x,y} = \mathcal{F}^{-1}(\hat{O}) = \\ &= \frac{1}{N_x N_y} \sum_{x=0}^{N_x-1} \mathrm{e}^{\mathrm{i}2\pi x k_x/N_x} \sum_{y=0}^{N_y-1} \mathrm{e}^{\mathrm{i}2\pi y k_y/N_y} \hat{O}_{k_x,k_y} \end{aligned} \tag{1.7}$$

Of course those are calculated computationally using the fast Fourier transform

(FFT), which explaining in detail would exert this thesis. The interested reader is referred to scientific documentation from [Cooley and Tukey, 1965] onward.

1.1.3 Shift Theorem

Multiplying a complex object O by a linear phase $\exp(\mathrm{i}\frac{2\pi}{N}\mathbf{nr})$ corresponds to a circular shift of its spectrum $\hat{O} : \hat{O}_{\mathbf{k-n}}$ and vica versa:

$$\begin{aligned} \mathcal{F}(O\,\mathrm{e}^{\mathrm{i}\frac{2\pi}{N}\mathbf{rn}}) &= \hat{O}_{\mathbf{k-n}} \\ \mathcal{F}^{-1}(\hat{O}_{\mathbf{r-n}}) &= \hat{O}\,\mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}\mathbf{kn}} \end{aligned} \tag{1.8}$$

where the subscript is again interpreted modulo N, as elaborated in Equation 1.5.

1.1.4 Convolution Theorem

The FT translates between *convolution* $(*)$ and *multiplication* $(\cdot)$ of functions. The convolution theorem states that a convolution of two sequences f, g;

$$\begin{aligned} h(x) &= f(x) * g(x) \\ &= \int_{-\infty}^{\infty} f(y)g(x-y)\mathrm{d}y \end{aligned} \tag{1.9}$$

can be obtained as the inverse transform of the product of the individual transforms:

$$\hat{h}(k) = \hat{f}(k) \cdot \hat{g}(k) \tag{1.10}$$

1.1.5 Cross-correlation Theorem

Similar to the convolution in section 1.1.4, it can be shown that if $h(x)$ is the cross-correlation $(\star)$ of $f(x)$ and $g(x)$;

$$\begin{aligned} h(x) &= f(x) \star g(x) \\ &= \int_{-\infty}^{\infty} f^*(x')g(x+x')\mathrm{d}x' \end{aligned} \tag{1.11}$$

then its FT $\hat{h}(k)$ is:

$$\hat{h}(k) = \hat{f}^*(k) \cdot \hat{g}(k) \tag{1.12}$$

where $\hat{f}^*$ denotes the *complex conjugate*, not to be confused with the *convolution operator* $(*)$.

1.1.6 Autocorrelation

As the name autocorrelation suggests, this process cross-correlates $(\star)$ a signal with a delayed copy of itself as a function of delay:

$$h(x) = f(x) \star f(x) \tag{1.13}$$

The autocorrelation is thus a special case of the cross-correlation, for which the FT is given by equation 1.12, and simplifies to:

$$\hat{h}(k) = \hat{f}^*(k) \cdot \hat{f}(k) = |\hat{f}(k)|^2 \tag{1.14}$$

1.2 Gerchberg-Saxton

The Gerchberg-Saxton (GS) was originally constructed to allow the reconstruction of the phase from two intensity measurements; One in the spatial domain $|\zeta|^2$, and one in the Fourier domain $|\hat{\chi}|^2$ [Gerchberg and Saxton, 1972].

The algorithm consists of only four steps (respective equations): 1^{st}, FT an estimate of the object O_ℓ (1.15); 2^{nd}, replace the modulus of the estimation $\hat{O}'_\ell$ whit the measured modulus $|\hat{\chi}|$ to get an estimation of the FT (1.16); 3^{rd}, inverse FT this estimate to O'_ℓ (1.17); 4^{th}, replace the modulus of the recovered image O'_ℓ whit the the measured modulus $|\zeta|$ to get an estimation of the image $O_{\ell+1}$ (1.18). For the n-th iteration:

$$\hat{O}_\ell = |\hat{O}_\ell|\,\mathrm{e}^{\mathrm{i}\hat{\phi}_\ell} = \mathcal{F}(O_\ell) \tag{1.15}$$

$$\hat{O}'_\ell = |\hat{\chi}|\,\mathrm{e}^{\mathrm{i}\hat{\phi}_\ell} \tag{1.16}$$

$$O'_\ell = |O'_\ell|\mathrm{e}^{\mathrm{i}\phi'_\ell} = \mathcal{F}^{-1}(\hat{O}'_\ell) \tag{1.17}$$

$$O_{\ell+1} = |\zeta|\,\mathrm{e}^{\mathrm{i}\phi'_\ell} \tag{1.18}$$

where ϕ_ℓ denotes the phase of O_ℓ, $|z|$ denotes the modulus of the complex value z

(see Equation 2.8).

This algorithm converges to a solution that satisfies the constraints both in spatial and in Fourier domain; where the change to each of the objects in the $(\ell + 1)$-th iteration would be negligible. This change is commonly monitored using the squared error, either of the object in spatial domain $E^2_{\ell,r}$ or in the Fourier domain $E^2_{\ell,k}$:

$$E^2_{\ell,r} = \sum_{\mathbf{r}} (|O_\ell| - |\zeta|)^2 \tag{1.19}$$

$$E^2_{\ell,k} = \frac{1}{\eta^2} \sum_{\mathbf{k}} (|\hat{O}_\ell| - |\hat{\chi}|)^2 \tag{1.20}$$

where η denotes the number of elements (pixels). E^2_ℓ can be shown to decrease with each iteration; hence the common name error reduction algorithm [Fienup, 1982]:

$$E^2_{\ell+1,k} \leq E^2_{\ell,r} \leq \hat{E}^2_{\ell,k} \tag{1.21}$$

A generalized Gerchberg-Saxton algorithm can be used for a plethora of problems, where partial constraints (measured data or information known a priori) are known in each of two domains, usually spatial and Fourier domain [Fienup, 1982].

Chapter 2

Methods

2.1 Fourier Ptychography

The experimental setup of the Fourier ptychographic microscope constructed for this thesis, first proposed by [Zheng et al., 2013], is intriguingly simple. A two-dimensional sample is placed on the focal plane of a generic 4f-microscope. Using a low-NA objective lens allows a wide field of view, with the downside of lower magnification, thus lower resolution. Instead of generic illumination (usually collimated), a 32×32 light emitting diode (LED)s RGB panel is mounted below the objective, capable of illuminating the sample with plane coherent light waves from $N = 1024$ different directions. A schematic of a Fourier ptychographic microscope prototype is shown in Figure 1.1.

Some exemplary documentation of the entire Fourier ptychographic microscope prototype is shown in Figure 2.1 (right), including the laptop running the Python script for acquisition (left). The microscope is shown in Figure 2.2, and consists of the following parts: A) microscope objective (4f-tube), B) x-axis adjustment screw, C) focus slider, D) sample holder (without sample), E) uEye camera, F) RGB LED panel, G) Arduino driving the panel.

On the contrary to ptychography, only intensity images are acquired, using a monochrome industrial camera (uEye). To operate in the frequency domain, these images ι_n are Fourier transformed to $\hat{\iota}_n$ (described in detail in section 1.1.1). Throughout this thesis, all variables wearing a hat are the FTs of their counterparts.

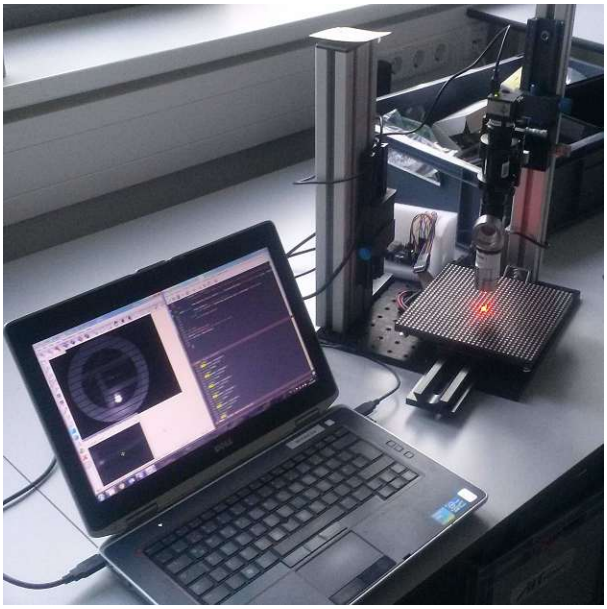


Figure 2.1: Fourier ptychographic microscope prototype version 1 (right), laptop running the Python script for acquisition (left).

These low-resolution images ι_n, are then used to reconstruct a high-resolution complex object $O = A\,e^{i\phi}$, consisting of amplitude $A = |O|$, and the corresponding phase Φ (Equation 1.1). Throughout this section, the uppercase letters (e.g. $\hat{O}, A, \Phi$) represent high-

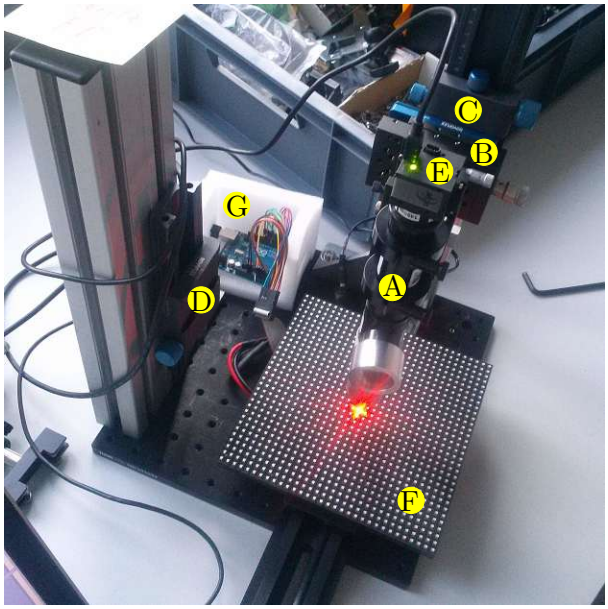


Figure 2.2: Fourier ptychographic microscope prototype version 1: A) microscope objective (4f-tube), B) x-axis adjustment screw, C) focus slider, D) sample holder (without sample), E) uEye camera, F) RGB LED panel, G) Arduino driving the panel.

resolution elements, whereas the lower-case elements (e.g. $\hat{o}, a_n, \phi$) denote their counterparts, cropped to the size of the low-resolution images ι_n.

Based on the Gerchberg-Saxton GS algorithm (section 1.2), the recovery alternates between spatial domain with the corresponding spatial coordinates $\mathbf{r} = (x, y)$; and the Fourier domain with the spatial frequency coordinates $\mathbf{k} = (k_x, k_y)$.

For the entire computation the language (and development ecosystem of) GNU Octave, the popular and free MATLAB sister is used, see the Appendix B for abbreviated code listings of the programs; the full code is online available [Siegel, 2021a]).

The complex light field of the object O, illuminated by a plane wave $\mathrm{e}^{\mathrm{i}\mathbf{k}_n\mathbf{r}}$ from the direction $\mathbf{k}_n$, arriving at the detector as the images $\iota_n = |o_n|^2$, can be modelled as a coherent imaging process:

$$o_n = H * (O\,\mathrm{e}^{\mathrm{i}\mathbf{k}_n\mathbf{r}}) \tag{2.1}$$

where $(*)$ denotes the two-dimensional convolution, $\mathbf{r}$ are spatial coordinates, and $\mathbf{k}$ denote spatial frequency coordinates. H denotes the point spread function (PSF) of the lens-system. Transforming this equation to the spatial frequency domain using Equation 1.2, and particularly the convolution theorem in Equation 1.10, one obtains the multiplication:

$$\hat{o} = \hat{H}\,\hat{O}_{\mathbf{k}-\mathbf{k}_n} \tag{2.2}$$

where $\hat{O}_{\mathbf{k}-\mathbf{k}_n}$ corresponds to the spectrum of the object $\hat{O}_n$, shifted about $\mathbf{k}_n$ in the Fourier domain (see the shift theorem, Equation 1.8). $\hat{H}$ denotes the optical transfer function (OTF), which is itself both the FT of the point spread function (H), and the autocorrelation $(\star)$ of the pupil function $\hat{p}$—which according to the cross-correlation theorem (Equation 1.14) is:

$$\hat{H} = \mathcal{F}(H) = \hat{p} \star \hat{p} \overset{ideal}{=} \hat{p} \tag{2.3}$$

where the ideal pupil function $\hat{p}$, assuming a perfect lens and coherent light, is defined as:

$$\hat{p} = \begin{cases} 1, & \text{if } |\mathbf{k}_n| < k_{co} \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

where k_{co} denotes the spatial cutoff frequency of this optical system:

$$k_{co} = \mathrm{NA}\,k_0 = \mathrm{NA}\,\frac{2\pi}{\lambda_0} \tag{2.5}$$

and k_0 respective λ_0 denote the wave number, respective wavelength, of the incident coherent light.

Through the ocular, the complex light field is focused on the detector, the resulting (still

complex) object o'_n essentially is the inverse FT of the complex light field $\hat{o}_n$:

$$o'_n = \mathcal{F}^{-1}(\hat{o}) = \mathcal{F}^{-1}(\hat{p}\,\hat{O}_{\mathbf{k}-\mathbf{k}_n}) \tag{2.6}$$

The n-th image captured by the camera ι_n now is a discretely sampled ($\leftarrow$) intensity of the (magnified) complex object o'_n, who's relation with its amplitude is:

$$a_n = \sqrt{\iota_n} \leftarrow |o'_n| \tag{2.7}$$

where $|z|$ denotes the modulus (absolute value) of the complex object z, which counts for both domains $\mathbf{r}$ and $\mathbf{k}$ alike, and is defined as:

$$|z| = \sqrt{zz^*} = \sqrt{\Re(z) + \Im(z)} \quad , \forall z \in \mathbb{C} \tag{2.8}$$

where this modulus $|z|$ always is a real value—as elaborated in Figure 2.3:

$$|z| \in \mathbb{R} \tag{2.9}$$

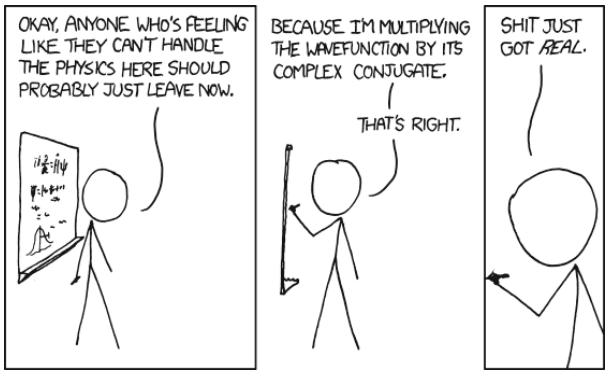


Figure 2.3: About complex conjugates [Randall, 2011a], obtained under CC-BY-SA license.

As for the GS algorithm, the proposed algorithm [Zheng et al., 2013] relies on a support constraint in two domains (compare to section 1.2). the constraint in spatial domain is the accordance of the low-resolution image with the corresponding part of the object. In Fourier domain, the coherent transfer function serves as a well-defined constraint. In a nutshell, the proposed algorithm is converged if the following two conditions are met for all n images:

$$|o_n| \approx a_n, \quad \forall n \tag{2.10}$$

$$\hat{o}_n * \hat{p} \approx \hat{a}_n, \quad \forall n \tag{2.11}$$

which can be both expressed as the squared error e_n^2 of the respective image n in the following way:

$$e_{n,r}^2 = \sum_{\mathbf{r}} (|o_n| - a_n)|^2 \tag{2.12}$$

$$e_{n,k}^2 = \frac{1}{\eta^2} \sum_{\mathbf{k}} |\hat{o}_n * \hat{p} - \hat{a}_n|^2 \tag{2.13}$$

where the lowercase e indicates that the error is in the dimension of the low-resolution images ι_n of the size η pixels. Since both errors are somewhat equivalent (see section 1.2), we choose $e_{n,r}$ because of it saves one FT of the low-resolution image. It makes sense to define a squared error $E_{\ell,r}^2$, either based on $e_{n,r}$ or $e_{n,k}$ per pixel, as the FPM algorithm loops over all N images once:

$$E_{\ell,r} = \sqrt{\frac{1}{N} \sum_n \frac{1}{\eta^2} \sum_{\mathbf{r}} |o_n| - a_n|^2} \tag{2.14}$$

This sheds light on the quality of the reconstruction, more specifically: how good is the accordance of the complex object with all the measured images. As the algorithm converges, the quality of the reconstruction increases up to a point limited by measurement accuracy, reaching zero only in theory:

$$E_{\ell,r} \overset{\ell \gg}{\to} E_{min,r} \tag{2.15}$$

Naturally we are quite interested in the convergence of the algorithm, thus how much actually changed during the last iteration ℓ, on the basis of the complex object O. This holds

for the counterpart $\hat{O}$, which is preferred here, because of it saves one—somewhat costly—FT of the high-resolution object. For sake of computational convenience, we define a root mean square (RMS) error $E_{\ell,k}$ per pixel, expressing the change of the whole high-resolution image $\hat{O}$ as the FPM algorithm loops over all n images successively ℓ times:

$$E_{\ell,k} = \sqrt{\frac{1}{\zeta^4} \sum_{\mathbf{k}} |\hat{O}_\ell - \hat{O}_{\ell-1}|^2} \tag{2.16}$$

where ζ denotes the size of the high-resolution object O. Based on Equations 1.19 for the convergence of the GS algorithm, we assume this holds equally well for FPM, so this root mean square error (RMSE) diminishes, as we converge towards the solution:

$$E_{\ell,n} \overset{\ell \gg}{\to} 0 \tag{2.17}$$

Approaching convergence, the change of the estimation O through further iteration become negligible, which can be expressed an exit criterion for the FPM algorithm:

$$\ell_{conv} = \ell : E_{\ell,k} < \varepsilon \quad \forall \varepsilon \in \mathbb{R} \tag{2.18}$$

2.2 Recovery Procedure

Now we have all the ingredients to deduct the procedure for a Fourier ptychographic microscope prototype as first built by [Zheng et al., 2013], and shown in Algorithm 1. In the detailed description below, we will refer to the lines of the code shown in Algorithm 1 in brackets. An abbreviated version of the code is shown in Section B.1.5 of Appendix B; the full code is online available [Siegel, 2021a]).

Algorithm 1 Fourier Ptychography Core

```
1:  function REKOVER( a )
2:      while E_ℓ > ε do
3:          ℓ ← ℓ + 1
4:          for all k_n do
5:              ô_n ← (ζ/η)^2 Ô_{k−k_n} p̂
6:              o_n ← F^{-1}(ô_n)
7:              φ_n ← ∠o_n
8:              o_n ← a_n e^{iφ_n}
9:              Ô_{k−k_n} ← (η/ζ)^2 F(o_n) p̂
10:         Ô_ℓ ← Ô
11:         E_ℓ ← sqrt(< |Ô_ℓ − Ô_{ℓ−1}|^2 >)
12:     O ← F^{-1}(Ô)
13:     A ← |O|
14:     I ← |O|^2
15:     Φ ← ∠O
16:     return (Ô, E, O, A, I, Φ)
```

As a brief example for the stack of amplitudes **a**, the FPM algorithm operates on, both the amplitude $|o_n|$ and the magnitude of its spectrum $|\hat{o}_n|$ of three low-resolution images for arbitrary illumination angles θ_n are shown In Figure 2.4. The two examples on the top two rows lie within the bright-field (BF) region, the bottom row outside. One can clearly see the autocorrelation circles in the BF spectra, described in detail in section 2.6.

First, the captured images ι_n are transferred to amplitude $a_n = \sqrt{\iota_n}$ according to Equation 2.7—since the GS algorithm is based on amplitude information—which is then fed to the FPM algorithm as one stack of images **a**, thus stated in vector notation.

The core of FPM is an iterative outer loop, in which an inner loop is covering the images ι_n from all n different incident illumination wavevectors $\mathbf{k}_n$ (angle θ_n) sequentially (Lines 2–11); until the FPM algorithm can be considered converged after ℓ outer loops.

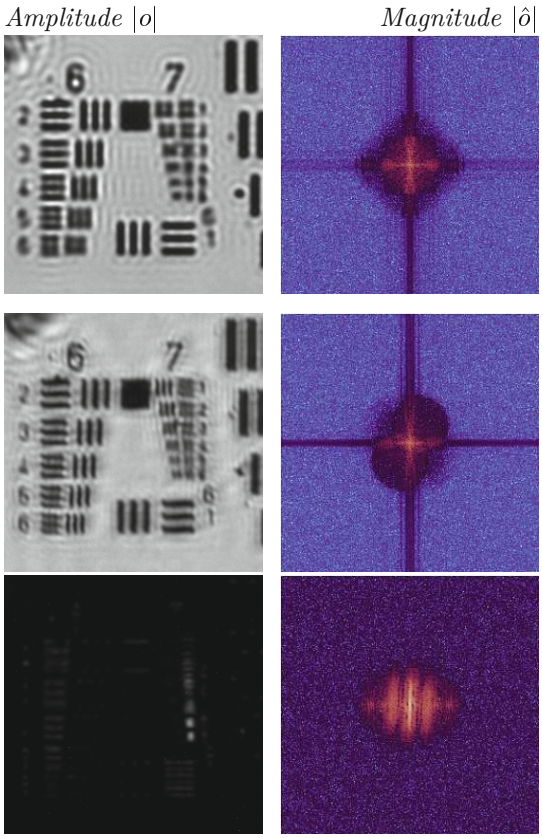


Figure 2.4: Comparison of both the amplitude of the object $|\iota_n|$ and the magnitude of the spectrum $|\hat{\iota}_n|^2$ for three arbitrary illumination angles θ_n; the top two examples within the BF region, the bottom one outside. One can clearly see the autocorrelation circles in the BF spectra.

Convergence is estimated by the pixel-wise RMSE E_ℓ, the deviation of the current high-resolution spectrum $\hat{O}_\ell$ to the last iteration's $\hat{O}_{\ell-1}$ (Line 11, Equation 2.16).

For all n low resolution amplitudes $a_n = \sqrt{\iota_n}$, in each of these ℓ loops; the part of the high-resolution (indicated by upper case letters) complex object's spectrum $\hat{O}$, corresponding to the incident illumination $\mathbf{k}_n$ (Equation 2.2), is rescaled to low-resolution (lower case letters) $\hat{o}_n$ with the factor $(\eta/\zeta)^2$

Figure 2.5: Magnitude of the spectrum of low-resolution complex object $\hat{o}$ during the first loop of FPM recovery. The size of the disc corresponds with the cutoff frequency $\pm k_{co}$.

and convolved with the pupil function $\hat{p}$ of the optical system (Line 5, Equation 2.3).

A snapshot of the complex object's spectrum $\hat{o}$ during the first loop is shown in Figure 2.5, note that the circle is not full, because the high-resolution spectrum only covers the parts accessed by previous locations $\mathbf{k}_n$.

The resulting low-resolution spectrum $\hat{o}_n$ is now inverse Fourier transformed to o_n (Line 6), and its phase ϕ_n is computed (Line 7). A snapshot of ϕ_n during the first loop is shown in Figure 2.6.

The estimation $\ell + 1$ is now given by the combination of the amplitude of the measured image a_n and the phase of the corresponding recovered object o_n (Line 8).

In the next step, the high-resolution spectrum $\hat{O}$ gets updated with the FT of that new estimation $\hat{o}_n$ (Line 9).

Reaching convergence, the resulting $\hat{O}$ is in-

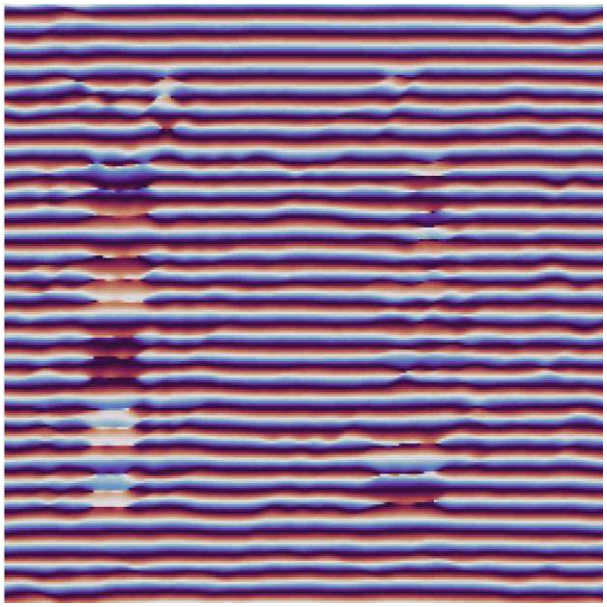


Figure 2.6: Snapshot of the estimated local phase ϕ_n during the first loop of FPM recovery. Clearly visible are vague structures of the Target, and sinusoidal patterns due to the shift k_n (see shift theorem Equation 1.8).

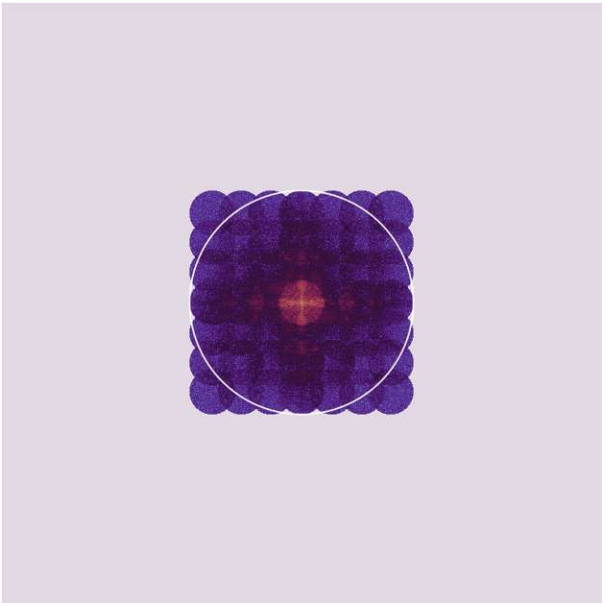


Figure 2.7: Magnitude of Spectrum of the high-resolution object $\hat{O}$ during the first loop of FPM recovery, with synthetic NA (white line). Clearly visible are the n updates (disks) to the spectrum. Note the size of the disks corresponds with the cutoff frequency $\pm\mathrm{k}_{co}$, also compare their relative size to the spectrum shown in Figure 2.5.

verse Fourier transformed, yielding the high-resolution complex object O; And thus amplitude A and phase Φ (Line 12–15, according to Equation 1.1).

2.3 Resolution & Sampling

2.3.1 Spatial Domain Sampling

Abbe showed 1873 that light with wavelength λ, traversing a medium with refractive index n and converging to a spot with half-angle θ leads to the resolvable distance d of:

$$d = \frac{\lambda}{2n\sin\theta} = \frac{\lambda}{2\ \mathrm{NA}} \tag{2.19}$$

where the numerical aperture (NA) is a dimensionless number characterising the range of angles over which the optical system is capable of accepting light:

$$\mathrm{NA} = n\sin\theta \tag{2.20}$$

Considering the refractive index $n \approx 1$ in air, the objective NA of our system being 0.055, and using the red LED (best coherence) with wavelength of about $\lambda_{red} \approx 632\,\mathrm{nm}$, we obtain a resolution limit of:

$$d_{obj} = \frac{\lambda_{red}}{2\ \mathrm{NA}_{obj}} \approx 5.7\,\mu\mathrm{m} \tag{2.21}$$

According to the Nyquist-Shannon sampling theorem this is twice the upper limit on the pixel-size of the imaging detector:

$$l_{ccd,max} = \frac{d_{obj}}{2} \approx 2.9\,\mu\mathrm{m} \tag{2.22}$$

2.3.2 Illumination & FPM NA

The total synthetic aperture NA_{fpm} of the FPM setup is shown by [Zheng et al., 2013] to be the sum of objective NA and illumination NA:

$$\mathrm{NA}_{fpm} = \mathrm{NA}_{obj} + \mathrm{NA}_{lum} \tag{2.23}$$

The illumination NA is easily calculated: The RGB panel features 15 LEDs in x and y direction, each 6 mm apart, giving a range of $\Delta x = \Delta y = 90\,\mathrm{mm}$. The relation of the illumination angle θ with panel dimensions $\Delta x, \Delta y$, and height of the objective lens Δz is according to the law of tangents:

$$\tan\theta = \frac{\Delta x}{\Delta z} \tag{2.24}$$

Resolving Equation 2.24 for θ, with Equation 2.20, and refractive index $n \approx 1$ for air yields:

$$\mathrm{NA}_{lum} = \sin\theta_{max} = \sin\left(\arctan\frac{\Delta x}{\Delta z}\right) \tag{2.25}$$

Given that in our setup the objective lens is located $z = 327\,\mathrm{mm}$ above the panel, the illumination NA is:

$$\mathrm{NA}_{lum} = \sin\left(\arctan\frac{90}{327}\right) \approx 0.27 \tag{2.26}$$

The total synthetic aperture NA_{fpm} of our FPM setup is according to Equation 2.23, the sum of objective NA (0.055) and illumination NA from Equation 2.26:

$$\mathrm{NA}_{fpm} = \mathrm{NA}_{obj} + \mathrm{NA}_{lum} \approx 0.32 \tag{2.27}$$

Taking this into account, Equation 2.21 suggests a lower limit for the resolution of the reconstructed image:

$$d_{fpm} = \frac{\lambda_{red}}{2\ \mathrm{NA}_{fpm}} \approx 990\,\mathrm{nm} \tag{2.28}$$

which is well below the theoretical resolution limit of the objective, see Equation 2.21, defining the term *super-resolution* microscopy.

2.3.3 Resolution Test Target

The resolution of the USAF1951 test target, see Table A.1, is evaluated the following way:

$$d_{usaf} = 2^{(n_g + (n_e - 1)/6)}\ \mathrm{lp/mm} \tag{2.29}$$

where n_g denotes the Group (-2 to 7), and n_e the Element (1 to 6) within that group, line pair (lp) means a black and a white line. One of the shown black lines then has a diameter of:

$$d_{line} = 2^{-(n_g + (n_e - 1)/6 + 1)}\ \mathrm{mm} \tag{2.30}$$

2.3.4 Fourier Domain Sampling

As elaborated in Section 2.1 the second convergence criterion the GS algorithm demands, is cast as a compulsory redundancy of the spectra stitched together in Fourier space. According to [Zheng et al., 2013] the optimum overlap is about 30%, which over satisfies the redundancy demand, as the Fourier space is successively overlapped; so most pixels are addressed significantly more than once.

2.3.5 Spatial Coherence

Fourier Ptychography generally operates in far-field, indicating that the detector is located at a distance d, *far* away from the emitting light source. Fraunhofer diffraction occurs when:

$$\frac{w^2}{d\lambda} \ll 1 \tag{2.31}$$

where w denotes the aperture, in our case resembled by the largest dimension of the source emitting light of the wavelength λ, viewed from distance d (the position of the object).

In our case the emitting area of the red LEDs is a square about 100 µm, estimated roughly by measuring the surface-mount-device (SMD) LED size, shown in Figure 2.8a; and comparing it with the emitting area of a single LED

captured by the microscope prototype, Figure 2.8b. As largest dimension we take the diagonal of the LED, which amounts to about 140 µm. So the object has to be set up at a distance d:

$$d \gg \frac{w^2}{\lambda} \approx 31\,\text{mm} \tag{2.32}$$

In all the experiments throughout this thesis the light source is positioned at far-field distance. A thorough discussion on spatial coherence in Fourier Ptychography is shown in [Dong et al., 2014].

Coherence Length

Assuming a circular aperture w (resembling a circular emitting area, in our case again approximated by the diagonal of the square emitting area of about 140 µm), the Van Cittert-Zernike theorem is often approximated to the following metric of a coherence length:

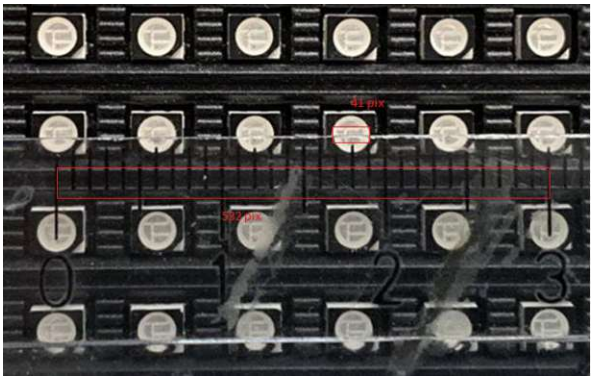
$$l_c = 1.22\frac{\lambda z}{w} \approx 1.8\,\text{mm} \tag{2.33}$$

where z is the distance of the object from the source, in our case this is 327 mm. At this object plane (z), one may assume an area with largest dimension of l_c as illuminated by a coherent plane wave.

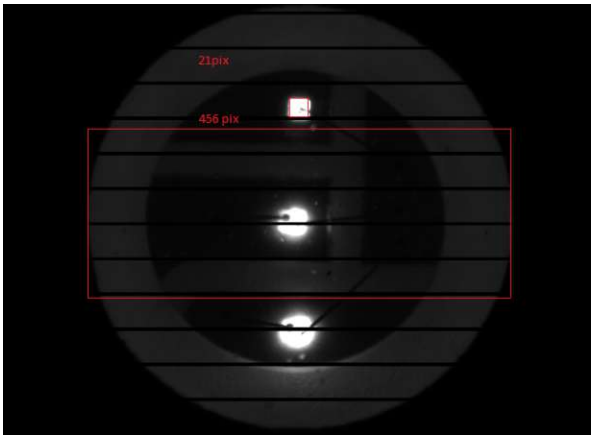
2.3.6 Temporal Coherence

Furthermore a LED emits somewhat (temporal) coherent light of a given wavelength, depending on the semiconductors used. The spectra for the red, green and blue LEDs of a typical SMD chip are schematically shown in Figure 2.9. The spatial coherence, full width half maximum (FWHM), is typically around:

$$\Delta\lambda_{led} \approx 30\,\text{nm} \tag{2.34}$$



(a) Photograph of the LED matrix, estimate of the individual RGB SMD LEDs size.



(b) Microscope image of one individual RGB SMD LED, estimate of the emitting area (red LED, top).

Figure 2.8: Images of the LED matrix respective of one individual RGB SMD LED, used for estimating the emitting area.

2.3.7 Field of View

In theory the FPM algorithm is able to operate on images of any size, as long as spatial coherence on the object-level is given [Zheng et al., 2013]; the field of view (FOV) of object needs to be smaller than the spatial coherence length of the illumination. Typically a LED panel is used for FPM, which has good spatial coherence, as derived in Section 2.3.5, so the analysed images can be quite large.

In practice though all these images need to

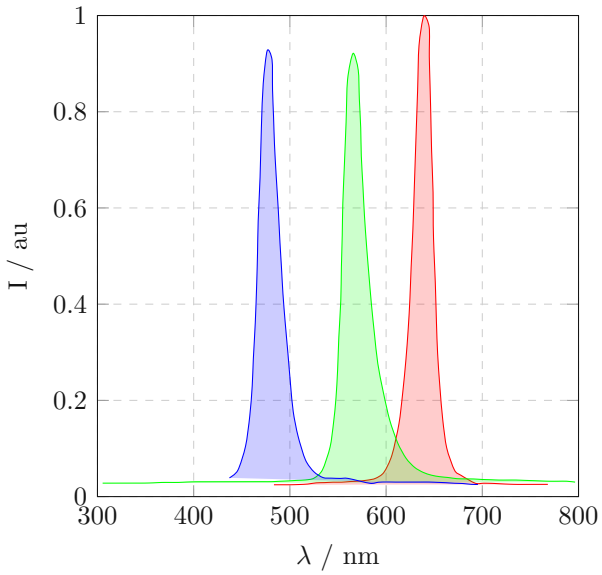


Figure 2.9: Schematics of a RGB SMD LED spectrum.

be kept in memory—otherwise one would need to repeatedly load all of them during every iteration of the recovery. Thus the captured images are usually not processed as whole, but are segmented and processed in patches, being stitched together subsequently [Zheng, 2016].

For the sake of simplification, throughout this thesis only the central part of the captured images is processed. Unless noted otherwise, this area of interest (AOI) is set to 256×256 px.

2.3.8 Directivity

In general a LED emits spatially and temporal coherent light to some degree; and so shows in both domains sharply peaked spectra. Spatially peaked means that—opposed to a point source emitting omnidirectional with the same intensity—the intensity of a LED source is depending on the view angle θ.

$$I_{led}(\theta) = I_{led}\ \Lambda(\theta) \tag{2.35}$$

where the attenuation Λ can be approximated roughly by a cosine function (Lambert's cosine law), depending on the illumination angle θ and the directivity δ:

$$\Lambda(\theta) \propto \cos\left(\frac{\theta}{\delta}\right) \tag{2.36}$$

In first approximation we may consider the SMD LED an isotropic point source, which results in a directivity δ of about 120 degrees FWHM (Lambert's cosine law), as shown in a polar chart in Figure 2.10: Full intensity at the centre, declining to 50% at 60 degrees off, to very little light outside of 60 degrees, thus constituting the directivity δ shown as dark green area.

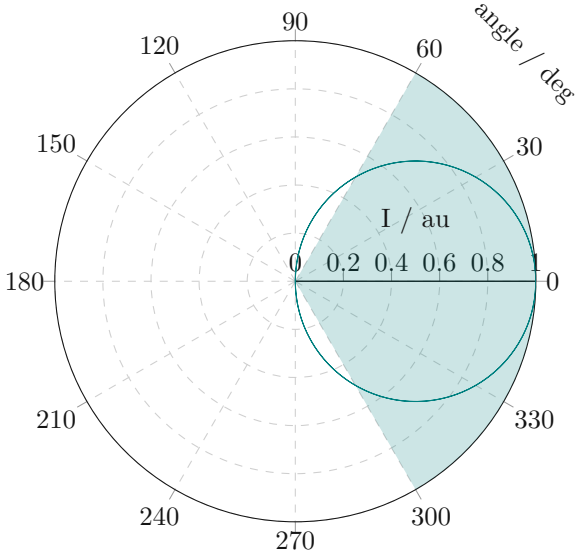


Figure 2.10: Schematics of the directivity of a single SMD LED

2.4 Convergence & Quality

The heart of FP is the iteration of the recursive Gerchberg–Saxton (GS) algorithm, which converges well on convex problems in infinite time [Fienup, 1982]. On first approximation our problem is convex enough [Zheng et al., 2013], but we still only have finite time to wait.

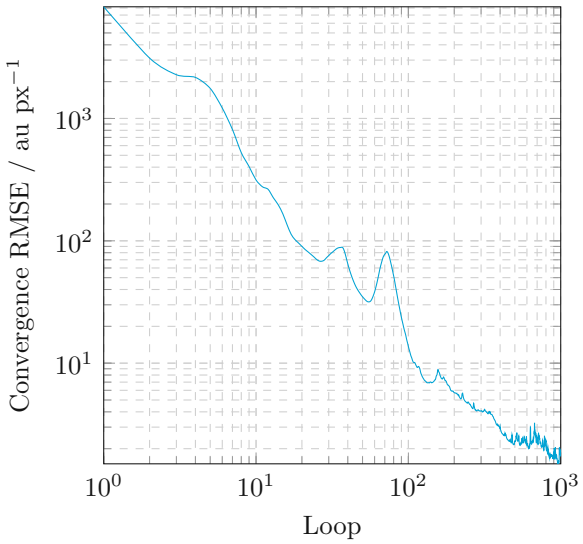


Figure 2.11: Convergences of FPM simulation reconstruction vs iterations (lower is better).

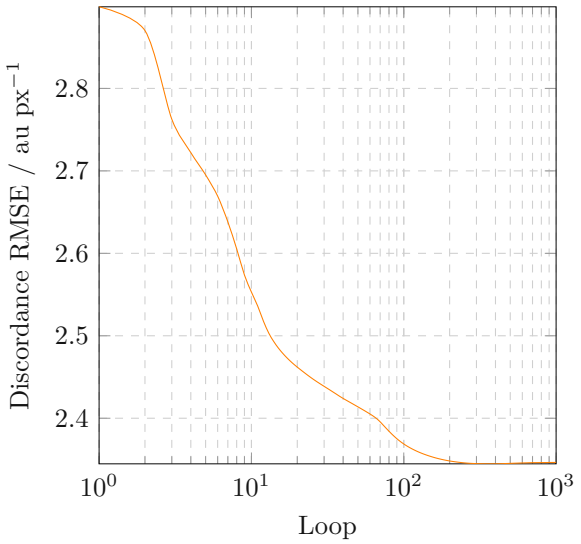


Figure 2.12: Discordances between FPM simulation reconstruction and low-resolution images vs iterations (lower is better).

As with all iterative algorithms it is crucial to define exit criteria. One first exit criterion will be the `loops` variable, that limits the number of overall iterations, and considers the result converged anyway. But how good actually is this result?

2.4.1 Convergence

Since recovering the unknown phase of a complex object leaves us no ground truth to compare the result with, we have to find other measures.

The first guess would be to compare the per-step changes to the complex object, as per Equation 2.16, described in detail in Section 2.1. The obtained `convergence` can and will be used two ways: Firstly, it serves as a second exit criterion for the recursive algorithm, to stop when the improvements obtained by further iteration get negligible (regardless if the maximum number of iterations has been reached or not).

Secondly, plotting `convergence` over iterations after the algorithm stopped (due reaching the limits of either `loops` or `eps`), allows us to estimate if the algorithm already converged or might have been stopped to early.

The `convergence` of an exemplary simulation is shown in Figure 2.11, where it follows Equation 1.21; it decreases continuously. But since this is always true, this fact alone makes a very bad marker for general recovery quality.

Consulting the speed of convergence (its derivation) is not overly helpful either, as our simulations show that both initial and finite convergence, as well as its speed, differ vastly between samples.

Additionally, [Fienup, 1982] showed, that plateaus are quite common in the convergence of the GS algorithm, so it would not be wise to abort the recovery, on the grounds of only small changes to the complex object alone.

We thus use a very conservative convergence exit criterion `ceps` = 1e0 / px.

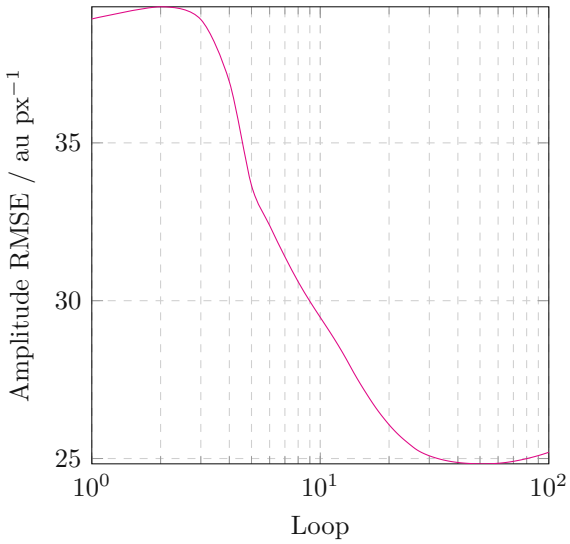


Figure 2.13: Quality of FPM simulation reconstructed amplitude vs iterations (lower is better).

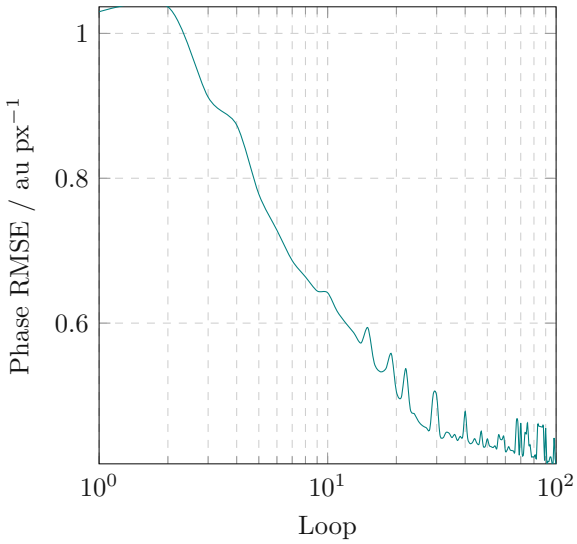


Figure 2.14: Quality of FPM simulation reconstructed phase vs iterations (lower is better).

2.4.2 Discordance

Similarly one might compare the amplitude of the low-resolution estimate $|o_n|$ of the n-th iteration with the amplitude of its respective low-resolution image a_n, summing the RMS error as per Equation 2.14, described in detail in Section 2.1. The so constructed and for the sake of simplicity termed `discordance`, has been shown by [Fienup, 1982] to closely fallow the `convergence`.

One exemplary case is shown in Figure 2.12; where the `discordance` is continuously decreasing with iterations, as anticipated by Equation 1.21.

Note that the `discordance` only measures the changes of the estimated amplitudes, ignoring the respective phases, so it is not the best marker for the quality of the whole reconstruction. Nevertheless we define a third exit criterion `aeps`, halting the iteration upon reaching. This limit is quite conservatively set to `aeps` = 1e0 / px.

2.4.3 Amplitude Quality

In simulations it might seem as if we were able to readily denote the quality of the reconstructed object: Every step during recovery, we can compare the complex object with the ground truth, summing over the root mean square (rms) of the differences. For the amplitude this `aquality` is easily interpreted, as it is real-valued: The lower the better. An example of a simulated recovery is shown in Figure 2.13.

Since the amplitude is only one side of the medal (the other being the phase), `aquality` alone does not tell a lot about the actual quality of recovery. So just because `aquality` is decreasing with iterations, as it is in this example, the phase might actually be improving at the same time; thus the overall quality might still be rising!

2.4.4 Phase Quality

Even worse, we found the RMS error of the phase to be not particularly useful at all. This seems puzzling first, but might actually be very

simple.

The phase space spans 2π, and it is cyclic, this means there is no *north* or *south* in phase space; its orientation is arbitrary. If one were to shift the whole phase image by adding a scalar, let's say π; qualitatively nothing happens. Quite on the contrary, if we are comparing the phase image with its ground truth, and then shift the whole phase image, the RMS error changes even if there is no qualitative change in physical properties!

In Figure 2.11 we can see the RMS error `pquality` of the phase of an exemplary simulated recovery increasing with iterations. This might, but does not have to be, a sign of divergence.

In conclusion, we will only use RMS error `convergence` over iterations as an estimator of overall quality of both experiment and simulation.

2.5 Confidence

Since FP recovers an unknown phase on a sophisticated use of amplitude information, we run into trouble where the object is black (near zero transmittance). Regardless of the origin of the light the amplitude at the detector is now equally zero. Starting with an arbitrary phase, but unable to improve due to lacking information, the GS algorithm thus returns an arbitrary phase.

Similarly for under- and overexposed pixels, including hot pixels of the sensor. In these cases the information is lost at the sensor, with identical consequences for the recovery.

Considering the fact that these hot/cold pixels do not contribute to the recovery, one might consider not using their information during the recovery at all! This is done throughout all the operations we describe in this thesis via a logical mask for each of the low-resolution images separately. One can optionally disable this by setting `sparsing` to `false`. The parameters for hot pixels are empirically set to, `hot = 245 uint8`, respective, `cold = 10 uint8`, where a unsigned integer (uint) n is defined as:

$$0 \leq n \leq 255, \quad n \in \mathbb{N} \tag{2.37}$$

2.5.1 Sparsely Sampling

This exact process is also proposed by [Zheng, 2016], terming it *sparsely sampling*, if only on very different grounds. There, the goal is to improve the quality of the recovery. This can evidently be done by combining each of the low-resolution-images by multiple ones, taken at different exposures, to one high-dynamic-range (HDR) image. Obviously this means a manifold of the picture taking process. Alternatively [Zheng, 2016] showed sparsely sampling to achieve similar effects, without raising the number of pictures to be taken, and with a negligible numerical cost of one matrix multiplication, generally even with a sparse matrix.

But considering sparsely sampling works on a low-resolution-image level, we would propose that the combination of HDR with sparsely sampling would still be a significant—if somewhat costly— improvement.

2.5.2 Amplitude Confidence

It becomes obvious, that the GS algorithm fails at recovering the amplitude at positions where the phase has sharp edges. In order to show this, we simulate using the most extreme cases; binary (black and white) images resembling a chess board and chess pieces, as shown in Figure 2.15. The amplitude is well reconstructed overall (chess board); except it shows fine lines where there should be none, at places where there is a sharp edge in the phase image (right).

This is shown for the amplitude in the bottom row (left), where white denotes a high con-

fidence in the reconstruction and black indicates low confidence.

Here we used the edge filter edge from Octave Forge's package *image*, with the method *Canny*, to generate the amplitude confidence map.

In all the simulations and recoveries throughout this thesis, this confidence map is used to evaluate the quality of the reconstruction, and are discussed where necessary.

2.5.3 Phase Confidence

Whatever the reasons why one ends up with a reconstructed phase image which is in parts arbitrary, it would be the least to know where the recovery most certainly did not work. The most convenient way to achieve this is a phase confidence map created at the end of the recovery, highlighting near black areas of the recovered amplitude.

Throughout all the simulations and recoveries we use a lower threshold of 3 uint8 to mark dark areas, in order to get an idea of the confidence of the recovered phase, as shown in Figure 2.15. The phase image is well reconstructed only at the white tiles of the chess board, and it fails completely in the black tiles; leading to patches of arbitrary phase.

This is shown for the phase in the bottom row (right), where white denotes a high confidence in the reconstruction and black indicates low confidence.

In all the simulations and recoveries throughout this thesis, this confidence map is used to evaluate the quality of the reconstruction, and are discussed where necessary.

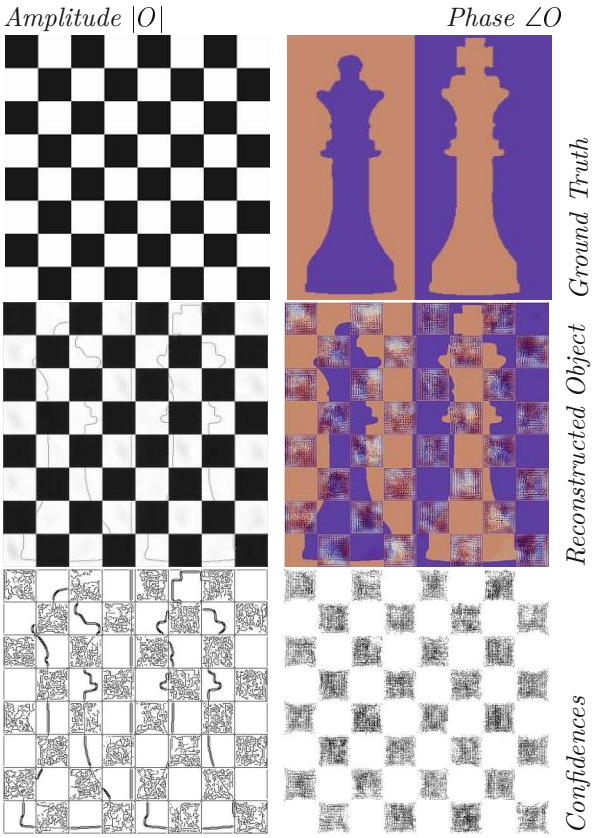


Figure 2.15: Comparison of the ground truth (top row), with the reconstructed object (middle row), and the confidence maps (bottom row); in all cases amplitude left, phase right. Both the amplitude and phase of the ground truth are binary (0 uint8 and 255 uint8, respective 0 rad and π rad), as are the confidence maps (bottom row).

2.6 Alignment

The scope of this work is to illuminate the robustness of FPM to variation of the critical system parameters. Based on the GS algorithm, FPM uses multiple images taken from different illumination angles, so recovery obviously relies on the knowledge of the illumination positions, but to what degree? One objective of this thesis is to investigate the six degrees of freedom (DOF) of the illumination setup for

its respective impact on the subsequent FP recovery.

Euler Angles

Lets first define the used nomenclature of the axes and the Euler angles. A visualisation of this shown in Figure 2.16, for the rotation operation:

$$f(\rho, \psi, \gamma) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow (\mathbf{x}', \mathbf{y}', \mathbf{z}') \qquad (2.38)$$

where a rotation around the z-axis is denoted as γ, and will be called *yaw* throughout this thesis. Likewise ρ, ψ denote rotations about the x-axis respective the y-axis; and will be called *roll*, respective *pitch*.

The corresponding translation operation, called *shift*, is not shown graphically, due to its simplicity of just shifting along the respective axis:

$$f(\Delta x, \Delta y, \Delta z) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow (\mathbf{x}', \mathbf{y}', \mathbf{z}') \quad (2.39)$$

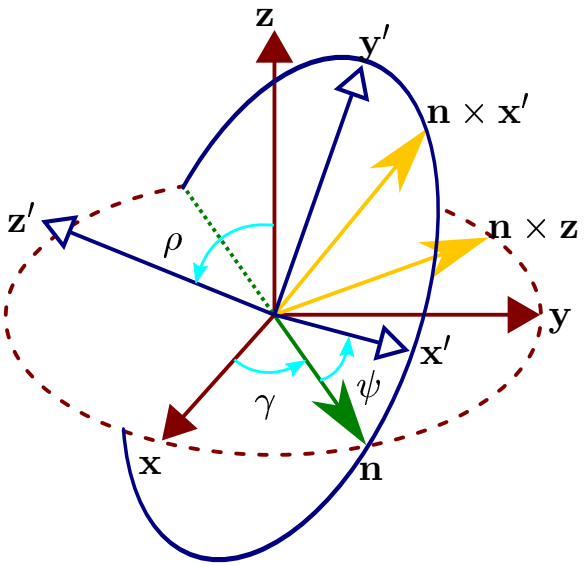


Figure 2.16: Schematics of the used nomenclature of the Euler angles for rotations roll ρ, pitch ψ and yaw γ around the respective axes of the orthonormal base (x,y,z).

2.6.1 Alignment Simulation

Simulating FPM recovery is quite a straight forward task, as the concept is basically embedded in the FPM algorithm itself. The procedure of this simulation is shown in Algorithm 2, which closely follows the nomenclature from Algorithm 1. An abbreviated version of the code is shown in Section B.2.2 of Appendix B; the full code is online available [Siegel, 2021a]).

Multiple images are rendered as if from various illumination angles $\mathbf{k}_n$, and so comprise different regimes in spatial frequency space of the complex object. So if one happened to know the complex object (amplitude and phase) beforehand (Line 2), its Fourier transform could be decomposed into a stack of simulated low-resolution amplitudes a_n (Lines 7–8)!

These stacks are generated step wise (Line 4) with respect to the system parameter of interest, e.g. lateral shift along the x-axis (Line 5), each time recovered using the standard FPM algorithm (Line 9).

Quite on the contrary to actual experiments, in simulations like this, we are able to readily denote the quality of the reconstruction: Every step during recovery, we can compare the complex object with the ground truth, summing over the root mean square error (RMSE) of the differences.

For the simulation of misaligned FPM recovery, the amplitudes a_n of a stack of simulated low-resolution images ι_n are first induced with noise of magnitude α_m in the following way:

The k-space positions $\mathbf{k}_n$ corresponding to the locations $\mathbf{r}_n$ of the physical LEDs in real space are defined according to Algorithm 3. An abbreviated version of the code is shown in Section B.1.3 of Appendix B; the full code is online

Algorithm 2 Alignment Simulation

1: **procedure** ALIGNING(A, Φ, m)
2: $O \leftarrow A\,\mathrm{e}^{\mathrm{i}\Phi}$
3: $\hat{O} \leftarrow \mathcal{F}(O)$
4: **for all** m misalignments α_m **do**
5: $\mathbf{k}_n \leftarrow$ KRID($\Delta x, \Delta y, \Delta z, \rho, \psi, \gamma$)
6: **for all** $\mathbf{k}_n$ **do**
7: $\hat{o}_n \leftarrow (\frac{\zeta}{\eta})^2\, \hat{O}_{\mathbf{k}-\mathbf{k}_n}\, \hat{P}$
8: $a_n \leftarrow \mathcal{F}^{-1}(\hat{o}_n)$
9: $O_m \leftarrow$ REKOVER(a)

available [Siegel, 2021a]).

To simulate misalignment of all six degrees of freedom, the LED's positions $\mathbf{r}_n$ are first rotated about the x-,y-, and z-axis (respective rotation matrices R_ρ, R_ψ, R_γ in Line 2). The rotated grid $\mathbf{r}$ is then shifted in x-,y-, and z-axis (Line 3) and mapped to the two-dimensional sensor plane (Line 4) in Fourier space, where $k_0 = 2\pi/\lambda$ denotes the spatial frequency of the incident light.

Algorithm 3 K-space Grid Variation

1: **function** KRID($\Delta x, \Delta y, \Delta z, \rho, \psi, \gamma$)
2: $\mathbf{r} \leftarrow R_\gamma R_\psi R_\rho \mathbf{r}$
3: $\mathbf{r} \leftarrow \mathbf{r} + \Delta x + \Delta y + \Delta z$
4: $\mathbf{k} \leftarrow -k_0 \sin\left(\arctan\left(\mathbf{r}_x./\mathbf{r}_z\right)\right)$

2.6.2 Alignment Calibration

A structural downside of FPM is that the illumination source is off-focus, so its alignment is—though crucial—not directly measurable.

The current state of the art is based on simulated annealing [Zhou et al., 2018], nested inside the FP routine. This poses two key problems: Simulated annealing crucially depends on a quite accurate initial guess of several parameters, furthermore this method is quite expensive (it takes a long computation time).

To circumvent these, [Eckert et al., 2018] proposed a surprisingly simple yet beautiful calibration algorithm using autocorrelation to correct for the positions—at least for the images taken under bright field conditions.

As icing on the cake, during the process this method necessarily calibrates the magnification of the optical system, which is usually known only to some degree. Based on [Eckert et al., 2018], the proposed method is a standalone calibration: Introducing a single step running prior to, and completely detached of, the standard FPM routine!

Bright-Field Calibration

In the experimental FPM setup, the camera records intensity images ι_n, where n indicates the illumination direction θ_n of a coherent plane wave which corresponds to the spatial frequency $\mathbf{k}_n$ in the high-resolution spectrum $\hat{O}$. Its two-dimensional FT $\hat{\iota}_n$, which according to Equation 1.1 is:

$$\hat{\iota}_n = \mathcal{F}\{\iota_n\} = \mathcal{F}\{|a_n|^2\} \tag{2.40}$$

where a_n denotes the amplitude of the object. Using the cross-correlation theorem (Equation 1.12), this spectrum amounts to:

$$\hat{\iota}_n = \hat{a}_n \star \hat{a}_n \tag{2.41}$$

where $(\star)$ denotes autocorrelation as a special case of cross-correlation shown in section 1.1.6. the FT $\hat{\iota}_n$ of n-th discretely sampled intensity image captured by the camera can be modelled according to Equation 2.1, and derived in detail in section 2.1 to:

$$\hat{\iota}_n \mathrel{\widehat{=}} (\hat{P} * \hat{O}_{\mathbf{k}-\mathbf{k}_n}) \star (\hat{P} * \hat{O}_{\mathbf{k}-\mathbf{k}_n}) \tag{2.42}$$

The term $\hat{P} * \hat{O}_{\mathbf{k}-\mathbf{k}_n}$ corresponds to the spectral object, shifted about $\mathbf{k}_n$, and convolved with the spectral pupil function. This creates a disk of radius k_{co} at the position $\mathbf{k}_n$, consisting of the spectral values of the object where $\mathbf{k} < k_{co}$, and zero outside. Typically a Fourier

spectrum decays sharply from the zero-order term of Fourier spectrum (DC) term—the average (arithmetic mean) of the entire image, located at frequency zero, thus at the centre—toward higher frequencies, usually about a few orders of magnitude.

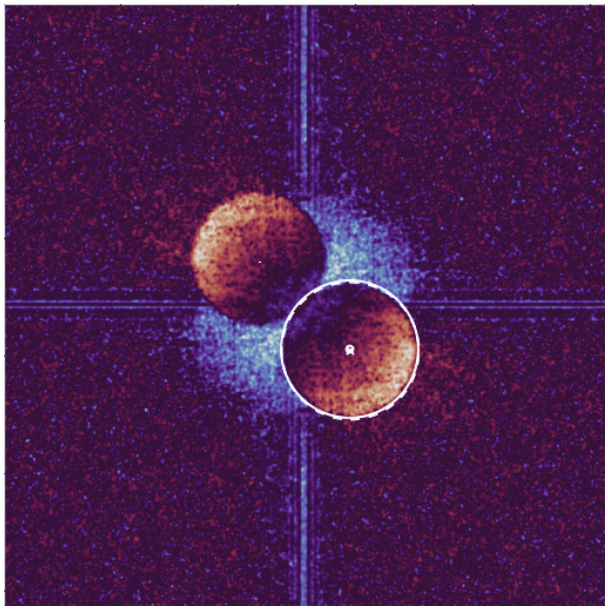


Figure 2.17: magnitude $|\hat{a}_n|$ of the spectrum of one exemplary low-resolution image, assumed (dashed circle) and corrected position (circle).

The autocorrelation operation ($\star$) now correlates mentioned disk with a conjugate copy of itself; effectively scanning the DC term of one disk over the pupil $\hat{P}$ of its conjugate. Coherently summing at each pixel to form $\hat{\imath}_n$, leads to high values where the pupil overlaps the DC term, and negligible outside.

Only when the image $\imath_n$ is illuminated under BF conditions, the DC term of the object spectrum $\hat{O}(\mathbf{k}-\mathbf{k}_n)$ is located within its pupil's pass band $\hat{P}$; leading to high values where the pupil overlaps the DC term, and negligible outside.

The amplitude $\hat{a}_n$ of the spectral image $\hat{\imath}_n$ now shows two distinct disks located at $\mathbf{k}_n$ and $-\mathbf{k}_n$, exemplary shown in Figure 2.17, along with the assumed position (dashed circle), and its correction (circle).

Circle Detection

The position correction problem now shifted to a much simpler image recognition problem! As [Eckert et al., 2018] show, one simply needs to fit two circles to the autocorrelation disks of each of the spectra $\hat{\imath}_n$, to find the illumination positions $\mathbf{k}_n$ and $-\mathbf{k}_n$.

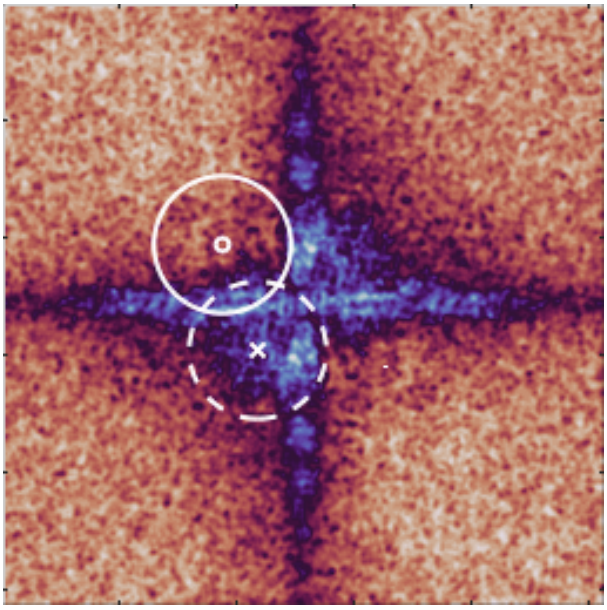


Figure 2.18: magnitude $|\hat{a}_n|$ of the spectrum of one exemplary low-resolution image just outside the BF area, with the assumed position (dashed circle) and flawed *corrected position (circle). Note that there are no autocorrelation disks visible.*

Note that outside the BF region, as shown exemplary in Figure 2.18 the autocorrelation does not overlap the DC term; there are no distinct circles, thus no position correction!

Yet the proposed algorithm suggests the existence of a disk, which has to be evaluated manually, and in this case quite obviously unreliable.

Algorithm 4 Alignment Calibration

1: $\hat{\imath}_n \leftarrow \mathcal{F}\{\imath_n\}$
2: $\hat{\imath}_n \leftarrow |\hat{\imath}_n| / \text{mean}_n(|\hat{\imath}_n|)$
3: $\hat{\imath}_n \leftarrow \log_{10}(\hat{\imath}_n)$
4: $\hat{\imath}_n \leftarrow \text{gauss}(\hat{\imath}_n), \sigma)$
5: **for all** m $\subset$ n images **do**
6: $\quad r_m \leftarrow \text{imfindcircles}(r \pm \Delta r,\ \epsilon)$
7: $r \leftarrow \text{median}_m(r_m)$
8: **for all** n images **do**
9: $\quad \mathbf{c}_1, \mathbf{c}_2 \leftarrow \text{imfindcircles}(r)$
10: $\quad \mathbf{c}_n \leftarrow \text{argmin}_{1,2}(|\mathbf{c}_{1,2} - \mathbf{k}_n|)$

The procedure is shown in Algorithm 4, the corresponding lines are shown in brackets. An abbreviated version of the code is shown in Section B.2.1 of Appendix B; the full code is online available [Siegel, 2021a]).

For a set of n low-resolution images $\imath_n$, the two-dimensional FT is computed via Equation 1.7, using FFT routines (Line 1).

The magnitude of these spectra is divided by their mean, to filter out the information about the sample, which we are not interested right now (Line 2).

The magnitude of the Fourier spectra usually span some orders of magnitude, even after the DC term is lost, so we take the logarithm (Line 3).

Since the following function for the circle detection, `imfindcircles` is part of the `image` package in Octave Forge. Being based on a Hough transform—alas on edge detection—it wants the background as smooth as possible. So we blur the image (Gaussian blur), where $\sigma = 2$ px was found empirically to work well (Line 4).

Since the Hough transform tests all possible circles, its speed heavily relies on the knowledge of the radii, which are in the case of FPM all the same (pupil function in Equation 2.3).

So we first run `imfindcircles` on a subset $m \subset n$ with our initial guess for the radius (the cutoff frequency in Equation 2.4) and some reasonable deviation, subsequently taking the median of the n best radii (Line 7).

Using this estimated radius to initialize the Hough transform, we search for the two best circles in all n images (Line 9), since all the low-resolution images contain two autocorrelation discs (at $\mathbf{k}_n$ and $-\mathbf{k}_n$), due to the symmetry of the problem.

Based on our initial positions $\mathbf{k}_n$, we take the one nearest of the two $\mathbf{c}_{1,2}$, in order to get all the different positions once (Line 10), leaving us with n corrected positions $\mathbf{c}_n$.

This calibration has to be performed only once for a given microscope setup, subsequent FPM recovery operations can all use the corrected BF positions $\mathbf{c}_n$, as shown exemplary for one experimental setup and three simulations in Figure 3.11.

Magnification Calibration

The shown alignment calibration method relies on a somewhat accurate initial guess of the radius r of the autocorrelation disc in Fourier space. For the circle detection to work out, we need to establish a relation for the radius r, based on our knowledge about the optical system. As a side-effect the alignment calibration process necessarily calibrates the magnification of the optical system [Eckert et al., 2018].

The magnification mag of an optical system, where an object with size l_{fov} is captured by a camera with sensor pixel-size l_{ccd}, may be described as:

$$mag = \frac{l_{fov}}{l_{ccd}} \tag{2.43}$$

Essentially the lowest possible frequency ν_{min} the optical setup is able to catch, is one

mode across the whole image space; which relates to the highest possible spatial frequency k_{max}:

$$k_{max} = \pi l_{fov} = \pi \frac{mag}{l_{ccd}} \tag{2.44}$$

alas the Fourier space is scaled inverse proportionally to real space, and spans:

$$\mathcal{F} : \{-k_{max}, -k_{max} + \delta k, \ldots, k_{max} - \delta k, k_{max}\} \tag{2.45}$$

where δk is the discreet step-width in Fourier space, defined by the resolution (amount of pixels) of the sensor n_{ccd}:

$$\delta k = \frac{2k_{max}}{n_{ccd}} \tag{2.46}$$

Now the radius r of the autocorrelation disc in Fourier space in pixels corresponds to the spatial cutoff frequency k_{co} of the pupil function (see Equation 2.5):

$$r = \frac{k_{co}}{\delta k} = \frac{\mathrm{NA}\, k_0}{\delta k} = \frac{2\pi \mathrm{NA}}{\lambda_0 \delta k} \tag{2.47}$$

Combining these three equations, one is able to obtain a relation for the magnification mag, that depends only on directly measurable variables, and so represents a possibility for calibration:

$$mag \approx \frac{\mathrm{NA} n_{ccd} l_{ccd}}{\lambda_0 r} \tag{2.48}$$

where n_{ccd} denotes the size of the low-resolution image ι in pixels; l_{ccd} the pixel-size of the sensor in meters; and the radius of autocorrelation disc in Fourier space, also in pixels.

2.6.3 Extrapolation to Dark-Field

We would advise against fully automatising alignment correction via extrapolation to the dark-field (DF) based on BF correction of section 2.6.2. The FPM spectra pose a hard problem for edge detection based Hough transforms—due to high speckle noise of the spectra etc—which led [Eckert et al., 2018] to

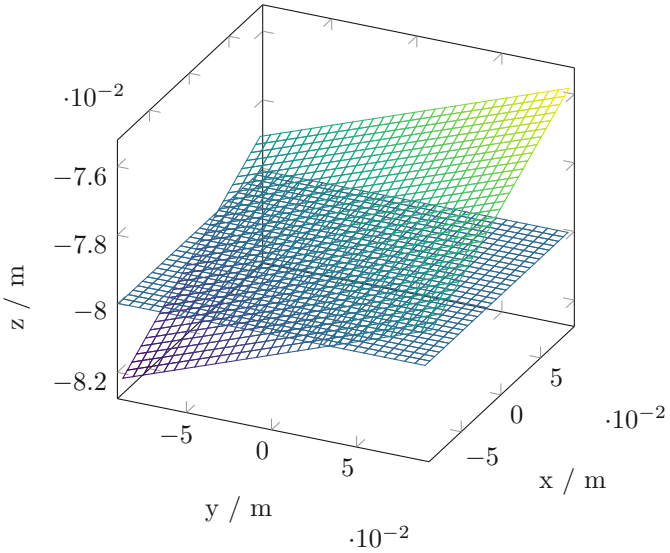


Figure 2.19: LED positions (nodes) of a slightly misaligned panel in 3D real space.

a random sample consensus (RANSAC) based automated outlier detection. For correcting fine displacements of the individual LEDs of a well positioned LED panel, this might be useful, but would classify a bad aligned panel en gros as outliers!

Additionally, if one takes into account at least yaw (if not tilt and roll) of the panel, the mentioned RANSAC method completely fails. Such an slightly misaligned LED panel is exemplary shown in Figure 2.19. In this case the LED panel with 6 mm spaced LEDs is positioned at a height of −80 mm (under the target), with a shift of 1 mm in x-, y-, z-axis, and 1° tilt, roll, and yaw, in respect to the centre.

The corresponding k-space grid is shown in Figure 2.20. Fitting the corrections $\mathbf{c}_n$ to this grid in particular, and to an even slightly misaligned panel in general, projected to a strict 2D grid in k-space, might just be a task too complex.

Manual inspection of the correction positions on the contrary, might reveal some systematic misalignment, for example a displace-

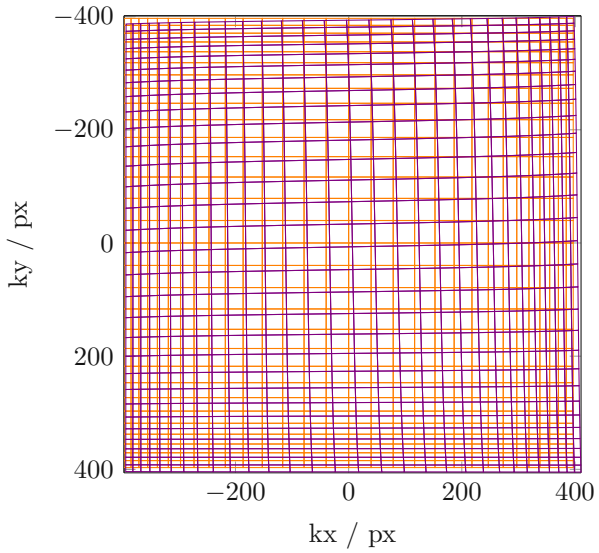


Figure 2.20: Comparison of the k-space projections $\boldsymbol{k}_n$ (nodes) of the LED positions of a slightly misaligned panel (violet) shown in Figure 2.19; versus the initial panel (orange).

ment (shift) of the whole LED panel. In such a case it is convenient to correct the whole array, based on the mean, median and standard derivation of the BF corrections.

2.7 Illumination

The most convenient and frequently used setup for the illumination is a shift-register controlled RGB LED matrix, made for displays in urban spaces. This makes it a cheap and easy to implement solution proven to work [Ou et al., 2013], in spite of not all the LEDs being equidistant to the target, and orientation of the individual LEDs. This work now inquires the impact of the resulting brightness variation on the FPM reconstruction.

2.7.1 Luminosity Simulation

The simulation of inconsistently illuminated FPM recovery is based on the general FPM simulation, as detailed in Section 2.6 for misalignment. Instead of the position, now the overall brightness of the resulting low-resolution images are altered: The amplitudes a_n of this stack of simulated low-resolution images ι_n are first induced with noise of magnitude ν (shown in Figure 2.22) in the following way:

$$a'_n = a_n + \nu s \tag{2.49}$$

where s denotes a pseudo-random variable,

$$s \in \mathcal{N}(\mu = 0, \sigma^2 = 1) \tag{2.50}$$

where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2, who's probability density function $f(\xi)$ (shown in Figure 2.21) is given by:

$$f(\xi) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{\xi-\mu}{\sigma}\right)^2} \tag{2.51}$$

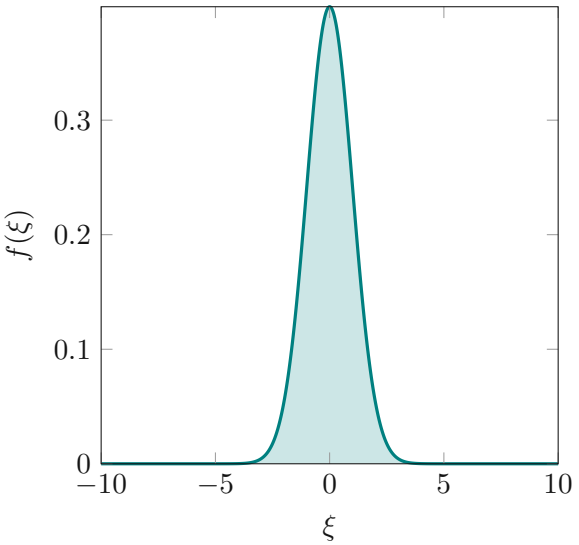


Figure 2.21: Probability density function $f(\xi)$ normal distribution of Equation 2.50.

Optionally the amplitudes a_n are then attenuated in respect to their illumination angle θ_n according to a given directivity δ_n from Equation 2.36:

$$a''_n = a'_n \Lambda(\theta_n) \tag{2.52}$$

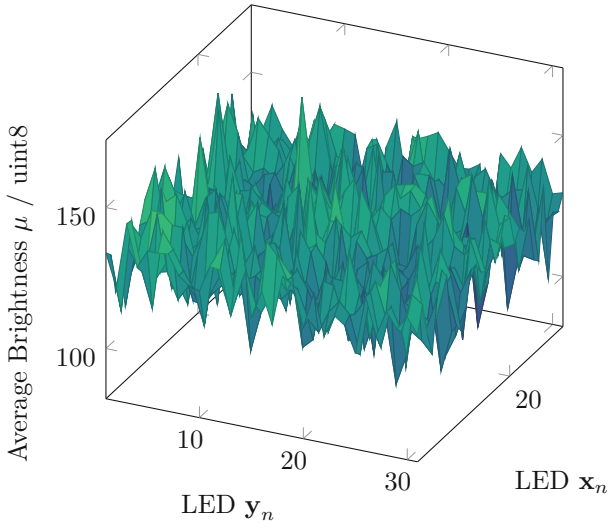


Figure 2.22: Average brightness μ of the noisy amplitudes a'_n, simulating inconsistent illumination from the LED positions $\boldsymbol{x}_n$,$\boldsymbol{y}_n$ (nodes); featuring noise νs.

Algorithm 5 Luminosity Simulation

1: **procedure** ALIGHT(A, Φ, ν_m)
2: $O \leftarrow A\,\mathrm{e}^{\mathrm{i}\Phi}$
3: $\hat{O} \leftarrow \mathcal{F}(O)$
4: $\mathbf{k}_n \leftarrow$ KRID( )
5: **for all** m noise magnitudes ν_m **do**
6: **for all** $\mathbf{k}_n$ **do**
7: $\hat{o}_n \leftarrow (\frac{\zeta}{\eta})^2\,\hat{O}_{\mathbf{k}-\mathbf{k}_n}\,\hat{P}$
8: $a_n \leftarrow \mathcal{F}^{-1}(\hat{o}_n)$
9: $a_n \leftarrow a_n + \nu_m\,\mathrm{randn}()$
10: **if** directivity **then**
11: $a_n \leftarrow a_n \cos(\theta\frac{120}{\delta})$
12: $\mu_n \leftarrow \mathrm{mean}(a_n)$
13: $\sigma_n \leftarrow \mathrm{std}(a_n)$
14: $\hat{O} \leftarrow$ REKOVER(a)

The procedure of this simulation is shown in Algorithm 5, the corresponding lines are shown in brackets. An abbreviated version of the code is shown in Section B.4.1 of Appendix B; the full code is online available [Siegel, 2021a]).

A set of arbitrary test-amplitudes is generated assuming perfect alignment (Lines 2–4). For each of the noise magnitudes ν_m (Line 5), this stack of low-resolution amplitudes a_n is now altered—either attenuating only (Line 9), or additionally accounting for directivity δ (Line 11)— each time recovered using the standard FPM algorithm (Line 14).

The attenuation factor $\Lambda(\theta_n)$ for directivity $\delta = 40°$ is shown for all n illumination positions in Figure 2.23.

Combining noise and attenuation due to directivity according to Equation 2.52, one obtains a simulation of inconsistent illumination, shown exemplary in Figure 2.24.

Similarly to simulating misalignment we are able to readily denote the quality of the reconstruction: Every step during recovery, we can compare the complex object with the ground truth, summing over the root mean square error (RMSE) of the differences.

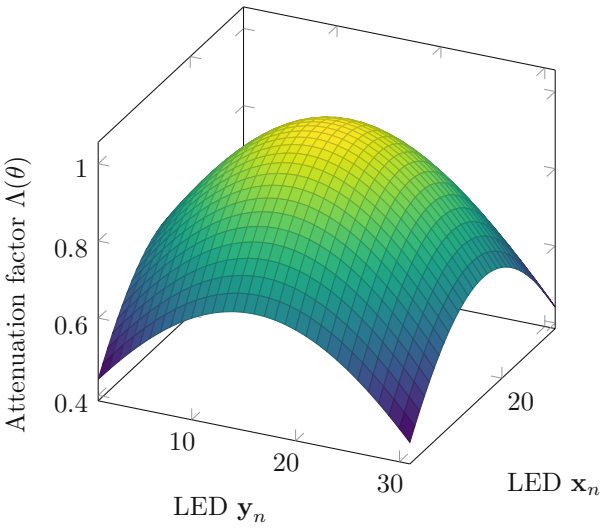


Figure 2.23: Attenuation factor $\Lambda(\theta)$ due to directivity $\delta = 40°$.

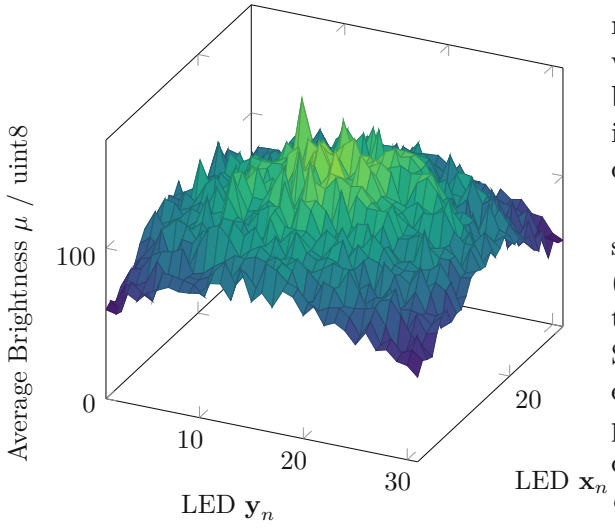


Figure 2.24: Average brightness μ of the amplitudes a_n simulating inconsistent illumination from the LED positions $\boldsymbol{x}_n$,$\boldsymbol{y}_n$ (nodes); attenuation $\Lambda(\theta)$ due to directivity δ and noise νs.

2.7.2 Luminosity Calibration

Assuming the LEDs are all equally bright (we can only hope so), we still do not end up with equal intensity at the sample. This is because the LEDs are assembled to the flat surface of the panel shining upwards. Due to the directivity described above, the intensity of the individual LEDs vary with the view angle; less and less light arrives at the sample, the farther the LEDs are located from the central position.

This issue has been investigated before [Zhang et al., 2019], where it is solved using simulated annealing embedded in the recovery process, with the downside, that it significantly prolongs the recovery time.

In order to compensate for the inconsistent illumination of the LED panel, we therefore propose a very simple, yet effective calibration step using a diffuser target. This calibration has to be done only once for the physical setup, and can be used for all subsequent measurements. In the following description we will refer to the lines of code in Algorithm 6 in brackets. An abbreviated version of the code is shown in Section B.4 of Appendix B; the full code is online available [Siegel, 2021a]).

First, all images $\boldsymbol{\iota}$ (this time the raw intensity images are used!) are loaded and analysed (Lines 2–4), where the bold notation indicates the vectorial property of this stack of images. Subsequently, the mean intensity value μ_n of each image (mean over the pixel values) is compared to the mean intensity of *all* images $\boldsymbol{\mu}$; confounding the *luminosity correction coefficients* Λ_n for each image (Line 5). This coefficients are exported as a vector $\boldsymbol{\Lambda}$ (Line 6), which can be used to normalise all further measurements captured with this physical setup.

The arithmetic mean is preferred here over the median, because the LED brightness variations are only very subtle (few %), and no far outliers are to be expected.

Algorithm 6 Luminosity Calibration

```
1: function KalibRight( ι )
2:     for all n images ι_n do
3:         μ_n ← mean(ι_n)
4:         σ_n ← std(ι_n)
5:     Λ_n ← mean_n(μ)/μ_n
6:     return ( Λ )
```

Targetless

For the BF region, the calibration of the LED luminosity is straight forward: The FP microscope is equipped with *no target* at all; which yields pure white images in the BF area, and dark images outside.

Mind that the resulting $\boldsymbol{\Lambda}_b$ are only to be used to normalise BF images; for a full set including dark field images, follow the approach in the following paragraph.

Diffused

The generalisation of this calibration process for all images, BF and DF alike, requires a slight modification of the procedure described above. Since when not using any target, the resulting DF images are pitch black; we need some sort of semi-transparent and diffuse target, to scatter some light into the microscope.

Fortunately we accidentally came up with a dedicated diffuser plate—which was initially sandwiched in the backlights of a comercial smartphone LCD.

Chapter 3

Results & Discussion

To investigate the robustness of Fourier ptychographic imaging to various system parameters, we performed simulations, for a variety of circumstances: Different simulated amplitude and phase *samples*, misalignment of all six degrees of freedom (Section 2.6), various levels of misalignment, jointly recovering the pupil function, based on the initial scheme [Zheng et al., 2013], sparse sampling (Section 2.5.1), noise, brightness variations (Section 2.7), different image dimensions (Section 2.3.7).

Discussing all these findings in detail would go beyond the scope of this thesis, so aside from the basics (Section 3.1) we focus on the most imported (Alignment Section 3.2) and the novelties (Illumination Section 3.3).

Parts of this Chapter (Section 3.2) were presented at the Electronic Imaging Conference 2021 (EI21), held by the Society for Imaging Science & Technology (IS&T); leading to publication in the conference proceedings [Siegel et al., 2021].

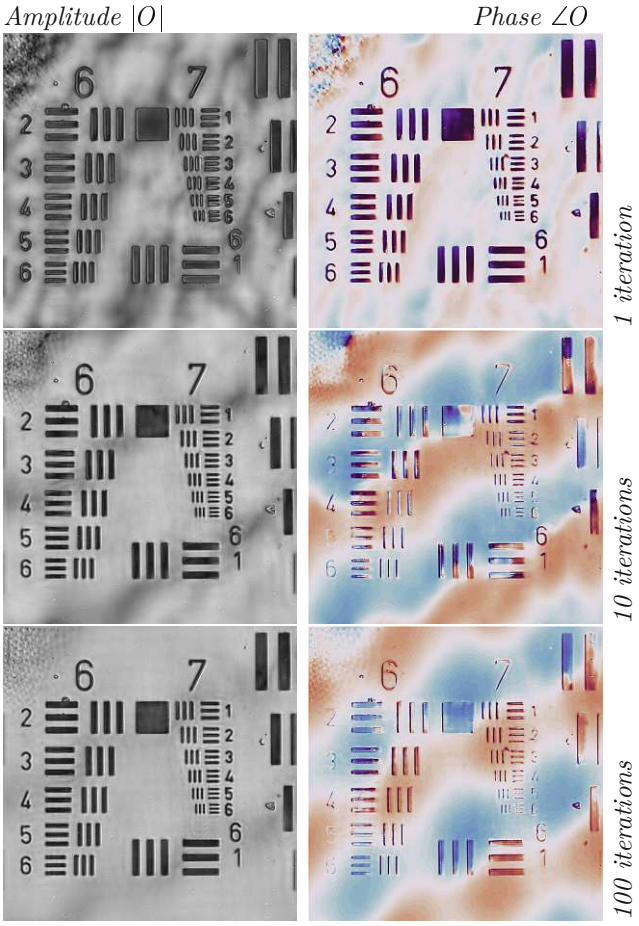


Figure 3.1: Comparison of both the amplitude (left column) and the phase (right column) of the recovered object at $\{1, 10, 100\}$ iterations. One can clearly see the amplitude improving.

3.1 Proof of Concept

Before we dive into the various aspects in detail, let us analyse FPM in general. In this section we discuss the proof of concept, the operation and behaviour of our prototype Fourier ptychographic microscope, built as described in detail in Section 2.1.

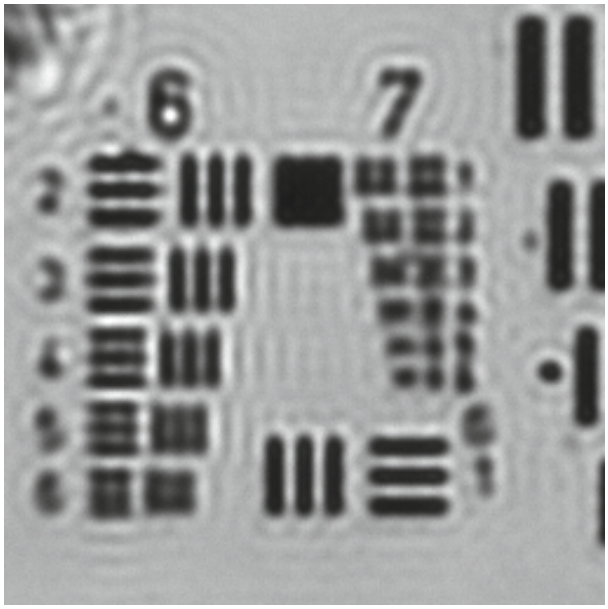


Figure 3.2: The best of the low-resolution images, around 350×350 µm.

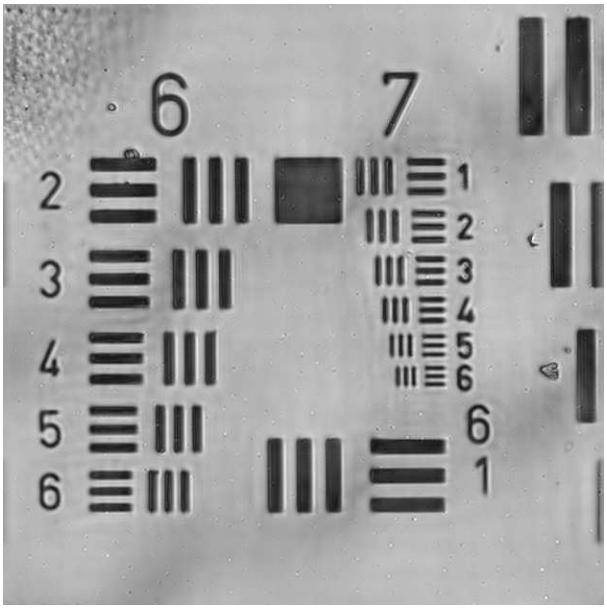


Figure 3.3: FPM recovered amplitude after 100 iterations, around 350×350 µm.

We discuss the obtained resolution, recovered phase and the constructed synthetic aperture.

As an example for the step-wise convergence of the FPM algorithm, detailed in Section 2.1; *snapshots* of the recovered amplitude and phase are shown in Figure 3.1 at the three arbitrary iterations {1,10,100}. The initial guess is *zero* (blank images) both for amplitude and phase, and thus not shown.

3.1.1 Microscope Resolution

Based on a stack of 931 intensity images, some exemplary ones are shown in Figure 2.4, a FPM recovery of the complex object—a USAF1952 resolution target—was obtained; throughout this thesis called Experiment 1. The best of the low-resolution images, the one illuminated by one central LED, is shown in Figure 3.2.

The elements 2 and three (top left) of the Group 6 in Figure 3.2 are quite clearly to distingish, wheras the lines of Element 4 start to blur, at least Element 5 is unreadable.

According to Equation 2.29, this sets the resolution res_{exp} of the microscope to:

$$res_{exp} \approx 90.5\,\mathrm{lp/mm} \tag{3.1}$$

where the line pair means a black and a white line, and where the width of the respective black line amounts to:

$$d_{exp} \approx 5.52\,\mu\mathrm{m} \tag{3.2}$$

which is in good accordance to the theoretical resolution of the microscope (illuminated with light of the wavelength of $\lambda_{red} = 632$ nm) as given by Equation 2.21:

$$d_{theo} \approx 5.75\,\mu\mathrm{m} \tag{3.3}$$

3.1.2 Super Resolution

A basic FPM recovery on this stack of images reveals the amplitude shown in Figure 3.3, where all the elements up to Group 7, Element 6 are perfectly clear!

This sets the resolution res_{fpm} of the Fourier ptychographic microscope to at least:

$$res_{fpm} \approx 228.1\,\mathrm{lp/mm} \tag{3.4}$$

where the line pair means a black and a white line, and where the width of the respective black line amounts to:

$$d_{fpm} \approx 2.19\,\mu\mathrm{m} \tag{3.5}$$

which is below the experimental and theoretical resolution limit of the physical microscope, shown in Equation 3.2 and 3.3, thus FPM effectively obtained *super-resolution*.

It is quite reasonable, that the actual resolution of this FPM setup is even higher—yet the resolution target in question did not cover the range beyond Group 7.

In that sense we would see this resolution as an upper limit:

$$d_{fpm} < 2.2\,\mu\mathrm{m} \tag{3.6}$$

3.1.3 Synthetic Aperture

The spectrum of the reconstructed complex object, being stitched together from the low-resolution-image discs now forms a disc itself (depends on the k-space stitching, in this case almost a square); with spectral information within, and zero outside. The magnitude of the spectrum of the recovered complex object is shown in Figure 3.4.

The radius of said disk constitutes the cut-off frequency—and thus the synthetic aperture—of the reconstructed complex object, quite analogous to the coherent transfer function in Equation 2.5:

$$\mathrm{NA}_{fpm} = \frac{k_{fpm}}{k_0} \approx 0.30 \tag{3.7}$$

which is in good accordance to the theoretical synthetic aperture from Equation 2.27,

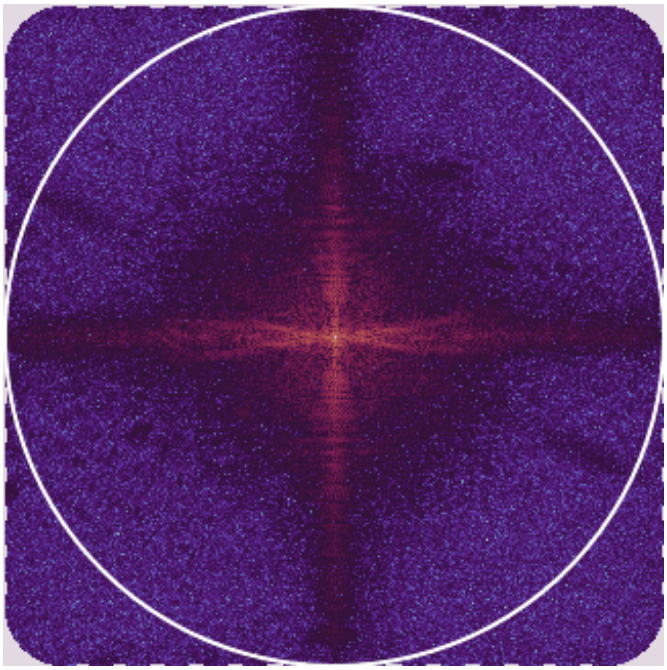


Figure 3.4: Magnitude of the FPM recovered spectrum, with cutoff frequency (circle).

and so our microscope effectively obtained *super aperture*.

The effective resolution limit of the FPM setup can be estimated using Equation 2.28:

$$d_{fpm} = \frac{\lambda_{red}}{2\ \mathrm{NA}_{fpm}} \approx 1050\,\mathrm{nm} \tag{3.8}$$

which is somewhat higher than the 990 nm we calculated as our theoretical FPM resolution from Equation 2.28, acting as a lower limit for the experimental resolution.

This confirms our expectation that our experimentally obtained resolution is actually higher than the 2.2 µm derived from the USAF target as shown in Equation 3.6.

Concluding we estimate the experimental FPM resolution to be:

$$2.2\,\mu\mathrm{m} > d_{fpm} > 1050\,\mathrm{nm} \tag{3.9}$$

3.1.4 Field of View

Unless noted the processed low-resolution images are a 256×256 px central part of the captured raw images. This corresponds to a field of view (FOV) of around 350×350 µm, which amounts to a captured area of interest (AOI) A_{256} of:

$$A_{256} \approx 0.12\,\mathrm{mm}^2 \tag{3.10}$$

On our machine, equipped with physical 8 GB RAM, we were able to process at best 441 low-resolution images of the size 1024×1024 px. Each of these images weighs only 100 kB on disk, but is stored in memory as cell array with one million entries, each occupying 4 byte (single precision), which amounts to about 1.8 GB in memory. Due to Octave's (and equally Matlab's) memory management, this nevertheless seems to be the upper limit, lest swapping heavily.

This 1024×1024 px images correspond to a FOV of 1.27×1.27 mm, which amounts to a captured AOI A_{1k} of:

$$A_{1k} \approx 1.6\,\mathrm{mm}^2 \tag{3.11}$$

3.1.5 Coherence

The largest dimensions of both the AOI A_{256} as well as the AOI A_{1k} are well below the theoretical coherence length, described in Section 2.3.5, a prerequisite for applying the GS algorithm embedded in Fourier Ptychography:

$$d_{A256} < d_{A1k} < l_{coh} \tag{3.12}$$

3.1.6 Phase Retrieval

A basic FPM recovery on the given stack of images recoveres the previously unknown phase of the sample, shown in Figure 3.5, where all the elemts up to Group 7, Element 6 are perfectly clear!

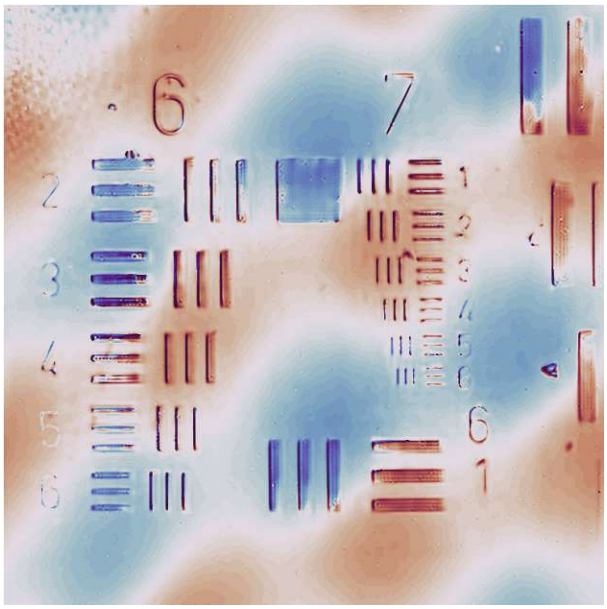


Figure 3.5: FPM recovered phase after 100 iterations.

The phase is shown in a cyclic colormap ranging $[-\pi, \pi]$, where white corresponds to 0, red to the upper half-axis $[0, \pi]$ and blue the lower half-axis $[\pi, 0]$. So the darkness effectively denotes the magnitude of the phase shift with reference to 0 (the darker, the higher the phase shift), whereas the color depicts the direction of the shift.

This phase image is in good accordance to the sample, a USAF1951 glass slide resolution target, who's parameters are listed in Table A.1.

The background of the target is of uniform thickness and transparent, so the phase shift should be equal everywhere; it is around zero in the experiment (white to light blue/red).

The structures consist of a thin chrome layer with high optical density. Theoretically there is no light whatsoever passing this sections, which means no phase shift in reference to the transparent sections either! As described in detail in section 2.5, this has to lead to arbitrary recovered phase; in this case the phase shift is

dark, which amounts to around $\pm\pi$ with reference to the transparent sections.

The wavy patterns of the phase image hint to some sort of misalignment, or another form of disturbance.

3.2 Alignment

Considering the geometry of FPM, it is hardly surprising to see a strong correlation between alignment perfection, and the quality of the obtained result.

To investigate the effect that alignment of the illumination source has on the FPM recovery, we performed simulations, for a variety of circumstances: Different simulated amplitude and phase *samples*, various levels of misalignment.

We always jointly recover the pupil, based on the initial scheme [Zheng et al., 2013]. A profound inspection of the recovered pupil function (like aberrations of the microscope setup) are beyond the scope of this thesis. We simply check if the recovered pupil is in agreement with the theoretical circular shaped pupil function. For example effects like aliasing would appear as regular grid of dots in the pupil.

For the sake of better comparability of different amplitude and phase samples, we do not use sparse sampling (Section 2.5.1) [Zheng, 2016], although it does improve the FPM recovery in our tests.

We analyse brightness variations in Section 3.3, so for the following alignment we consider the illumination source to be a perfect plane wave.

Although we can in principle perform FPM on larger images, our setup is somewhat hardware limited (see also different image dimensions in Section 2.3.7). The simulations even exceed the standard FPM recovery in terms of memory requirements, so we opt to smaller images of 256×256 px.

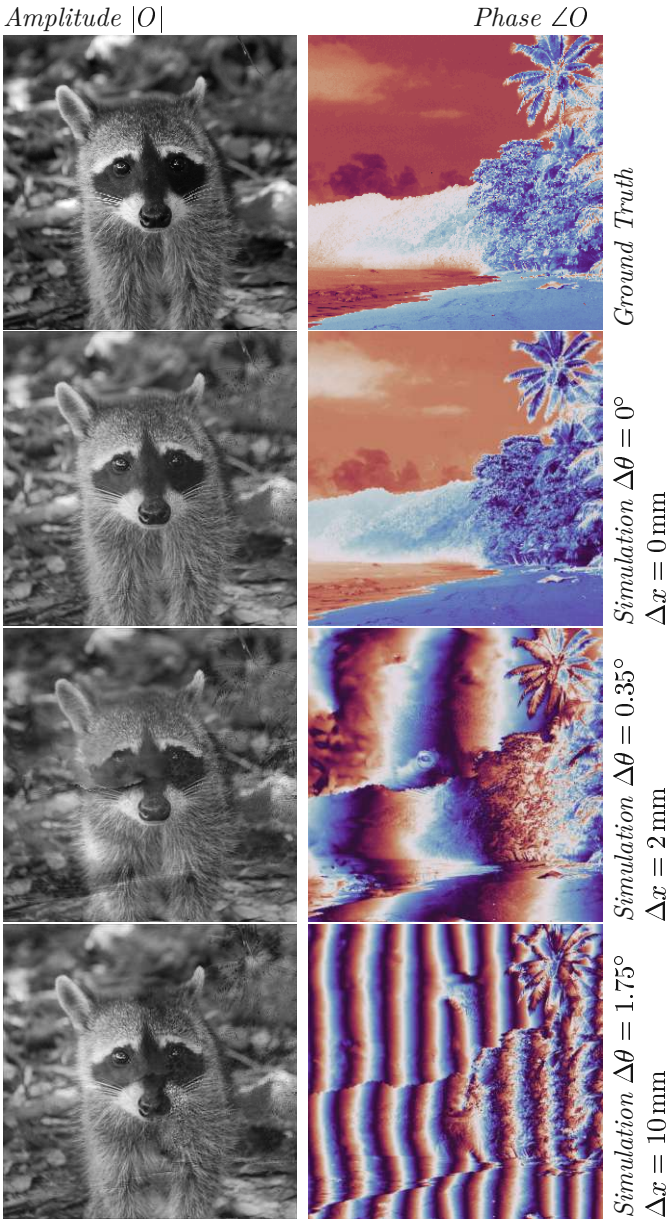


Figure 3.6: Demonstrating the impact of misalignment of the whole LED panel on reconstruction amplitude (left) and phase (right) in respect to ground truth (top). Even a misalignment of mere $0.35°$ *(*$2\,\mathrm{mm}$ *shift) renders the recovered phase quite useless.*

Even narrowed down to just a few parameters, it still is impossible to discuss all these findings in detail within the framework of this thesis. We give an overview based on a appropriate example, and conclude providing broad overall limits.

These limits are to be considered as a rule of thumb on how each type of misalignment is effecting the overall FPM operation.

This work shows on the basis of simulations, that even a misalignment of the illuminating LED matrix of only a 0.2 degrees (equivalent to a shift of 1 mm for the used LED matrix' optimal setup) poses a serious threat to FPM recovery!

This result is exemplary shown for three different misalignment of the LED panel (shift in x-axis) in Figure 3.6, using arbitrary images as a basis for amplitude and phase. Interestingly FPM is able to recover the amplitude even under bad conditions, as one can see all of the racoons quite clearly.

Unfortunately the recovered phase is heavily distorted even at tiny misalignment of 0.35° (2 mm shift)! This explains the need for careful calibration and correction algorithms used widely [Eckert et al., 2018].

3.2.1 Simulated Misalignment

For a more thorough simulation of misaligned FPM recovery, the amplitudes a_n of a stack of simulated low-resolution images ι_n are altered as decribed in Section 2.6.1. The alignment simulation is performed numerous times with logarithmically spaced misalignment magnitudes α_m, in this case Δx. For each of these, a full FPM recovery with 100 loops is done. For each alignments in consideration (shift, yaw, pitch & roll) in the following sections, convergence, amplitude and quality are closely monitored. Yet for the sake of simplicity we present a detailed explanation on these topics on one exemplary misalignment only: *shift*, so every subsequent notion of α_m in this section refers to misalignment magnitudes of lateral shift Δx.

Convergence

The convergence of the algorithm is shown in Figure 3.7 for all α_m, where for the sake of distinguishing the individual lines, the plot features a double-logarithmic scale.

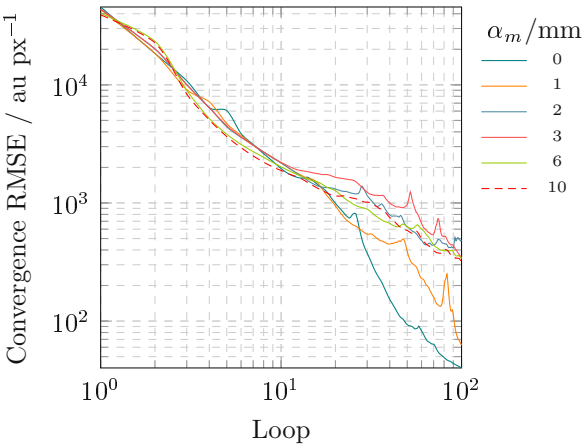


Figure 3.7: Exemplary convergences of FPM simulation reconstruction under various misalignment magnitudes α_m in this case of Δx (lines), versus loop in double-logarithmic scale (lower is better). Clearly perfect alignment (Δx=0 mm) leads to fast convergence—which is obviously not linear in misalignment.

Starting at approximately the same point, the convergence is almost oblivious to misalignment for the first 10 iterations. Clearly, the better the alignment (lower α_m), the better the convergence (lower resulting changes) gets in the following iterations. The convergence is not at all linear in misalignment though.

the convergence *speed*—that is the rate in which the converges changes over iterations—is pretty similar for all the misalignment; rapidly decaying over iterations! Mind that this is a double logarithmic plot.

Discordance

On the grounds of convergence alone, this simulations would suggest (quite wrongly) that the recovery is working good, no matter the uniformity of illumination, so we take a look at the discordance: Shown in Figure 3.8, again for all α_m, again on a double-logarithmic axis.

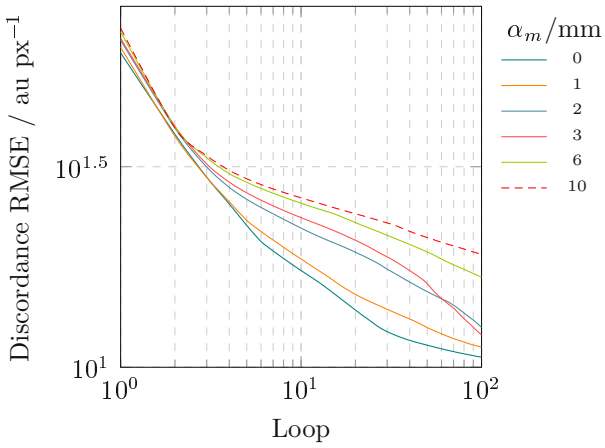


Figure 3.8: Discordances between FPM simulation reconstruction and low-resolution images under various misalignment magnitudes α_m in this case of Δx (lines) versus loop in double-logarithmic scale (lower is better). The better the alignment (low Δx), the better the start, but discordance is obviously not linear in misalignment either.

Obviously the evolution of discordance—the RMSE deviations of the absolute value of the reconstruction versus the low-resolution images—differs substantially in respect to the misalignment α_m.

Quite clearly the discordance is highly nonlinear, but dependent of the alignment of the illumination source: The better aligned, the lower the resulting discordance—which is good. Overall they perform impressively similar considering the double-logarithmic scale.

Amplitude Quality

Fortunately in the case of simulations, we hold the opportunity to compare the recovery to the ground truth at any time. The amplitude quality—RMSE deviation in amplitudes—is shown in Figure 3.9 for all α_m on a double logarithmic scale.

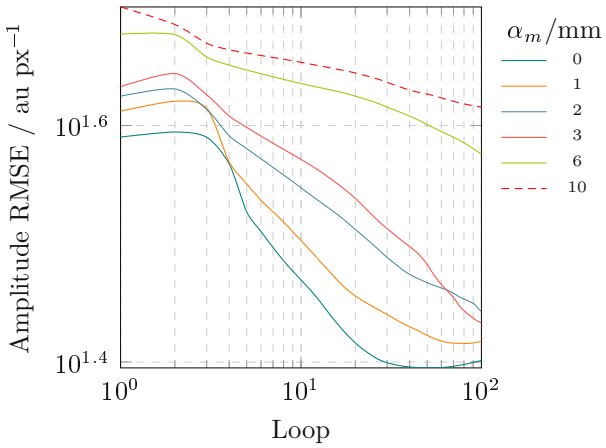


Figure 3.9: Quality of FPM simulation reconstructed amplitude under various misalignment magnitudes α_m in this case of Δx (lines): Amplitude RMSE versus loop in semi-logarithmic scale (lower is better). The better the alignment (low Δx), the better the start (low Amplitude RMSE), but Amplitude RMSE is clearly not linear in misalignment either.

Surprisingly, the perfect aligned as well as the reasonably well aligned simulations ($\alpha \leq 3$), show a increase in amplitude RMSE during the first iterations; so their recovered amplitudes are actually *decreasingly* representing the captured (simulated) low-resolution images!

We repeatedly encountered this behaviour, which we attribute to the fact that the FPM algorithm is in this stage building the phase image from *nothing*, so there is a lot happening and improving behind the curtain.

Even considering the fact, that the noisier simulations show a decline in amplitude

RMSE—which is good—over the whole course, they stay on a quite high level. This indicates that the recovery probably got stuck in some local minimum not resembling the real object at all.

Only the perfect aligned as well as the reasonably well aligned simulations reach a plateau at the end, indicating convergence, which is not unusual for FPM [Fienup, 1982].

On the basis of amplitude quality, it essentially depends on the threshold of RMS deviation one sets to accept. At least for simulations featuring lateral shift of ($\alpha \geq 6$), the recovered object is clearly distorted; here the RMS deviaiton is already around 40 uint8/px or 16%$_{uint8}$.

Phase Quality

Last but not least, the quality of the recovered phase is estimated as RMS deviation to the ground truth, shown in Figure 3.10 for all α_m on a double logarithmic scale.

Obviously the quality of the recovered phase is highly depending on the alignment perfection: The perfect aligned as well as the next best aligned simulations ($\alpha = 1$) decline relatively similar, whereas all the worst simulations ($\alpha \geq 2$) even rise from the beginning!

Again there is no linear correlation between quality of phase and the uniformity of illumination, but overall the better the alignment (low α_m), the higher the quality (low phase RMSE).

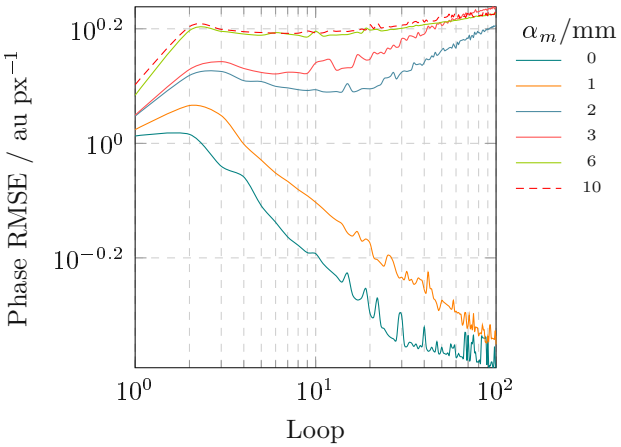


Figure 3.10: Quality of FPM simulation reconstructed phase under various misalignment magnitudes α_m in this case of Δx (lines): Phase RMSE versus loop in semi-logarithmic scale (lower is better). There is no linear correlation between quality of phase and misalignment, but overall the lesser the misalignment (low Δx), the higher the quality (low phase RMSE).

3.2.2 Bright-Field Calibration

The susceptibility of FPM to the alignment of the illumination source, as well as the complicated nature of aligning the LED panel due to being off-focus, explains the need of calibration [Zhang et al., 2019] or even digital post-correction [Zhou et al., 2018].

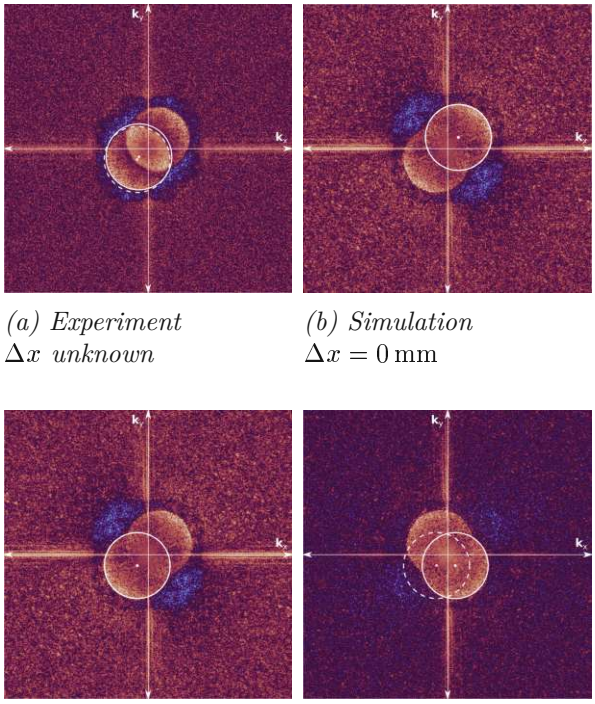
In order to test the proposed BF calibration method (Section 2.6.2) based on[Eckert et al., 2018]; a set of pictures simulating a misaligned experimental setup were generated, as described in Section 2.7.

The main goal here is to find misalignment of the whole LED panel, even though it is conceivable that some individual LEDs are mispositioned on panel assembly level too.

As an example the spectra of both experiment (3.11a) and simulation (3.11b–3.11d), for the same arbitrary illumination angle $\theta_n = 1.5°$, the latter at increasing misalignment (lateral shift) are shown in Figure 3.11. The corresponding illumination positions and pupil radii are shown for assumed (x, dashed line) and found (o, line) disks.

3.2.3 Shift

Since the most common experimental setup for FPM is a LED panel, the most impor-



(a) Experiment Δx *unknown*

(b) Simulation $\Delta x = 0\,\mathrm{mm}$

(c) Simulation $\Delta x = 2\,\mathrm{mm}$

(d) Simulation $\Delta x = 10\,\mathrm{mm}$

Figure 3.11: Alignment Correction: Comparison of exemplary spectra $|\hat{\imath}_n|$ *(Algorithm 4 Line 4) of both experiment (3.11a) and simulation (3.11b–3.11d), for the same arbitrary illumination angle* $\theta_n = 1.5°$ *at increasing shift; One can clearly see the autocorrelation circles, the corresponding illumination positions and pupil radii are shown for assumed (x, dashed line) and found (o, line) disks.*

tant alignment calibration probably is lateral shift along the x-axis, respective the y-axis [Eckert et al., 2018]. The height (shift in z) is comparatively easy to adjust by direct measurement, whereas the shift in x, y, is extremely tedious to measure. This is due to the fact, that the LED panel is off focus, so there is no easy way of answering the question where the central LED is, in respect to the optical path.

In this simulation the whole grid was shifted in x-axis, respective y-axis given the parameters Δx_m, respective Δy_m shown in Table 3.1 together with results of the BF calibrations. Both the mean with sample standard deviation (STD) and median with median absolute deviation (MAD) is shown for comparison.

Table 3.1: Calibration of shifted simulations for three examples.

Parameter	Δx / mm	Δy / mm
Simulation 1		
real	0	0
mean ± STD	-0.1 ± 0.3	0.0 ± 0.3
median ± MAD	0.0 ± 0.3	0.0 ± 0.2
Simulation 2		
real	-5	8
mean ± STD	0 ± 20	0 ± 20
median ± MAD	-4.9 ± 0.6	8.3 ± 0.8
Simulation 3		
real	3	-1
mean ± STD	3 ± 2	-1 ± 8
median ± MAD	3.2 ± 0.3	-1.1 ± 0.3
Experiment 1		
real	unknown	unknown
mean ± STD	0.7 ± 0.6	0.6 ± 0.5
median ± MAD	0.6 ± 0.5	0.6 ± 0.3

For all three simulations (Simulation 1, not shifted; and Simulation 2 and 3, both shifted ones), their median with MAD is in good accordance with the real values. Their mean with STD on the other hand is heavily distorted, rendering the mean with STD results insignificant.

On the contrary we may conclude from the Δx and Δy values of Experiment 1, that the experimental setup was slightly shifted:

$$\Delta x_{exp} = 0.6 \pm 0.5\,\mathrm{mm} \tag{3.13}$$

$$\Delta y_{exp} = 0.6 \pm 0.3\,\mathrm{mm} \tag{3.14}$$

where we used the median values, which we consider the median more significant than the arithmetic mean, as it is conceivable that some of the LEDs are mispositioned on the level of production—which does not represent a shift of the whole panel.

Descriptive Statistics

The reason is that the mean with STD is weighting the outsiders higher; so if a few outsiders (that are far off) are not to be taken seriously, the median with MAD allows a better description of the sample.

In Figure 3.12 the order of magnitude of the calibrated misalignment, $\mathcal{O}(\Delta x_n) = \log_{10}(\Delta x_n) \,/\, \log_{10}(\mathrm{m})$, for each LED is shown for Simulation 2.

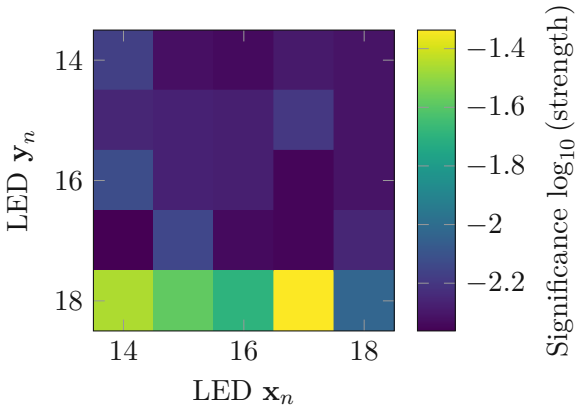


Figure 3.12: Order of magnitude of the position misalignment for the BF region of the LED panel.

One can clearly see, that the majority of the values are quite in the same range, except those in the bottom row, where they are far off (roughly one order of magnitude). Though the explanation for this is not obvious, but may be neared with a cautious look at the spectra in question. It turns out, that due to the misalignment of $\Delta x = -5\,\mathrm{mm}, \Delta y = 8\,\mathrm{mm}$, the bottom row LEDs are shifted just outside the cutoff frequency, thus into the DF!

It is a hard prerequisite for the proposed calibration though, that the images are taken under BF conditions, so these values are not to be considered. But how to find out if the given image was indeed a BF image, thus the calibration result could be considered significant?

A first estimation of the corrections significance conveniently comes included in the procedure `imfindcircles`: the *strength* of the found circles; sort of the probability, that the found circle is indeed a circle. To visualise this, the order of magnitude of the strength is exemplary shown for Simulation 2 in Figure 3.13.

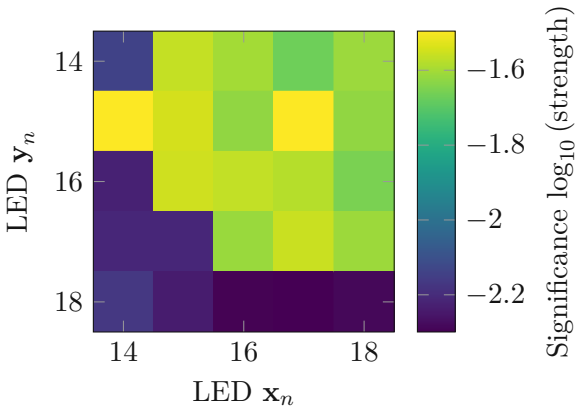


Figure 3.13: Strengths (significance) of the found radii $s_{x,y}$ of simulation vs LED positions x_n, y_n for Simulation 2.

It appears, that the results with a strength below the order $\mathcal{O}(-2)$ are probably not significant, and may be rendered outliers. To visualise this, the initial and actual (shifted) grid of Simulation 2 is shown in Figure 3.14, together with the corrected positions (circles), colour-coded in order of magnitude of their strength.

The relation between the strengths and the validity of the corrections are quite evident, yet the precise border between in- and outliers is made empirically, and unfortunately have to be expected to vary with respect to the sample in question!

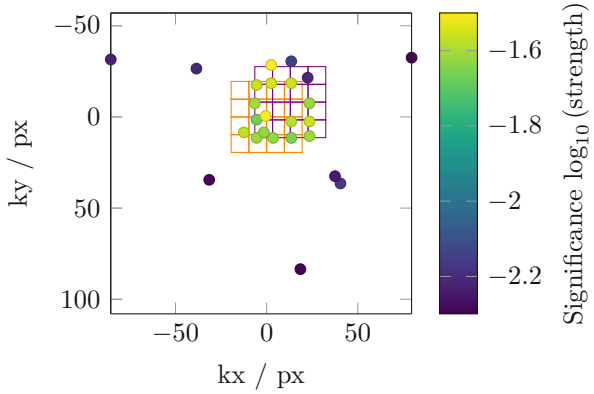


Figure 3.14: Correcting shift: Comparison of the k-space projections $\boldsymbol{k}_n$ (nodes) of the BF LED positions of the slightly shifted panel (violet); versus the initially assumed grid (orange); versus the corrected positions (circles), colour-coded in the order of magnitude of their strengths (colour, higher is better).

As an example of further analysis—that in this case confirms prior judgement—an exemplary spectrum of low strength is shown with the estimated and the corrected position in Figure 3.15 (middle row left). Quite obviously there is no autocorrelation disk present, thus no BF calibration possible for this image.

3.2.4 Yaw

A considerably harder problem is the case, where the LED panel is turned around the z-axis: Yaw. Like shift it is not that easy to align in a physical calibration step, as one has to lower the focus down to the LED panel. Additionally, yaw results in a misalignment that is proportional to the distance of the LED to the centre of the panel, rendering previously used descriptive statistics of the whole panel as one ensemble, including state of the art correction methods [Eckert et al., 2018], useless.

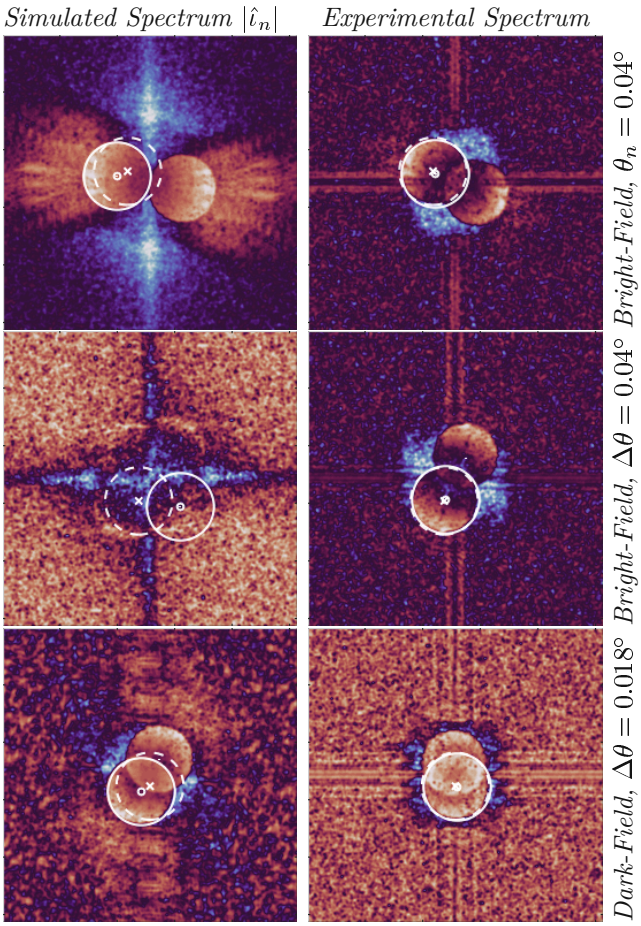


Figure 3.15: Alignment Correction: Comparison of the spectra $|\hat{\imath}_n|$ of both simulation (left column) and experiment (right column), for three arbitrary illumination angles θ_n; One can clearly see the autocorrelation circles, the corresponding illumination positions and pupil radii are shown for assumed (x, dashed line) and corrected (o, line). Note the malfunctioning circle detection, middle row left

Inclination & Fit

The solution proposed here is to fit the set of corrected positions to a grid, and subsequently to evaluate the angle γ between this and the initial grid for each row and each column sep-

Table 3.2: Calibration of yawed simulations for three exemplary samples.

Parameter	γ_x / deg	γ_y / deg
Simulation 1		
real	0	0
mean ± STD	0.5 ± 0.8	-0.1 ± 0.9
median±MAD	0.5 ± 0.7	0.0 ± 0.6
Simulation 4		
real	3	3
mean ± STD	3.2 ± 0.8	3.1 ± 0.9
median±MAD	2.9 ± 0.6	2.93 ± 0.08
Simulation 5		
real	11	11
mean ± STD	11 ± 2	11.4 ± 0.5
median±MAD	11.6 ± 0.2	11.5 ± 0.1
Experiment 1		
real	unknown	unknown
mean ± STD	-0.5 ± 0.3	-0.7 ± 0.7
median±MAD	-0.62 ± 0.02	-0.60 ± 0.06

arately:

$$\gamma \approx \arctan(\beta) \tag{3.15}$$

where β refers to the slope of the *horizontal* and *vertical* lines of the grid in respect to the initial grid. In the horizontal case, $\boldsymbol{\zeta}$ denotes the vector $\mathbf{k}_x$, and $\boldsymbol{\chi}$ denotes the $\mathbf{k}_y$. In the vertical case vica versa. Since the data points of these lines have to be considered noisy, we assume a linear progression, via solving the least squares fit:

$$\boldsymbol{\zeta} = \beta\, \boldsymbol{\chi} + \delta \tag{3.16}$$

Depending on the size of the BF area of $n_b \times n_b$ LEDs, we end up with n_b horizontal inclinations γ_x and n_b vertical ones γ_y, which can be considered one statistical ensemble each. Again, careful observation of the deviations reveals if the panel is yawed (low deviations) or otherwise skewed (high deviations); And thus if we can deduce a meaningful mean and median.

In this simulation the whole LED panel was turned around the z-axis, which results in a k-space grid inclined in respect to the k-axis about the angle γ_n. Three different simulations featuring various inclinations are stated together with the results of the BF calibrations—both the mean with STD and median with MAD—in Table 3.2.

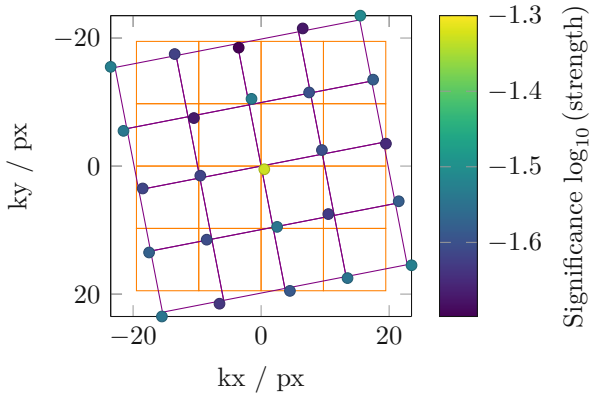


Figure 3.16: Correcting yaw: Comparison of the k-space projections $\boldsymbol{k}_n$ (nodes) of the corrected bf LED positions of the highly misaligned panel of Simulation 5 (circles), colour-coded in the order of magnitude of their strengths (fill, higher is better); versus the actual grid with yaw $\gamma = 11°$ (violet); versus initially assumed grid (orange).

For all three examples (Simulation 1, not yawed; and Simulation 2 and 3, both yawed ones), both their mean with STD, as well as their median with MAD is in good accordance with the real values.

Finally this allows the following conclusion: if—and only if—the both inclinations γ_x and γ_y are almost equal, we may deduce that this grid was only yawed. All other inclinations (e.g. pitch or roll) would have tilted the grid in a way, such that $\gamma_x \neq \gamma_y$.

Likewise we may conclude from the γ values

of Experiment 1, that the experimental setup was not tilted ($\gamma_x \approx \gamma_y$), but slightly yawed:

$$\gamma_{exp} = -0.61 \pm 0.04^\circ \tag{3.17}$$

As an example, the BF area of the k-space grid of a slightly misaligned panel ($\gamma = 11^\circ$) is shown in comparison with the actual and initial grids in Figure 3.16.

The proposed fits provide a method more resilient than fitting to a periodic rectangular grid—especially for nonlinear skewed grids (like pitch or roll).

3.2.5 Pitch & Roll

The methods described so far provide a direct measure of the shift, respective yaw in the k-space grid of the LED panel, both of which are translation invariant in z-direction. Unfortunately these are the special cases of a bijection; where the misaligned LED panel is still strictly parallel to the object (and sensor), so it has a direct relation to the 2D grid in k-space.

Considering pitch and roll (rotations around the x-, respective the y-axis), the LED panel is not parallel to the sensor anymore. This results in a much harder problem: The mapping between the 3D real space and the measured 2D k-space is not bijective anymore. Thus it does not have an inverse function; it is not possible to infer to the 3D real space misalignment by measurement of the 2D k-space grid!

As a general example of this problem, two simulations featuring profoundly tilted LED panels are compared with the experiment in Table 3.3.

An example for such a heavily tilted LED panel, Simulation 6, is shown in Figure 3.17: roll $\rho = 30^\circ$, no pitch.

The grid is still rectangular, but unlike previous examples the kx-, and ky-axes are not spaced equidistantly anymore! On the one hand the grid is *compressed* in ky-axis; one the other hand it forms a trapezoid. This is only very slightly visible in both shown examples, due to the panel being much farther away (327 mm in z-direction) as opposed to the difference in height caused by the tilt (±45 mm

Table 3.3: Calibration of tilted simulations for three examples.

Parameter	γ_x / deg	γ_y / deg
Simulation 6		
mean ± STD	0.4 ± 0.8	0 ± 2
median±MAD	-2.91 ± 0.03	0 ± 2
Simulation 7		
mean ± STD	0.5 ± 0.9	10.1 ± 0.7
median±MAD	0.1 ± 0.2	9.63 ± 0.09
Experiment 1		
mean ± STD	-0.5 ± 0.3	-0.7 ± 0.7
median±MAD	-0.62 ± 0.02	-0.60 ± 0.06

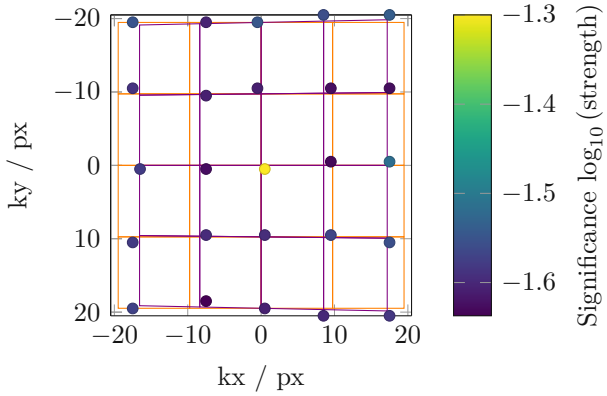


Figure 3.17: Correcting roll & pitch: Comparison of the k-space projections $\boldsymbol{k}_n$ (nodes) of the corrected bf LED positions of the highly misaligned panel of Simulation 6 (circles), colour-coded in the order of magnitude of their strengths (colour, higher is better); versus the actual grid with roll $\rho = 30^\circ$ (violet); versus initially assumed grid (orange);

in z-direction).

Unsurprisingly the resulting mean γ_x and γ_y values are both around zero, but with huge standard deviations. The mean is proffered here, because we do not have to consider extreme outliers shifted into DF. Either way, the spread of the values due to the compression & trapezoid effects is quite high—which is a strong indicator that the setup exceeds correctability.

This results in the median being off -2.91, although the precision seems to be quite high (0.03)—which is dangerous caveat!

Simulation 6 gives an example of such a profoundly tilted LED panel—roll $\rho = 21°$ and pitch $\psi = 30°$— shown in Figure 3.18.

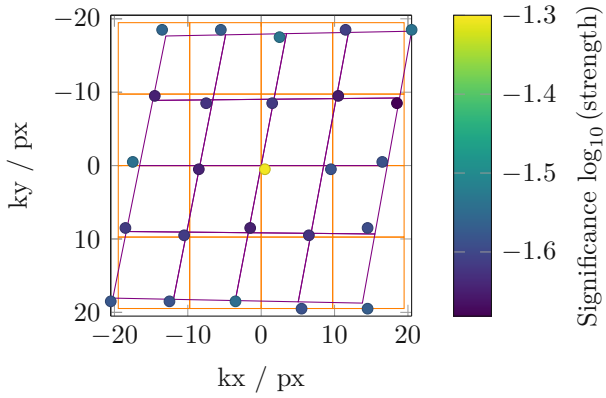


Figure 3.18: Correcting roll & pitch: Comparison of the k-space projections $\boldsymbol{k}_n$ (nodes) of the corrected bf LED positions of the highly misaligned panel of Simulation 6 (circles), colour-coded in the order of magnitude of their strengths (colour, higher is better); versus the actual grid with roll $\rho = 21°$ and pitch $p = 30°$ (violet); versus initially assumed grid (orange).

Evidently, the grid is not rectangular anymore. Analysis correctly depicts this: γ_x and γ_y values are both around zero for the kx-axis, yet around 10° for the ky-axis. Both median and mean deviations support this claim, which inherently points to tilt of the panel in x- and or y-axis.

Quite on the contrary for Experiment 1, where all the γ values are sound at $-0.6°$; which leads to the conclusion of no roll and no pitch in this particular setup:

$$\rho_{exp} \approx 0° \tag{3.18}$$

$$\psi_{exp} \approx 0° \tag{3.19}$$

3.2.6 Automatic Self-Calibration

Based on these analysis on the nonlinear consequences of three dimensionally misaligned setup, we would strongly advice against any form of automatic self-correction procedure, if roll & pitch cannot be assumed negligible without doubt. In which case, for automatising the remaining much simpler problem of lateral shift, the interested reader may refer to [Zhou et al., 2018] for an approach using simulated annealing alone, or to [Eckert et al., 2018] for an improvement based on autocorrelation, RANSAC outlier detection and finally also simulated annealing.

Aware of the dangers of silently automatising intricate processes, as highlighted by [Randall, 2011b] in Figure 3.19, we would rather abstain from fully automating this calibration processes anyway.

Our approach nevertheless provides well articulated information to correct the physical alignment.

3.3 Illumination

Considering the requirements of FPM for the illumination source—equidistant coherent point source generated plane waves—it is hardly surprising to see a strong correlation between uniformity of the illumination, and the quality of the obtained result.

To investigate the effect that brightness variations of the individual LEDs have on the FPM

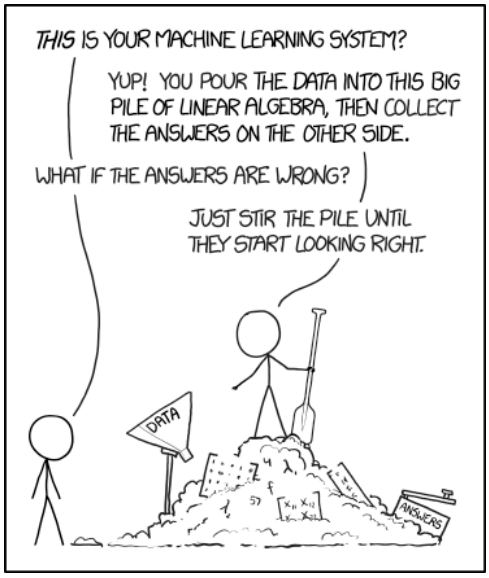


Figure 3.19: About Machine Learning [Randall, 2011a], obtained under CC-BY-SA license.

recovery, we performed simulations, for a variety of circumstances:Different simulated amplitude and phase *samples*, various levels of brightness deviation.

Similar to Section 3.2 we always jointly recover the pupil function, yet we do not use sparse sampling.

We analyse alignment in Section 2.6.1, so for the following luminosity we consider the illumination source perfectly aligned.

To illustrate the effect of inconsitent illumination, three FPM recoveries of simulated, progressively inconsistent illuminated arbitrary objects (right: amplitude, left: phase) are compared to ground truth (top) in Figure 3.20. Even a non-uniformity of the average brightness of the images in the order of a few % uint8 devastates the recovery, as for the last row with $8\,\%_{\text{uint8}}$ ($\nu_m = 22\,\text{uint8}$).

Although we can in principle perform FPM on larger images, our setup is somewhat hardware limited (see also different image dimensions in Section 2.3.7). The simulations even

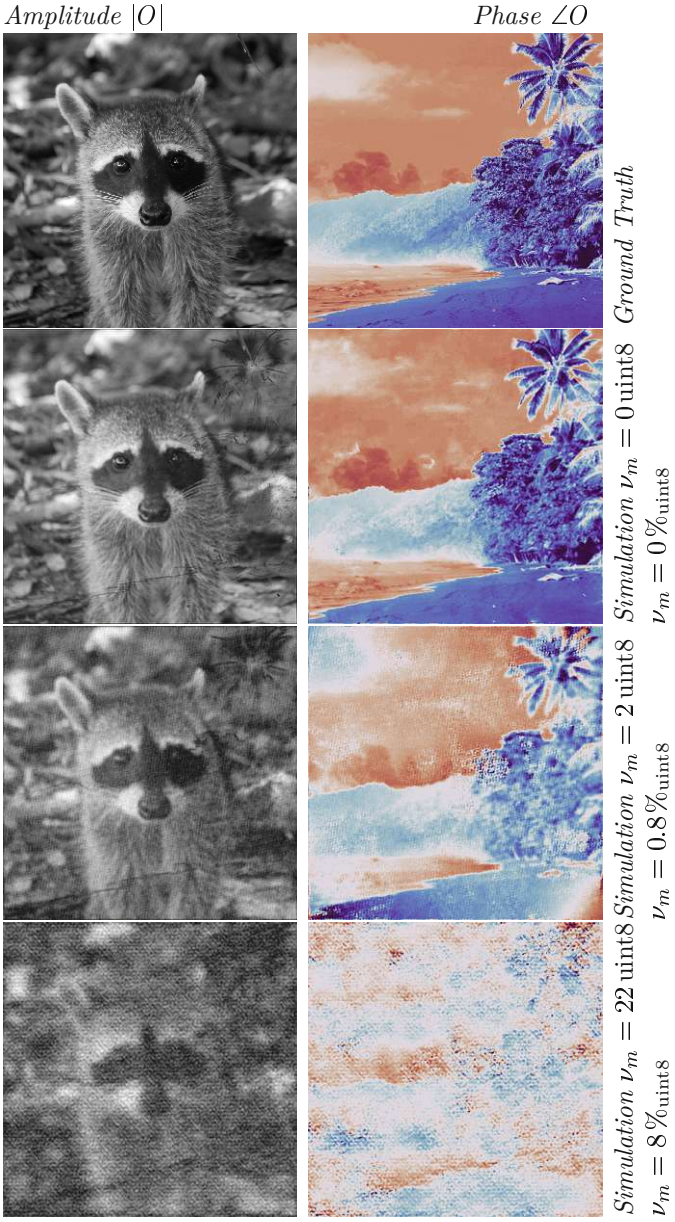


Figure 3.20: Demonstrating the impact of inconsistent illumination on reconstruction amplitude (left) and phase (right) in respect to ground truth (top). Even a non-uniformity of the average brightness in the order of mere $8\,\%_{\text{uint8}}$ ($\nu_m = 22\,\text{uint8}$) renders the recovered amplitude and phase quite useless.

exceed the standard FPM recovery in terms of memory requirements, so we opt to smaller images of 256×256 px.

Even narrowed down to just a few parameters, it still is impossible to discuss all these findings in detail, within the framework of this thesis. We give an overview based on examples, and conclude providing broad limits. These are to be considered as a rule of thumb on how each type of misalignment is effecting FPM.

3.3.1 Simulated Inconsistency

For the simulation of inconsistently illuminated FPM recovery, the amplitudes a_n of a stack of simulated low-resolution images ι_n are altered as decribed in Section 2.7.1. To best fit our data sets, we model a directivity of $\delta = 40°$. The luminosity simulation is performed numerous times with logarithmically spaced noise magnitudes ν_m, listed in Table 3.4.

Table 3.4: Noise magnitudes ν_m simulating inconsistent illumination

Simulation m	ν_m / uint8	ν_m / $\%_{uint8}$
1	0	0
2	0.1	0.04
3	0.2	0.08
4	0.5	0.2
5	1	0.4
6	2.2	0.9
7	4.6	1.8
8	10	3.9
9	22	8.6
10	46	18
11	100	39

For each of these noise magnitudes ν_m, a full FPM recovery with 100 loops is done.

Convergence

The convergence of the algorithm is shown in Figure 3.21 for all ν_m, where for the sake of distinguishing the individual lines, the plot features a double-logarithmic scale.

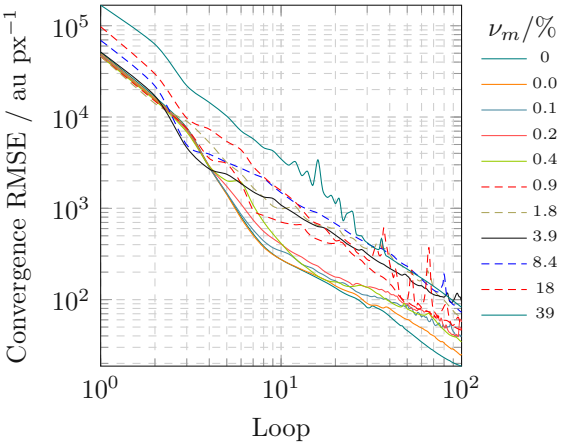


Figure 3.21: Convergences of FPM simulation reconstruction under various luminosity noise ν_m (lines) versus loop in double-logarithmic scale (lower is better).

Clearly, the more uniform the luminosity (lower ν), the better the convergence (lower resulting changes). Though the convergence is not at all linear in noise, the convergence *speed*—that is the rate in which the converges changes over iterations—is still pretty much dependent of noise magnitude: Less noise means lower initial (and resulting) changes to the complex object—and even better—less noise means *faster* convergence! Mind that this is a double logarithmic plot, so what appears to be a line is actually not.

Also note, that the convergence for the most noisy simulations with $\nu = \{18, 39\}$ % exhibits some fluctuations starting with around 10 loops, which might be an early sign of trouble.

Discordance

On the grounds of convergence alone, this simulations would suggest (quite wrongly) that the recovery is working good, no matter the uniformity of illumination, so we take a look at the discordance: Shown in Figure 3.22, again for all ν_m, again on a double-logarithmic scale.

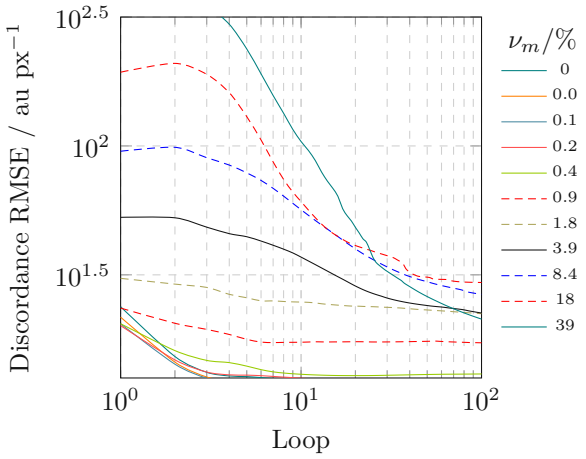


Figure 3.22: Discordances between FPM simulation reconstruction and low-resolution images under various luminosity noise ν_m (lines) versus loop in semi-logarithmic scale (lower is better).

Obviously the evolution of discordance—the RMSE deviations of the absolute value of the reconstruction versus the low-resolution images—differs substantially in respect to the illumination uniformity ν_m.

Notably, the less noisy simulations show a decrease at the beginning, stagnating slowly. Overall they perform impressively similar considering the double logarithmic scale! This hints to some invariance to illumination incosistence with a standard deviation below one percent.

Quite clearly, the resulting discordance is highly nonlinear, but dependent of the uniformity of illumination: The lesser noise, the very better (lower) the resulting discordance.

From the discordance alone, one might be tempted to conclude, that the degree of noise from 2 to 40 % does not matter, since the final discordances are almost equal. This is certainly not the case! Sadly for experimentation, this is about it in terms of prediction.

Amplitude Quality

Fortunately in the case of simulations, we hold the opportunity to compare the recovery to the ground truth at any time. The amplitude quality—RMSE deviation in amplitudes—is shown in Figure 3.23 for all ν_m on a double logarithmic scale.

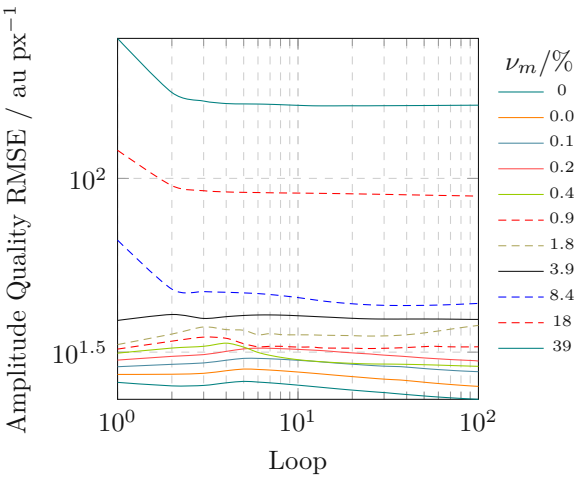


Figure 3.23: Quality of FPM simulation reconstructed amplitude under various luminosity noise ν_m (lines): Amplitude RMSE versus loop in semi-logarithmic scale (lower is better).

Even considering the fact, that the noisier simulations show a fast decline in amplitude RMSE—which is good—during the first iteration, they stay more or less constant from the second iteration onward; on a quite high level. This indicates that the recovery probably got stuck in some local minimum not resembling the real object at all.

The simulations featuring low noise (under

1%) start with significantly lower deviation, slowly converging further.

On the basis of amplitude quality, it essentially depends on the threshold of RMS deviation one sets to accept. At least for simulations featuring noise above 4%, the recovered object clearly does not resemble the target; here the RMS deviaiton is already around 50 uint8 or 20%!

Phase Quality

Last but not least, the quality of the recovered phase is estimated as RMS deviation to the ground truth, shown in Figure 3.24 for all ν_m on a double logarithmic scale.

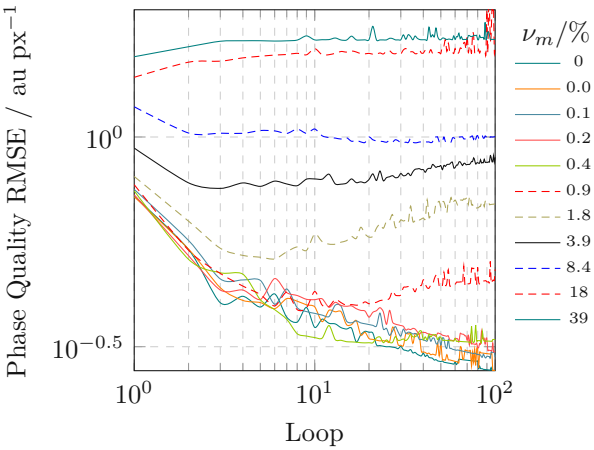


Figure 3.24: Quality of FPM simulation reconstructed phase under various luminosity noise ν_m (lines): Phase RMSE versus loop in semi-logarithmic scale (lower is better).

Obviously the quality of the recovered phase is highly depending on the uniformity of illumination: The less noisy simulations ($\nu_m < 1\%$) decline relatively similar, whereas the worst simulations (ν_m of 18% and 39%) even rise from the beginning! In between this two extremes, the RMS deviations decrease for some iterations, sometimes rising again later.

There is no linear correlation between quality of phase and the uniformity of illumination, but overall the lesser the noise (low ν), the higher the quality (low phase RMSE).

3.3.2 Luminosity Calibration

Targetless

To illustrate the variation in brightness of the RGB panels SMD LEDs, a patchwork of segments of BF images is shown in Figure 3.25. Undeniably these images feature variations in average brightness about a few percent, except for the farthest out images, which are considerably darker (up to 2.5 times).

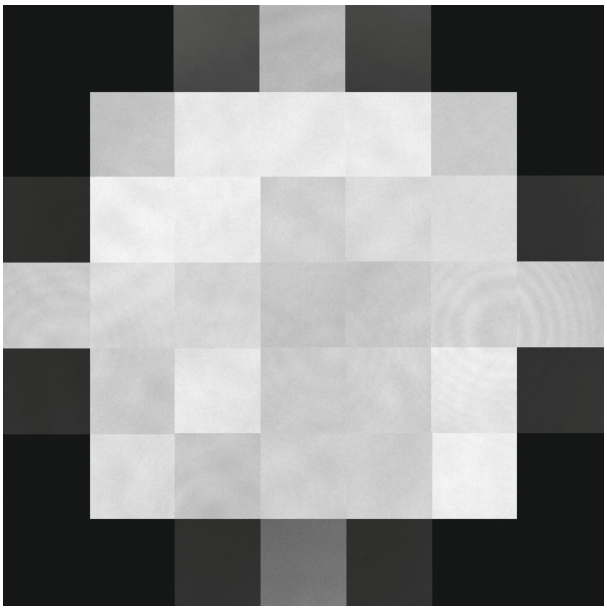


Figure 3.25: A patchwork of segments of the BF area (bright images) of targetless images, obviousely of varying brightness.

These captured BF images are then analysed using Algorithm 6; the resulting calibration factor matrix $\Lambda_{b,n}$ for our experimental setup is shown in Figure 3.26. A brief comparison with the patchwork in Figure 3.25 reveals:

The darker images feature a $\Lambda_{b,n} > 0$, so they get amplified, whereas the brighter images get attenuated with $\Lambda_{b,n} < 0$.

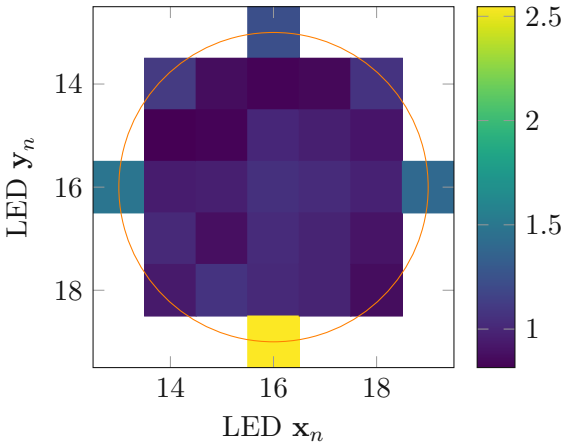


Figure 3.26: BF luminosity calibration factor $\Lambda_{b,n}$ for each of the n images, and NA (circle) seperating BF from DF; versus the LED positions $\boldsymbol{x}_n$, $\boldsymbol{y}_n$.

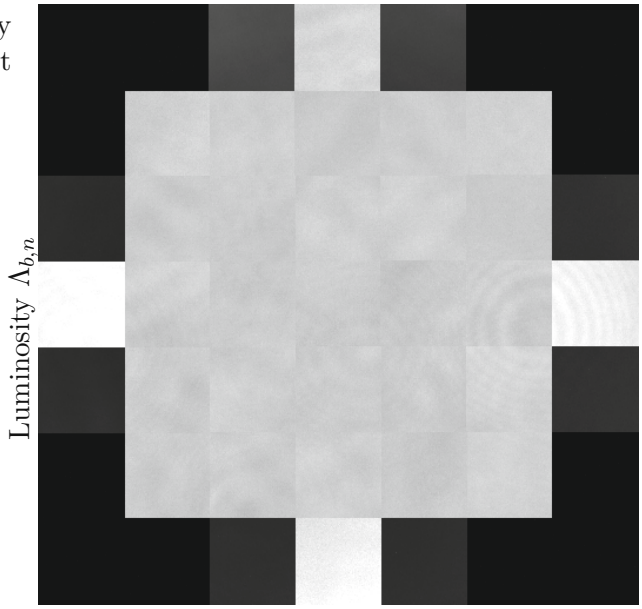


Figure 3.27: A patchwork of segments of the BF area (bright images) of luminosity calibrated targetless images.

In order to visualise this calibration, the same patch as in Figure 3.25, now created out of calibrated images is shown in Figure 3.27.

Note that we only calibrate the BF region here. So if parts of the detector are already in the DF regime (as happened to be the case for some images close to the NA), the average of the image is considerably lower than it should be, which renders the luminosity calibration useless.

To stay on the safe side, images close to the NA should be treated with caution.

Diffused

Using a diffuser as described in Section 2.7.2, one may even calibrate the DF images, as light gets scattered into the lens by the diffuser. The average brightness μ_n of the amplitudes a_n of the diffused images $\imath_n$ captured by the camera are shown in Figure 3.28 for all 931 LED positions $\mathbf{x}_n$,$\mathbf{y}_n$. The SMD LEDs of our LED panel indeed feature variations in average brightness of a few percent, as can be seen quite clearly in Figure 3.28 Also the brightness declines with distance to the centre: The attenuation due to directivity.

The calibration utilising Algorithm 6 with the set of captured diffused intensity images yields a normalisation Λ for all n images—in our experimental setup 931 LEDs—as shown in Figure 3.26 together with the NA of the optical system indicating the BF area. Note for the comparison that the calibration factor Λ is the inverse proportional to the average brightness μ: A measured brightness μ_n of an image a_n that is lower than the average of *all* the images $\mu_n < \boldsymbol{\mu}$, means this whole image has to be *raised*; so the corresponding $\Lambda_n > 1$, and vica versa.

To visualise the result of our proposed luminosity normalisation, the average brightness μ'_n of the calibrated amplitudes a'_n of diffused images $\imath_n$, is shown in Figure 3.30 for all il-

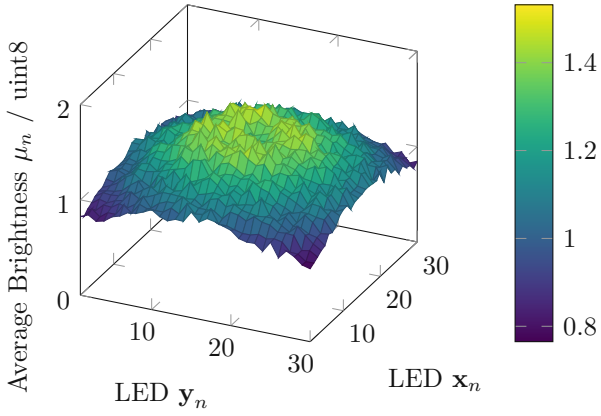


Figure 3.28: Average brightness μ_n (color) of the amplitudes a_n of diffused images ι_n, inconsistently illuminated from the LED positions $\boldsymbol{x}_n$,$\boldsymbol{y}_n$ (nodes), and NA (circle) seperating BF from DF.

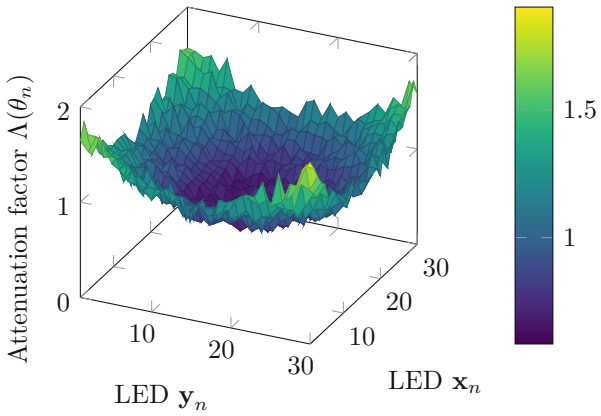


Figure 3.29: Attenuation factor $\Lambda(\theta_n)$ (z, color) for the amplitudes a_n of diffused images ι_n, inconsistently illuminated from the LED positions $\boldsymbol{x}_n$,$\boldsymbol{y}_n$ (x,y nodes), and NA (circle) seperating BF from DF.

lumination positions $\mathbf{x}_n$,$\mathbf{y}_n$, at the same scaling as Figure 3.28. The surface appears to be completely flat, so all the n images now share roughly the same average brightness μ_n.

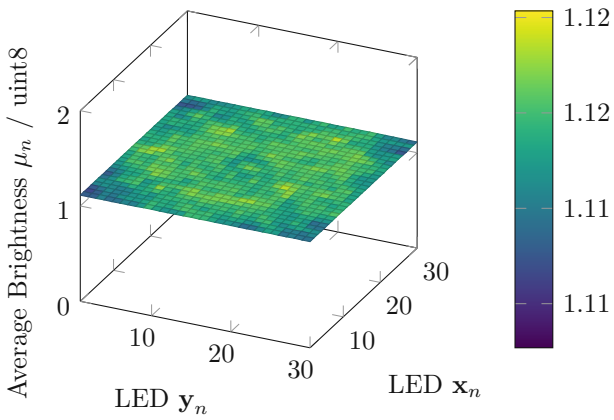


Figure 3.30: Average brightness μ_n (z, color) of the calibrated amplitudes a'_n of diffused images ι_n, inconsistently illuminated from the LED $\boldsymbol{x}_n$,$\boldsymbol{y}_n$ (x,y nodes). Mind that this is in fact a 3D plot: for better comparability the z-axis scaling is the same as for Figure 3.28 (0 to 2); yet appears completely flat, showing the uniformity.

Directivity

The measured brightness that LED to the luminosity calibration factor Λ_n further allows for an estimation of the directivity of the individual LEDs. To achieve that, the pixel-wise arithmetic mean $\mu_n(\theta)$ of the analysed images is normalised and plotted against the illumination angle θ_n of the respective LED, as shown in Figure 3.31.

Interestingly, the shown brightness does not decline with inclination quite as anticipated, estimated by; $\mu_n(\theta) \propto \cos(\theta)$, Equation 2.36.

Instead we find a much faster decline in brightness $I_{exp}(\theta)$, which hints on the actual directivity δ_{exp} being far higher:

$$I_{exp}(\theta) \propto \cos(3\theta) \tag{3.20}$$

$$\delta_{exp} \approx 40° \tag{3.21}$$

For comparison, the brightness curve for $\delta = 40°$ is shown in Figure 3.31, together with the assumed directivity $\delta = 120°$, as described in

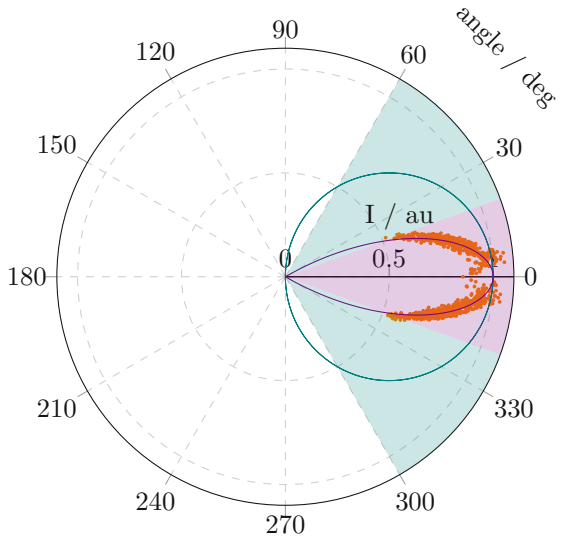


Figure 3.31: Directivity of the LEDs of the whole panel, normalised average brightness versus angle (dots); directivity of 120° *(green); directivity of* 40° *(violet). Note that close to* 0°*, the spread is very high, probably due to sensor saturation.*

Section 2.3.8. Note that close to 0°, the spread is very high, probably due to sensor saturation.

3.3.3 Empirical Illumination Limit

Combining the insights of convergence, discordance, amplitude and phase quality; we can confirm the initial thesis: Illumination uniformity is important for FPM.

We might further estimate normal deviations in the luminosity above 1% (±2.5 uint8) to be fatal for FPM recovery:

$$\nu_{max} = 1\% \tag{3.22}$$

Careful analysis of the histogram, spectrum, amplitude and phase as well as mean and standard deviation support this claim. Yet this threshold is arbitrary: The less uniform the illumination, the worse the recovery in general.

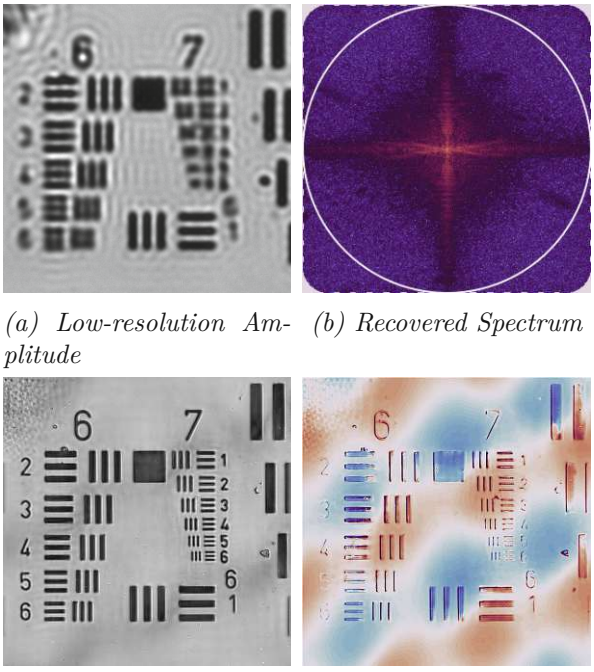
Chapter 4

Conclusion & Outlook

4.1 Fourier Ptychography

A Fourier ptychographic microscope successfully obtains super-resolution, as shown in detail in Section 3.1. A qualitative Phase Image of the sample is recovered, albeit non-trivial interpretation as elaborated in Section 2.5.

An exemplary FPM recovery is shown in Figure 4.1; spectrum (4.1b), amplitude (4.1c) and phase (4.1d), where one can clearly see the improved resolution compared to the best single low-resolution image (4.1a).



(a) Low-resolution Amplitude *(b) Recovered Spectrum*

(c) Recovered Amplitude *(d) Recovered Phase*

Figure 4.1: Examples from the FPM process; The central (best) low resolution image (4.1a), recovered high-resolution spectrum (4.1b), amplitude (4.1c) and phase (4.1d).

4.2 Alignment

The proposed methods, described in detail in Section 2.6 enable the direct study of the impact, misalignment of the illumination source has on FPM recovery, shining light on which system parameters are more critical than others. Our calibration procedures permit precise correction of the experimental setup, where possible, as shown in Section 3.2. Additionally the stated analysis allows assessment of present misalignment, even if uncorrectable by known methods.

Since FPM is a very ill-posed problem, the reconstruction of the phase heavily depends on the object itself, so it is not possible to give a strict limit where misalignment breaks the algorithm. Nevertheless we might name empirical limits based upon our simulations, up to which degree of misalignment the recovery still works.

Empirical limits of misalignment for conventional FPM, indicators of the respective mis-

alignment, and the feasibility of quantitative correction are shown in Table 4.1 for all six degrees of freedom. Even if quantitative correction of the whole panel is not possible (roll and/or pitch), our method at least provides a warning, based on a qualitative analysis.

Table 4.1: Robustness of FPM to misalignment for all six degrees of freedom; empirical limits, indication and correction feasibility.

Parameter	Lim	Indication	Correction
Shift $\Delta\theta_x$	0.2°	low spread $\Delta\theta_x$	yes
Shift $\Delta\theta_y$	0.2°	low spread $\Delta\theta_y$	yes
Shift Δz	2%	high spread $\Delta\theta_x, \Delta\theta_y$	no
Roll ρ	3°	$\gamma_x \neq \gamma_y$	no
Pitch ψ	3°	$\gamma_x \neq \gamma_y$	no
Yaw γ	2°	$\gamma_x \approx \gamma_y$	yes

To our knowledge, such an analysis was never demonstrated before.

4.3 Illumination

Our proposed methods, described in detail in Section 2.7 shed light on the impact, uniformity of the illumination source has on FPM recovery. Our one-step calibration procedures permit fast (computationally cheap) yet precise correction of the experimental setup, as shown in Section 3.3.

The complex, nonlinear nature of FPM again forbids to give a strict limit where illumination non-uniformity breaks the algorithm. Nevertheless we might name empirical limits based upon our simulations, up to which degree of misalignment the recovery still works. Based on the result shown in Section 3.3, we conclude that FPM demands a high illumination uniformity, we estimate deviations in the luminosity above 1% (±2.5 uint8) to be fatal for FPM recovery:

$$\nu_{max} = 1\% \tag{4.1}$$

Considering, one might want to correct those tangible aspects about the illumination; covering directivity, individual LED's assembly on the panel as well as luminosity.

Those have in common, that they do not change over measurements, so one may correct all takes from the same physical setup using our one-step calibration method, described in Section 2.7. This enables the automatic correction of *all* the subsequent measurements with the same physical setup.

To our knowledge, such a tool was never developed before.

In addition we still face various other sources of error: Current fluctuations during the exposure window, dynamic range and its exposure dependence, to name only a few [Zhang et al., 2019]. Since these arise during each individual measurement in a unique manner, our proposed method does not account for them.

We suggest further research in those areas, based on our findings that luminosity deviations do matter.

4.4 Calibration Comparison

So far we approacched the effect several system parameters have on FPM recovery. To conclude, we present a brief comparison of our experimental data featuring generic recovery, alignment calibration, luminosity calibration, and both in conjunction.

A juxtaposition featuring the recovered amplitude and phase is shown in Figure 4.3. As these are experimental data, we don't know the ground truth.

Surprisingly, at the point of convergence after 100 interations, all of these recoveries are quite similar. Barely visibly, the phase interferes with the amplitude, as described in much detail in Section 2.5 (dark *stairs* in the amplitude's white background).

The confidence and discordance of all four recoveries are shown together in Figure 4.2, respective Figure 4.4.

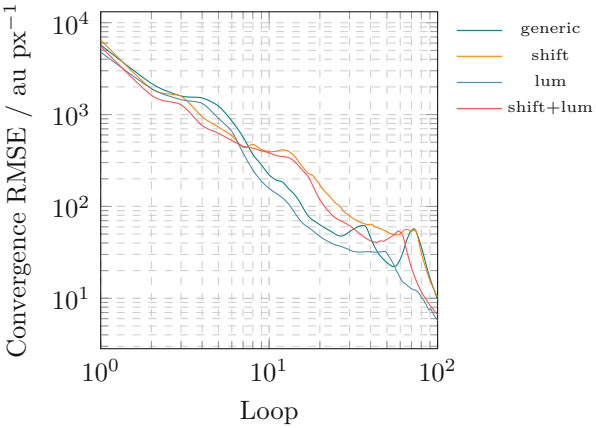


Figure 4.2: Comparison of the convergences of FPM reconstruction versus loop in double-logarithmic scale of generic data, alignment corrected (shift), luminosity corrected, and both (lower is better).

All four examples exhibit a quite similar nature, compared to our alignment & luminosity simulations in Sections 3.2, respective 3.3; which show vastly different characteristics depending on the luminosity uniformity.

Taking into account the calibrated misalignment of our experimental setup in Section 3.2, we have credible evidence that our experimental setup is already quite accurate aligned; alas, we should not anticipate drastic increases in reconstruction quality due to further alignment calibration.

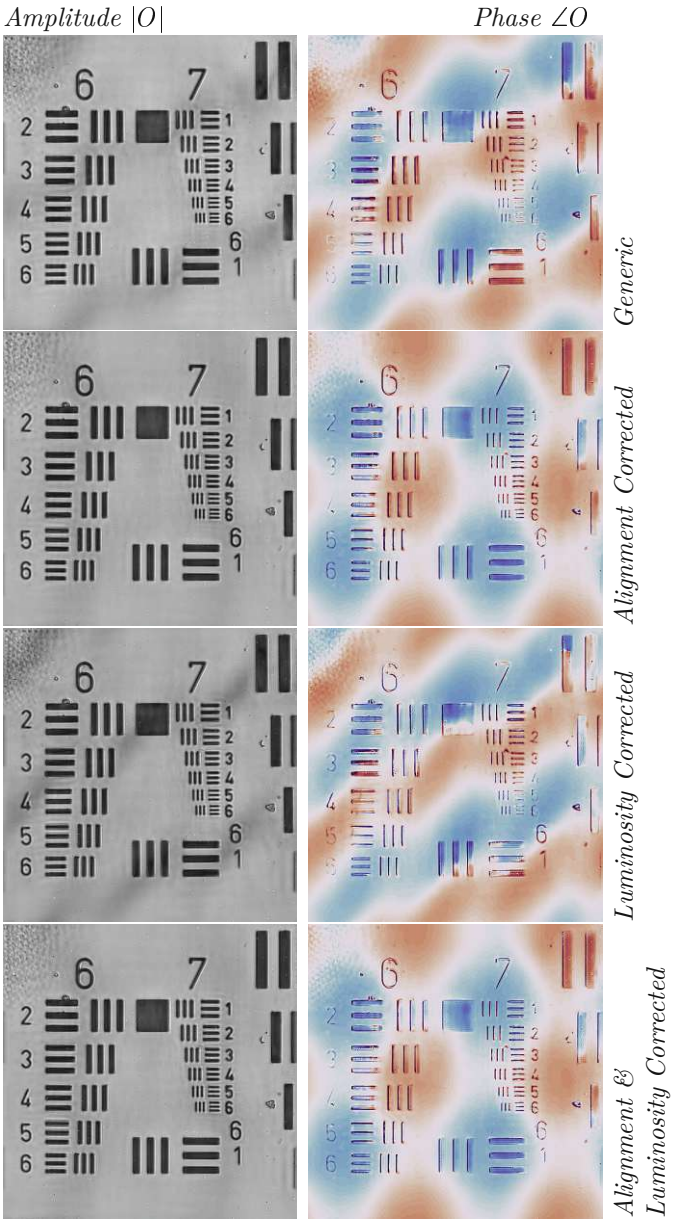


Figure 4.3: Demonstrating the results of FPM recovery of our experimental setup using the proposed methods for correction alignment and/or inconsistent illumination. Reconstructed amplitude (left) and phase (right) in respect to generic recovery (top).

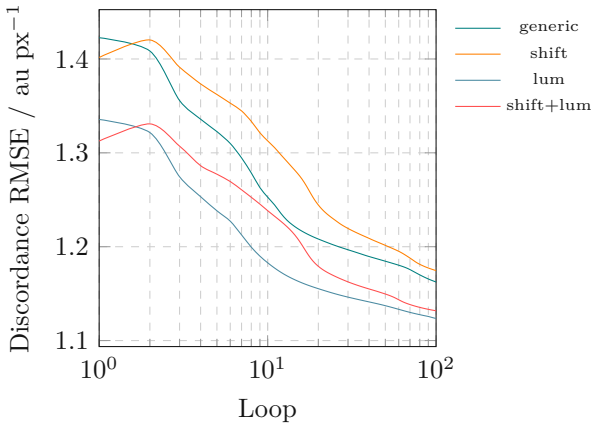


Figure 4.4: Comparison of the discordances between FPM reconstruction and low-resolution images versus loop in semi-logarithmic scale of generic data, alignment corrected (shift), luminosity corrected, and both (lower is better).

From convergence & discordance alone one might even conclude, that the alignment & luminosity calibrated recoveries perform slightly *worse* than the generic recovery—which we believe not to be the case; our physical setup is simply aligned reasonably well already.

Concluding we suggest, that our experimental setups misalignment is close to or even below our alignment calibration's measurement uncertainty; so the calibration becomes negligible.

Analogous for luminosity: From Figure 4.3 we find that luminosity-calibrating our experimental setup obviously does not enhance the results dramatically. This is a direct contradiction of our findings in Section 3.3, based on extensive simulations that even a non-uniformity of about 1% downgrades the recovery notably. Yet we suspect the true non-uniformity of the illumination to be much higher than 1%, based on the luminosity analysis in Section 3.3, even if we do not take directivity into account! Nevertheless, the effect of calibrating luminosity in Figure 4.3 seems negligible.

It is conceivable, that the improvement due to calibration on less well aligned, and less consistently lit experimental data would exceeds the performance gain shown here.

In conclusion we suggest further research on the effects of luminosity fluctuations on FPM.

4.5 Microscope Setup

To summarise, we promote the following procedure of calibration and digital post-correction for a FPM setup. For a very detailed manual calibration procedure similar to the Part 1 listed below, the interested reader is referred to [Zhang et al., 2019].

1. Mechanical Alignment of the illumination source:
   (a) Focus on the central LED: centralise camera.
   (b) Focus on the whole panel: align tilt, roll, pitch; get the LED rows and columns straight.
   (c) Focus on the object plane: align object normal to the optical path.
2. Background measurement whiteout illumination, shutter noise should be negligible.
3. BF luminosity calibration without object.
4. Luminosity calibration without object using diffuser; should be similar to (3).
5. Broad Alignment calibration:
   (a) Large search space (e.g ±20 px), big steps.
   (b) Adjust microscope setup mechanically if necessary.
   (c) Estimate magnification (from NA).
6. Fine alignment calibration:

(a) Import broad alignment corrections & corrected NA.

(b) Small search space (e.g. ± 1 px).

(c) Calibrate magnification.

7. Recover Pupil; check for aliasing or other anomalies.

8. Fast, Full FPM recovery using luminosity & alignment calibration matrices:

   (a) Sample 1.

   (b) Sample 2.

   (c) ...

   (d) Sample n.

Appendix A

Equipment

Table A.1: Equipment used for the construction and classification of the Fourier ptychographic microscope prototype.

Part	Description
Controller	Arduino Due
Illumination	RGB LED Matrix panel 32×32
Camera	Imaging Development Systems (IDS) UI-1220SE-M-GL Rev.2 CMOS Mono
Tube lens	Mitutoyo Plan APO 2x Na 0.055 Microscope Objective
Laptop	Dell E6420 Laptop running Windows 7
Oscilloscope	Agilent Technologies InfiniiVision DSO-X 2024A Digital Storage Oscilloscope
Photodiode	
Diffuser	Generic smartphone display diffuser
Test target	1951 USAF Resolution Target

Appendix B

Code

All the algorithms and programs described in this thesis are shown here. Most of them are written for GNU Octave [Eaton et al., 2020], a well documented free software similar to (and mostly compatible with) MATLAB; indicated by the file name ending with *.m* (e.g. `program.m`). Some programs are written in Python, indicated by the file name ending with *.py* (e.g. `program.py`).

For the sake of demonstration, all the code listed here shows abbreviated versions of the respective programs; a slightly polished excerpt, lacking e.g. log, plot or save commands.

A digital unabridged version of all the source code shown here is available in my git repository (unless noted otherwise) [Siegel, 2021a]:

`https://github.com/imrahilias/fourierptychography`

The preceding construction of our FPM prototype, including the programs for illumination and camera side are written in C, respective in Python. For the sake of readability, this part of the code is omitted in this thesis, since it is extensively covered in my Projektarbeit / Student Project at the TU Wien [Siegel, 2021b]. A digital version of all the source code is available in my git repository [Siegel, 2021a].

B.1 Fourier Ptychography

B.1.1 Main Program: kockpit.m

This is sort of the cockpit where FPM recovery is controlled, hence the name `kockpit.m`. All system parameters in the part have to be set correctly, in order to function! For the sake of demonstration, the code listed here is an excerpt from the program, slightly polished. All the necessary functions and routines are called from this program, and will be described in the following sections in order of appearance.

```
#!/bin/octave
## does fourier ptychography. loads a deck of lores images under
## different illumination angles, assuming spatially cohenrence (coherent
## transfer function). enters the matrix and retrieves phase information
## that the gods stored somewhere deep inside quantum mechanics.
## iteratively reconstructs a hires complex image. estimates the
## confidence of the reconstructed amplitude & phase.
## @ moritz siegel

## \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## soft settings \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
global hd = "~/fourierptychographie" # parent directory, needs trailing
 ↪  slash!
global wd = "sets/2020_01_21_183706/256"
global nn = ""; # additional name suffix (default: "")
global lxn = 170; # /px, dimensions of square lores images
global pic = 15; # 0<n<16, number lores input images considered in every
 ↪  direction.
global step = 1; # skip every step image? (default: 1)
densebrightfield = false; ## /bool, if step>1, still use step=1 in
 ↪  brightfield area? (default: true)
global ceps = 0.1; # convergence reached when mean difference per px of
 ↪  succesive iterations is less than eps ... (default: 1)
global deps = 0.1; # convergence reached when discordance with lores images
 ↪  is less than eps ... (default: 1)
global loop = 100; # ... otherwise: loop maximum (default: 10)
dx = 0; # /m, assumed dx misalignment (default: 0)
dy = 0; # /m, assumed dy misalignment (default: 0)
dz = 0; # /m, known real dz misalignment for plotting (default: 0)
roll = 0; # /deg, assumed roll misalignment (default: 0)
pitch = 0; # /deg, assumed pitch misalignment (default: 0)
yaw = 0; # /deg, assumed yaw misalignment (default: 0)
global sparsing = false; # /bool, use sparsly sampling = ignore hot & cold
 ↪  pixels? (default: false)
```

```
global hot = 255; # /uint8, hot pixel (if sparsly sampling) (default: 245)
global cold = 0; # /uint8, cold pixel (if sparsly sampling) (default: 10)
global illuminate = false; # /bool, correct brightness via calibration using imported normalisation matrix? (default: false)
global pupilla = false; # /bool, correct pupil function interatively (embedded) beware of grid aliasing due to mislignement! (default: false)
global shifting = false;  # /bool, correct misaligment by importing correction matrix? (default: false)
global live = false; # /bool, create zeitraffer from recovery? expensive! (default: false)
global confidence = false; # plot confidence map? (very expensive!) (default: false)
## hard settings ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
global r = 1; # /bool, led color
global g = 0;
global b = 0;
global ledgap = 6e-3; # /m, between leds on panel (default: 6e-3)
global ledheight = 327e-3; #327e-3; # 135e-3 /m, dist panel-object (default: 327e-3)
global wavelength = r*632e-9 + g*532e-9 + b*472e-9; # /m, wavelength of the led, 20nm bandwisth, for now! (default: r*632e-9 + g*532e-9 + b*472e-9)
global ccdpix = 2.2e-6; # /m, pixel size of the ccd, assuming quadratic pixel (default: 2.2e-6)
global na = 0.055; # numerical apertur of lens system  (default: 0.055)
global mag = 1.678; # /px, overall magnification of the optical (default: 1.678)
global xpb = 0.9; #/s, brightfield exposure, go measure (default: 0.9)
global xpd = 3; #/s, darkfield exposure, as hi as it gets (default: 3)
global led = 31; # size of led panel, wants to be odd, symmetric around ursprung (default: 31)
## end of settings: hands off ////////////////////////////////////////////
## ////////////////////////////////////////////////////////////////////////

## onwards, through the looking glass, into the dreamworld
global k0 = 2*pi/wavelength; # 1/m
global cutoff = na * k0;
global kmax = pi / ccdpix * mag; # 1/m
[kxc kyc] = meshgrid( -kmax:kmax/( (lxn-1)/2 ):kmax );
ctf = ( ( kxc .** 2 + kyc .** 2 ) < cutoff**2 ); # pupil function circ(kmax); no aberration

## geodesics
[xs ys] = skuiral( (2*pic + 1)**2, step );

```

```
## kgeodesics
global dkx = 2*kmax / lxn;
[ kx, ky, x, y, z ] = krid( roll, pitch, yaw, dx, dy, dz );
[~, ix] = max( sqrt( kx(:).**2 + ky(:).**2 ) ); # find the farthest insert position
global hxn = ceil( max( [ abs(kx(ix)) abs(ky(ix)) ] ) + lxn/2 ) * 2; # optimal high-res image size

## in the shade
global pum = ( (kx.**2 + ky.**2) < (cutoff/dkx)**2 ); # brightfield (1) or darkfield (0)?
global exposure = pum + ~pum * xpb/xpd; # exposure correction matrix

## load images
[ xs, ys, ims, ice, lava, am, sigma, spars ] = stakk( xs, ys );

## rekover
[ orec, skorec, pupil, convergence, discordance, quality ] = rekover( ims, ctf, xs, ys, kx, ky, spars, ref=[] );

## plot
pikture( orec, skorec, pupil, convergence, discordance, quality, xs, ys, ims, am, sigma, ice, lava, intersection, roll, pitch, yaw, dx, dy, dz );
```

B.1.2 Create Spiral Pattern: skuiral.m

```
#!/bin/octave
## @ moritz siegel
function [stop, go] = skuiral( len, step=1, stop=16, go=16, pos=0, side=1, pm=1, llim=1, ulim=32, corner=0 )
    ## usage: [stop, go] = skuiral( len, opt )
    ##
    ## this recursive function creates a len-by-len square spiral pattern as
    ## a load-order for led panel positions used for fourier ptychography.
    ##
    ## optional parameters:
    ##
    ## step         stepsize (default=1)
    ## stop, go     start positions (default=16,16)
    ## pos          position to start if stop,go are vectors (default=0)
    ## side         length of the side of the squiral square (default=1)
    ## pm           math pos (+1) or math neg (-1) rotation (default=+1)
```

```
    ## llim, ulim   lower, and upper limits of the squiral (default=0,31)
    ##
    ## example: [x, y] = squiral( 10 ); figure; plot( x, y, '-o');
    ##
    len = round( len );
    for p = 1 : side
        pos = pos + 1;
        next = go( pos ) - pm * step;
        if ( pos == len || next < llim || next > ulim )
            return
        endif
        stop( pos + 1 ) = stop( pos );
        go( pos + 1 ) = next;
    endfor
    if ( corner == 0 )
        [stop, go] = skuiral( len, step, go, stop, pos, side=side, pm=-pm,
        ↪  llim, ulim, corner=1 );
    else
        [stop, go] = skuiral( len, step, go, stop, pos, side=side+1, pm=pm,
        ↪  llim, ulim, corner=0 );
    endif
endfunction
```

B.1.3 Define Kspace Grid: krid.m

```
#!/bin/octave
## @ moritz siegel
function [ kx, ky, x, y, z ] = krid( roll=0,pitch=0,yaw=0,dx=0,dy=0,dz=0 )
    ## usage: [ kx, ky, x, y, z ] = krid( opt )
    ##
    ## this function creates a (led-by-led) grid in 3d real space and
    ## projects it to k-space. (x,y,z) are postition and orientation of
    ## the leds in respect to the object at (0,0,0), typically at
    ## position (0,0,-ledheight). optionally it tilts succesively in all
    ## 6 dof, in the order: roll,pitch,yaw,dx,dy,dz. where
    ## (roll,pitch,yaw) are rotations of the whole panel, whereas
    ## (dx,dy,dz) allow shifts of the individual leds, if matrices are
    ## given. (kx,ky) is the output plane in px. where km2kp is the
    ## conversion factor, that translates angular frequency /m to px of
    ## the given sensor.
    ##
    ## optional parameters:
```

```
18      ##
19      ## roll,pitch,yaw   /deg, rotation around x,y,z axis ( default = 0 )
20      ## dx,dy,dz         /m, shift in x,y,z axis ( default = 0 )
21      ##
22      ## example: [x, y, z] = krid( ); figure; mesh( x, y, z );
23
24      global led
25      global ledgap
26      global ledheight
27      global k0
28      global dkx
29
30      ## position of one row of leds
31      xv = linspace( -floor( led / 2 ), floor( led / 2 ), led ) * ledgap; # /m
32
33      ## position of all leds as grid
34      [x y] = meshgrid( xv ); # /px
35      z = zeros( led,led );
36
37      ## spaghettise matrices
38      v = [ x( : ), y( : ), z( : ) ]';
39
40      ## define angles
41      cx = cosd( roll );
42      sx = sind( roll );
43      cy = cosd( pitch );
44      sy = sind( pitch );
45      cz = cosd( yaw );
46      sz = sind( yaw );
47
48      ## rotation matrices
49      rox = [ 1, 0, 0; 0, cx, -sx; 0, sx, cx ];
50      roy = [ cy, 0, -sy; 0, 1, 0; sy, 0, cy ];
51      roz = [ cz, -sz, 0; sz, cz, 0; 0, 0, 1 ];
52
53      ## rotate succesively
54      v = rox * v; # roll
55      v = roy * v; # pitch
56      v = roz * v; # yaw
57
58      ## reshape to matrix
59      x = reshape( v( 1, : ), size( x ) );
60      y = reshape( v( 2, : ), size( x ) );
61      z = reshape( v( 3, : ), size( x ) );
```

```
62
63      ## shift
64      x = x + dx;
65      y = y + dy;
66      z = z - ledheight + dz;
67
68      # k-space
69      kx = - k0 / dkx * sin( atan( x ./ z ) );
70      ky = - k0 / dkx * sin( atan( y ./ z ) );
71
72  endfunction
```

B.1.4 Load & Stack Images: stakk.m

```
1   #!/bin/octave
2   ## @ moritz siegel
3   function [ xs, ys, ims, ice, lava, am, sigma, spars ] = stakk( xs, ys )
4       ## usage:
5       ##
6       ## creates a deck of lores images under different illuminosityination
        ↪  angles,
7       ## from lores images, and analyses mean, standard derivation, hot & cold
        ↪  pixels.
8       ##
9       ## optional parameters:
10      ##
11      ## sparse   create bool matrices indicating pixel != hot/cold
12      ##
13      ## example:
14      ##
15      ## init array & import lores images & corrluminosity & save to stack
16
17      global illuminate
18      global hd wd
19      global r g b led
20      global exposure
21      global hot cold
22      global sparsing
23
24      ## import brightness normalisation matrix
25      if ( illuminate == true )
26          try
```

```
            load luminosity.mat luminosity
        catch
            disp( "warning: luminosity matrix not found! skipping!" );
            luminosity = ones( led, led );
        end_try_catch
    else
        luminosity = ones( led, led );
    endif

    ims = cell( led, led );
    spars = cell( led, led );
    lava = ice = am = sigma = nan( led, led );
    l = 1;
    empty = 0;
    while ( l <= numel( xs ) )
        disp( sprintf( "    importing lores images: %d of %d or %d%%", l,
        ↪  numel( xs ), l/(numel( xs ))*100 ) );
        x = xs( l );
        y = ys( l );

        try
            im = single( imread( sprintf( "%s/%s/%02d%02d%1d%1d%1d.png", hd,
            ↪  wd, x-1, y-1, r, g, b ) ) );
            l = l + 1;
        catch
            empty = empty + 1;
            xs( l ) = []; # remove this position if the image doesnt exist
            ys( l ) = [];
            continue
        end_try_catch

        ## correct amplitude accordingly to respective exposure
        im = im .* exposure( x, y );

        ## normalize using imported normalisation matrix
        im = im .* luminosity( x, y );

        im = sqrt( im ); # assuming that the measured images are intensities,
        ↪  need to become amplitudes
        ims{ x, y } = single( im ); # cut image & save to stack as single
        ↪  precision

        ## detect hot & cold pixels
        warm = im < sqrt( hot );
```

```
        cool = im > sqrt( cold );
        ice( x, y ) = 1 - mean( cool(:) ); # abundance of cold pixels
        lava( x, y ) = 1 - mean( warm(:) ); # abundance of hot pixels
        if ( sparsing == true )
            spars{ x, y } = logical( warm & cool );
        else
            spars{ x, y } = logical( ones( size( im ) ) );
        endif
        clear warm cool

        try
            am( x, y )  = mean( im( spars{ x, y } == true ) );
            sigma( x, y ) = std( im( spars{ x, y } == true  ) );
        catch
            am( x, y ) = nan;
            sigma( x, y ) = nan;
        end_try_catch
        clear im

    endwhile

    if ( empty > 0 )
        disp( sprintf("warning: %d images missing! skipped.", empty ) );
    endif

    endfunction
```

B.1.5 The Core of FPM: rekover.m

```
#!/bin/octave
## @ moritz siegel

function [ orec, skorec, pupil, convergence, discordance, quality ] =
↪   rekover( ims, ctf, xs, ys, kx, ky, spars, ref=[] )
    ## usage:
    ##
    ## does fourier ptychography: enters the matrix and retrieves phase
    ↪  information,
    ## that the gods stored somewhere deep inside quantum mechanics.
    ## iteratively reconstructs a hires complex image,
    ## given a deck of lores images under different illumination angles,
    ## and assuming spatially cohenrence (coherent transfer function).
```

```
12      ## estimates the confidence of the reconstructed amplitude & phase.
13      ##
14      ## optional parameters:
15      ##
16      ## example:
17      ##
18      global lxn hxn led loop ceps deps cmts mt mts live pupilla rf
19
20      orec = ones( hxn );
21      skorec = fftshift( fft2( orec ) );
22      skorecold = skorec;
23      skref = fftshift( fft2( ref ) );
24      discordance = nan( loop,1 );
25      convergence = nan( loop,1 );
26      if ( size( ref ) != [0 0] )
27          quality = nan( loop,1 );
28      else
29          quality = nan;
30      endif
31      rmse = nan( led, led );
32      pupil = 1;
33
34      for l = 1 : loop
35          for t = 1 : numel( xs )
36              x = xs(t);
37              y = ys(t);
38              if ( live == true );
39                tit = sprintf( "loop %d\nloop %d%%\nimage %d\nimage %d%%\nx =
                    %d\ny = %d", l, round(l/loop*100), t,
                    round(t/numel(xs)*100), x, y );
40                zeitraffer( skorec, rmse, tit );
41              endif
42              im = (hxn/lxn)**2 * double( ims{ x, y } );
43              kxc = (hxn+1)/2 + kx( x, y );
44              kyc = (hxn+1)/2 + ky( x, y );
45              kxl = round( kxc - (lxn-1)/2 );
46              kxh = round( kxc + (lxn-1)/2 );
47              kyl = round( kyc - (lxn-1)/2 );
48              kyh = round( kyc + (lxn-1)/2 );
49              if ( pupilla == true )
50                  sklores1 = skorec( kxl:kxh,kyl:kyh ) .* ctf .* pupil; #
                    (lxn/hxn)**2 * ... ?
51                  lores = ifft2( ifftshift( sklores1 ) );
52                  rmse( x, y ) = sqrt( mean(mean( ( abs(lores) - im ).**2 )) );
```

```
            lores = im .* exp( 1i.*angle( lores ) ) .* spars{ x, y } +
            ↪  lores .* ~spars{ x, y };
            sklores2 = fftshift( fft2( lores ) ) .* ctf ;
            skorec( kxl:kxh,kyl:kyh ) = skorec( kxl:kxh,kyl:kyh ) + conj(
            ↪  ctf .* pupil) ./max(max(abs(ctf.*pupil).**2)) .* (
            ↪  sklores2 - sklores1 );
            pupil = pupil + conj( skorec( kxl:kxh,kyl:kyh ) ) ./
            ↪  max(max(abs(skorec( kxl:kxh,kyl:kyh )).**2)) .* (
            ↪  sklores2 - sklores1 );
        else
            sklores = (lxn/hxn)**2 * skorec( kxl:kxh,kyl:kyh ) .* ctf;
            lores = ifft2( ifftshift( sklores ) );
            rmse( x, y ) = sqrt( mean(mean( ( abs(lores) - im ).**2 )) );
            lores = im .* exp( 1i.*angle( lores ) ) .* spars{ x, y } +
            ↪  lores .* ~spars{ x, y };
            sklores = fftshift( fft2( lores ) ) .* ctf ;
            skorec( kxl:kxh,kyl:kyh ) = skorec( kxl:kxh,kyl:kyh ) .* ~ctf
            ↪  + sklores;
        endif
    endfor

    ## reconstructed complex object vs lores images
    discordance( l ) = sum( rmse( ~isnan( rmse ) ) ) / numel( xs );

    ## per loop changes in the reconstructed complex object
    convergence( l ) = sqrt( mean(mean( abs( skorec - skorecold ).**2 ))
    ↪  );
    skorecold = skorec;

    ## compare with ground truth if present
    if ( size( ref ) != [0 0] )
        quality( l,1 ) = sqrt( mean(mean( abs( skorec - skref ).**2 )) );
        orec = ifft2( ifftshift( skorec ) );

        ## estimate phase shift between ref & orec, and correct (shift)
        ↪  ref & compare
        phasespace = linspace( -pi, pi, 1000 );
        hi = hist( angle( ref(:) ), phasespace ); # histogram of phase
        [him himx] = max( hi ); # value & index of maximum in phase hist
        phi = phasespace( himx );
        ho = hist( angle( orec(:) ), phasespace ); # histogram of phase
        [hom homx] = max( ho ); # value & index of maximum in phase hist
        pho = phasespace( homx );
        dph = phi - pho; # phase shift between ref & orec
```

```
88              sref = ref .* exp( -1i .* dph ); # correct phase shift: shift ref
                ↪  to orec
89
90              ## delta & rms errata
91              dsref =  abs(orec) - abs(sref); # /uint8 delta amplitude map
92              dsrefp = angle(sref) - angle(orec); # /pi rad delta phase map
93              mdsrefp = mod( dsrefp, 2*pi); # /rad delta phase is also periodic
                ↪  in 2pi
94              mmdsrefp = min( abs(mdsrefp), abs(2*pi-mdsrefp) ); # periodic
                ↪  part 2
95
96              quality( l,2 ) = sqrt( mean( dsref(:).**2 ) ); # should be the
                ↪  same as 1
97              quality( l,3 ) = sqrt( mean( abs( dsrefp(:) ).**2 ) );
98              quality( l,4 ) = sqrt( mean( abs( mmdsrefp(:) ).**2 ) );
99              quality( l,5 ) = dph;
100         endif
101
102         #3 second quit criterium: convergence reached
103         if convergence( l ) < ceps || discordance( l ) < deps;
104             break
105         endif
106
107         disp( sprintf( "    loop %d of max %d : min %d%% , discordance = %e ,
            ↪  convergence = %e", l, loop, round((l/loop)*100), discordance(l),
            ↪  convergence( l ) ) );
108
109     endfor
110
111     orec = ifft2( ifftshift( skorec ) );
112
113     disp( sprintf( "converged with final discordance = %e , convergence = %e
        ↪  ", discordance(l), convergence( l ) ) );
114
115 endfunction
```

Minimal Working Example

For the sake of simplicity, a minimal working example of the Fourier ptychographic core program `rekover.m`, as proposed by [Zheng et al., 2013] is shown here. A full version of the source code shown here in excerpts is available in my git repository [Siegel, 2021a]:

```
## init.
orec = ones( hyn,hxn ); # out of the blue.
skorec = fftshift(fft2( orec ));
skorecold = skorec;

while
    for tt = 1:arraysize
        x = xs(tt); # get current x & y.
        y = ys(tt);
        im = double( ims{ x, y } ); # retrieve image from stack.
        kxc = (hxn+1)/2 + kxm( y, x );
        kyc = (hyn+1)/2 + kym( y, x );
        kxl = round(  kxc - (lxn-1)/2 ); # crop window in pixel dims.
        kxh = round(  kxc + (lxn-1)/2 );
        kyl = round(  kyc - (lyn-1)/2 );
        kyh = round(  kyc + (lyn-1)/2 );

        sklores = (lxn/hxn)**2 * skorec(kyl:kyh,kxl:kxh) .* ctf;
        lores = ifft2( ifftshift( sklores ) );
        lores = (hxn/lxn)**2 * im .* exp( 1i.*angle( lores ) );
        sklores = fftshift( fft2( lores ) ) .* ctf ;
        skorec(kyl:kyh,kxl:kxh) = skorec(kyl:kyh,kxl:kxh) .* ~ctf + sklores;
    endfor

    ## estimate per loop changes /px in the reconstructed complex object
    ## rms deviation to last loops skorec.
    dskorec = sqrt( sum(sum( abs( skorec - skorecold ).**2 )) /hxn/hyn );
    skorecold = skorec;

    ## second quit criterium: convergence reached.
    if dskorec < eps;
        break
    endif

endwhile

## finally drag kspace monster back to reality.
orec = ifft2( ifftshift( skorec ) );
sorec = abs(orec) / max( abs( orec(:) ) ) * 255; # scale orec to uint8.
```

Integrated Pupil Recovery

For the sake of completeness, a minimal working example an excerpt of the Fourier ptychographic main program `rekover.m`, featuring integrated pupil recovery, as proposed by [Zheng et al., 2013]. A full version of the source code shown here in excerpts is available in my git repository [Siegel, 2021a]:

```
## recover hires image & unknown pupil
orec = ones( hyn,hxn ); # out of the blue
skorec = fftshift(fft2( orec ));
skorecold = skorec;
pupil = 1; # unknown pupil function

for ll = 1:loop
    for tt = 1:arraysize

        x = xs(tt); # get current x & y
        y = ys(tt);
        im = double( ims{ x, y } ); # retrieve image from stack
        kxc = (hxn+1)/2 + kxm( y, x ); # geometric center (between pixels!)
        kyc = (hyn+1)/2 + kym( y, x );
        kxl = round(  kxc - (lxn-1)/2 ); # crop window, pixel dims
        kxh = round(  kxc + (lxn-1)/2 );
        kyl = round(  kyc - (lyn-1)/2 );
        kyh = round(  kyc + (lyn-1)/2 );

        sklores1 = skorec(kyl:kyh,kxl:kxh) .* ctf .* pupil;
        lores = ifft2( ifftshift( sklores1 ) );
        lores = (hxn/lxn)**2 * im .* exp( 1i.*angle( lores ) )
            .* spars{ x, y } + lores .* ~spars{ x, y };
        sklores2 = fftshift( fft2( lores ) ) .* ctf ;
        skorec(kyl:kyh,kxl:kxh) = skorec(kyl:kyh,kxl:kxh)
            + conj( ctf .* pupil) ./max(max(abs(ctf.*pupil).**2))
            .* ( sklores2 - sklores1 );
        pupil = pupil + conj( skorec(kyl:kyh,kxl:kxh) )
            ./ max(max(abs(skorec(kyl:kyh,kxl:kxh)).**2))
            .* ( sklores2 - sklores1 );

    endfor

    ## estimate per loop changes in the reconstructed complex object
    ## /px rms deviation to last loops skorec
    dskorec(ll,:) = sqrt( sum(sum( abs( skorec - skorecold ).**2 )) /hxn/hyn
     ↪  );
```

```
37      skorecold = skorec;
38
39      ## second quit criterium: convergence reached
40      if dskorec(ll) < eps;
41          break
42      endif
43
44  endfor
45
46  ## finally drag kspace monster back to reality
47  orec = ifft2( ifftshift( skorec ) );
48  sorec = abs(orec) ./ max( abs( orec(:) ) ) * 255; # scale orec to uint8
```

B.2 Alignment

B.2.1 Alignment Calibration: cirkles.m

The following program `cirkles.m` performs the alignment calibration explained in Section 2.6, based on [Eckert et al., 2018].

```
#!/bin/octave
## loads a deck of lores images under different illumination angles
## (fourier ptychography), assuming spatially cohenrence (coherent
## transfer function). autocorrelates the lores images to correct the
## assumed illumination positions.
## @ moritz siegel
## \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## soft settings \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
global hd = "~/fourierptychographie" # parent directory, needs trailing slash!
global wd = "sets/2021_03_09_132545_sim_wasch_512_palme_512_sc3p15na0.055ld0.327x0.003y0.007h0r0p0y0dir0rain0lum0"
global nn = ""; # additional name suffix (default: "")
global lxn = 170; # /px, dimensions of lores images
dx = 0e-3; # /m, assumed dx misalignment (default: 0)
dy = 0e-3; # /m, assumed dy misalignment (default: 0)
dz = 0e-3; # /m, known real dz misalignment for plotting (default: 0)
roll = 0; # /deg, assumed roll misalignment (default: 0)
pitch = 0; # /deg, assumed pitch misalignment (default: 0)
yaw = 0; # /deg, assumed yaw misalignment (default: 0)
blur = 2; # /px, sigma of gaussian smoothing (default: 2)
global sparsing = false; # /bool, use sparsly sampling = ignore hot & cold pixels? (default: true)
global hot = 245; # /uint8, hot pixel (if sparsly sampling) (default: 245)
global cold = 10; # /uint8, cold pixel (if sparsly sampling) (default: 10)
global illuminate = false; # /bool, correct brightness via calibration using imported normalisation matrix? (default: false)
shiftit = false;  # /bool, correct misaligment by importing correction matrix? (default: false)
plotex = false; # /bool, plot png & pdf+tex (expensive)? (default: false)
plotac = false; # /bool, plot autocorrelations? (default: false)
dxr = 0; # /px, known real dx misalignment for plotting (eg from simulation) (default: 0)
dyr = 0; # /px, known real dy misalignment for plotting (eg from simulation) (default: 0)
## hard settings ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
global r = 1; # /bool, led color
```

```
global g = 0;
global b = 0;
global ledgap = 6e-3; # /m, between leds on panel (default: 6e-3)
global ledheight = 327e-3; #327e-3; # 135e-3 /m, dist panel-object (default: 327e-3)
global wavelength = r*632e-9 + g*532e-9 + b*472e-9; # /m, wavelength of the led, 20nm bandwisth, for now! (default: r*632e-9 + g*532e-9 + b*472e-9)
global ccdpix = 2.2e-6; # /m, pixel size of the ccd, assuming quadratic pixel (default: 2.2e-6)
global na = 0.055; # numerical apertur of lens system  (default: 0.055)
global mag = 1.678; # overall magnification of the optical (default: 1.678)
global xpb = 0.9; #/s, brightfield exposure, go measure (default: 0.9)
global xpd = 3; #/s, darkfield exposure, as hi as it gets (default: 3)
global led = 31; # size of led panel, wants to be odd, symmetric around ursprung (default: 31)
## end of settings: hands off ////////////////////////////////////////////
## ////////////////////////////////////////////////////////////////////////

## onwards, through the looking glass, into the dreamworld
global k0 = 2*pi/wavelength; # 1/m
global cutoff = na * k0;
global kmax = pi / ccdpix * mag; # 1/m
[kyc kxc] = meshgrid( -kmax : kmax/( (lxn-1)/2 ) : kmax );
ctf = ( ( kxc .** 2 + kyc .** 2 ) < cutoff**2 ); # pupil function circ(kmax); no aberration

## kgeodesics
global dkx = 2*kmax / lxn;
[ kx0, ky0, x0, y0, z0 ] = krid( );
[ kx, ky, xm, ym, zm ] = krid( roll, pitch, yaw, dx, dy, dz );

## in the shade
global pum = ( (kx.**2 + ky.**2) < (cutoff/dkx)**2 ); # brightfield (1) or darkfield (0)?
global exposure = pum + ~pum * xpb/xpd; # exposure correction matrix
global shade = sin( tan( na ) * ledheight );
[xs ys] = skuiral( round(shade/ledgap+2)**2 );

## pixel vs mm
global pixgap = abs( kx( xs(1) , ys(1) ) - kx( xs(2) , ys(2) ) ); # how many pix between two adjacent kx?
global mm2px = k0  / dkx * sin( atan( 1e-3 ./ ledheight ) );

## load
```

```
[ pux, puy, ims, ice, lava, am, sigma, spars ] = stakk( pux, puy );

akims = cell( led, led );
for l = 1 : numel( pux )
    x = pux(l); # get current x & y
    y = puy(l);
    kims{ x, y } = fftshift( fft2( ims{ x, y } )); # stack ordered as cell
    amkim( :, :, l ) = abs( kims{ x, y } ); # stack as matrix for mean
endfor
makim = mean( amkim, 3 ); # global mean for normalisation
clear amkim;

## analyse
raxe = nan( led, led ); # store the estimated radii
dx = nan( led, led ); # store the local shifts accaounting to cost 1
dy = nan( led, led );
strengths = nan( led, led );
siz = sprintf( "-S%d,%d", lxn, lxn );
for ll = 1:length( pux )
    x = pux( ll ); # get current x & y
    y = puy( ll );
    iname = sprintf( "%02d%02d%1d%1d%1d", x-1, y-1, r, g, b ); # python
    ↪  starts naming with 0
    akim = abs( kims{ x, y } ); # load image
    eclipse = log10( akim ./ makim );
    eclipse = imsmooth( eclipse, "Gaussian", 1 );

    ## find all circles
    radiusse = [ cutoff/dkx - cutoff/dkx/10, cutoff/dkx + cutoff/dkx/10 ];
    [ centers, radii, lstrengths ] = imfindcircles( eclipse, radiusse,
    ↪  "Sensitivity", 1 );

    ## keep the best one
    raxe( x, y ) = radii( 1 );
    strengths( x, y ) = lstrengths( 1 );

    ## reference that dx to either (kx,ky) or (-kx,-ky)
    dx1 = centers( 1, 2 ) - (lxn+1)/2 - kx( x, y );
    dx2 = - centers( 1, 2 ) + (lxn+1)/2 - kx( x, y ) + 1;
    dy1 = centers( 1, 1 ) - (lxn+1)/2 - ky( x, y );
    dy2 = - centers( 1, 1 ) + (lxn+1)/2 - ky( x, y ) + 1;
    dst1 = sqrt( dx1.**2 + dy1.**2 );
    dst2 = sqrt( dx2.**2 + dy2.**2 );
    if ( dst1 < dst2 )
```

```
        dx( x, y ) = dx1;
        dy( x, y ) = dy1;
    else
        dx( x, y ) = dx2;
        dy( x, y ) = dy2;
    endif

endfor
kx2 = kx0 + dx;
ky2 = ky0 + dy;
x2 = tan( asin( kx2 * dkx / k0 ) ) * ledheight;
y2 = tan( asin( ky2 * dkx / k0 ) ) * ledheight;
dx2 = x2 - x0;
dy2 = y2 - y0;

dx3 = dx2( ~isnan( dx2 ));
dy3 = dy2( ~isnan( dy2 ));
disp( sprintf( "mean dx2 +- std dx2  = %d +- %d",  mean( dx3 ), std( dx3 )
↪  ));
disp( sprintf( "mean dy2 +- std dy2  = %d +- %d",  mean( dy3 ), std( dy3 )
↪  ));
disp( sprintf( "median dx2 +- mad dx2  = %d +- %d",  median( dx3 ), median(
↪  abs( dx3 - median( dx3 ) ) ) ));
disp( sprintf( "median dy2 +- mad dy2  = %d +- %d",  median( dy3 ), median(
↪  abs( dy3 - median( dy3 ) ) ) ));

ra3 = raxe( ~isnan( raxe ));
disp( sprintf( "mean radius = %d +- %d",  mean( ra3 ), std( ra3 ) ));
disp( sprintf( "median radius = %d +- %d",  median( ra3 ), median( abs( ra3 -
↪  median( ra3 ) ) ) ));

magnew = na*lxn*ccdpix/wavelength/median( ra3 );
disp( sprintf( "magnification (from median radius) = %d",  magnew ));

## inclination xaxis vs x0axis
for m = 1:size(kx,1)
    xline = kx2(m,:);
    if ~any( xline ); continue; endif
    xline = xline( ~isnan( xline ));
    yline = ky2(m,:);
    if ~any( yline ); continue; endif
    yline = yline( ~isnan( yline ));
    xline2 = [ xline', ones( length(xline) , 1 ) ]; # non zero intercept
    ↪  (offset?)
```

```
148     b = xline2 \ yline'; # linfit
149     plot3( xline', xline2 * b, ones(size(xline')) );
150     patchwork = [ patchwork', [xline', (xline2 * b)]' ]';
151     gamma( m ) = atand( b(1) ); # fitted
152     #alpha( m ) = 90 + atand(( xline(1) - xline(end) )/( yline(end) -
        ↪  yline(1) )); # raw
153 endfor
154 gammax = gamma( gamma != 0 );
155 disp( sprintf( "mean gammax +- std gammax  = %d +- %d",  mean( gammax ), std(
    ↪  gammax ) ));
156 disp( sprintf( "median gammax +- mad gammax  = %d +- %d",  median( gammax ),
    ↪  median( abs( gammax - median( gammax ) ) ) ));
157
158 ## inclination yaxis vs y0axis
159 for m = 1:size(kx,1)
160     xline = ky2(:,m)';
161     if ~any( xline ); continue; endif
162     xline = xline( ~isnan( xline ));
163     yline = kx2(:,m)';
164     if ~any( yline ); continue; endif
165     yline = yline( ~isnan( yline ));
166     xline2 = [ xline', ones( length(xline), 1 ) ]; # non zero intercept
        ↪  (offset?)
167     b = xline2 \ yline'; # linfit
168     plot3( xline2 * b, xline', ones(size(xline')) )
169     patchwork = [ patchwork', [(xline2 * b), xline']' ]';
170     gamma( m ) = -atand( b(1) ); # fitted
171     #alpha( m ) = 90 + atand(( xline(1) - xline(end) )/( yline(end) -
        ↪  yline(1) )); # raw
172 endfor
173 gammay = gamma( gamma != 0 );
174 disp( sprintf( "mean gammay +- std gammay  = %d +- %d",  mean( gammay ), std(
    ↪  gammay ) ));
175 disp( sprintf( "median gammay +- mad gammay  = %d +- %d",  median( gammay ),
    ↪  median( abs( gammay - median( gammay ) ) ) ));
```

B.2.2 Alignment Simulation: simalign.m

The following program `simalign.m` performs the alignment simulation explained in Section 2.6.1, based on B.2.1.

```
#!/bin/octave
## simulates the forward imaging process of Fourier ptychography.  create
## a deck of lores images via a coherent transfer function, simulating
## different illumination angles.  enters the matrix and retrieves phase
## information that god stored somewhere deep inside quantum mechanics.
## iteratively reconstructs a hires complex image.  estimates the impact
## of various misalignments on the reconstruction by respective deviation
## to reference image.  checks and balances are optimised for only one
## varying dof, though one could so test up to all six dof
## simultainously.
## @ moritz siegel
## \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## soft settings \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
hd = "~/fourierptychographie" # parent directory
wd = "sims/alignment"
aa = "dschungel_512"; # amplitude image name
pp = "wasch_512"; # phase image name
nn = ""; # additional name of simulation
scale = 3; # resolution scaling (per dimension)
global pic = 15; # 0<n<16, number lores input images considered in every
 ↪  direction.
global step = 1; # skip every step image? (default: 1)
densebrightfield = false; ## /bool, if step>1, still use step=1 in
 ↪  brightfield area? (default: true)
global ceps = 1e-1; # convergence reached when mean difference per px of
 ↪  succesive iterations is less than eps ... (default: 1)
global aeps = 1e-1; # convergence reached when discordance with lores images
 ↪  is less than eps ... (default: 1)
global loop = 10; # ... otherwise: loop maximum (default: 10)
dings = round( [ 0 logspace(0,1,5) ] ); # variation magnitude vector
dx = 0e-3 # /m, assumed dx misalignment (default: 0)
dy = 0e-3 # /m, assumed dy misalignment (default: 0)
dz = 0e-3 # /m, known real dz misalignment for plotting (default: 0)
roll = 0 # /deg, assumed roll misalignment (default: 0)
pitch = 0 # /deg, assumed pitch misalignment (default: 0)
yaw = 0 # /deg, assumed yaw misalignment (default: 0)
global sparsing = false; # /bool, use sparsly sampling = ignore hot & cold
 ↪  pixels? (default: true)
global hot = 245; # /uint8, hot pixel (if sparsly sampling) (default: 245)
global cold = 10; # /uint8, cold pixel (if sparsly sampling) (default: 10)
global pupilla = true; # /bool, correct pupil function interatively
 ↪  (embedded) beware of grid aliasing due to mislignement! (default: false)
```

```
global shifting = false;  # /bool, correct misaligment by importing correction matrix? (default: false)
global live = false; # /bool, create zeitraffer from recovery? expensive! (default: false)
global confidence = false; # plot confidence map? (very expensive!) (default: false)
phaseshift = true; # shift phase image for better comparison with imref?
## hard settings ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
global r = 1; # /bool, led color
global g = 0;
global b = 0;
global ledgap = 6e-3; # /m, between leds on panel (default: 6e-3)
global ledheight = 327e-3; #327e-3; # 135e-3 /m, dist panel-object (default: 327e-3)
global wavelength = r*632e-9 + g*532e-9 + b*472e-9; # /m, wavelength of the led, 20nm bandwisth, for now! (default: r*632e-9 + g*532e-9 + b*472e-9)
global ccdpix = 2.2e-6; # /m, pixel size of the ccd, assuming quadratic pixel (default: 2.2e-6)
global na = 0.055; # numerical apertur of lens system  (default: 0.055)
global mag = 1.678; # /px, overall magnification of the optical (default: 1.678)
global xpb = 0.9; #/s, brightfield exposure, go measure (default: 0.9)
global xpd = 3; #/s, darkfield exposure, as hi as it gets (default: 3)
global led = 31; # size of led panel, wants to be odd, symmetric around ursprung (default: 31)
## end of settings: hands off ////////////////////////////////////////////
## ////////////////////////////////////////////////////////////////////////

## simulate the high resoluiton complex image
IMAGE_PATH( strcat( hd, "/kulissen" ))
imrefa = double(imread( aa, "png" )); # pics better be pngs!
imrefp = double(imread( pp, "png" ));
imrefp = 2*pi*imrefp./max(max(imrefp)) - pi; # phase ranges -pi to pi
imref = imrefa .* exp( 1i .* imrefp );
global hxn = size( imref, 2 );
skimref = fftshift( fft2( imref ) );

## onwards, through the looking glass, into the dreamworld
global k0 = 2*pi/wavelength; # 1/m
global cutoff = na * k0;
global kmax = pi / ccdpix * mag; # 1/m
global lxn = floor( hxn/scale );
global dkx = 2*kmax / lxn;
[kxc kyc] = meshgrid( -kmax:kmax/( (lxn-1)/2 ):kmax );
```

```
ctf = ( ( kxc .** 2 + kyc .** 2 ) < cutoff**2 ); # pupil function circ(kmax); no aberration

## geodesics
global shade = sin( tan( na ) * ledheight );
[xs ys] = skuiral( (2*pic + 1)**2, step );
[ kx0, ky0, x0, y0, z0 ] = krid( );

##variate: recover high resolution image under varying misalignment
for dd = 1 : length( dings );
  close all # close figures to be able to redraw
  ding = dings( dd ); # generate wrong assumtion of led positions
  [ kx, ky, xm, ym, zm ] = krid( roll*ding, pitch*ding, yaw*ding, dx*ding, dy*ding, dz*ding );

  ## in the shade
  global pum = ( (kx.**2 + ky.**2) < (cutoff/dkx)**2 ); # brightfield (1) or darkfield (0)?
  global exposure = pum + ~pum * xpb/xpd; # exposure correction matrix

  ## create: lores ipsus
  skorec = fftshift(fft2( imref ));
  lava = zeros( size( kx ));
  ice = zeros( size( kx ));
  for ll = 1 : numel( xs ) # generate lores images
    x = xs(ll); # get current x & y
    y = ys(ll);
    skxc = (hxn+1)/2 + kx( y, x ); # geometric center (between pixels!)
    skyc = (hxn+1)/2 + ky( y, x );
    skyl = floor( skyc - (lxn-1)/2 ); # crop window, pixel dims
    skyh = floor( skyc + (lxn-1)/2 );
    skxl = floor( skxc - (lxn-1)/2 );
    skxh = floor( skxc + (lxn-1)/2 );
    kim = (lxn/hxn)^2 * skorec(skyl:skyh,skxl:skxh) .* ctf;
    im = ifft2( ifftshift( kim ) );
    im = uint8( abs( im ) ); # write lores image to stack resembling sensor data in uint8
    ims{ x, y } = im;
    warm = im < hot;
    cool = im > cold;
    ice( x, y ) = 1 - mean( cool(:) ); # abundance of cold pixels
    lava( x, y ) = 1 - mean( warm(:) ); # abundance of hot pixels
    if ( sparsing == true )
      spars{ x, y } = warm || cool;
```

```
113       else
114         spars{ x, y } = ones( size( im ) );
115       endif
116       am( x, y )  = mean( im( spars{ x, y } != 0 ) );
117       sigma( x, y ) = std( im( spars{ x, y } != 0 ) );
118     endfor
119
120     ## recover
121     [ kx, ky, xm, ym, zm ] = krid();
122     [ orec, skorec, pupil, convergence, discordance, quality ] = rekover( ims,
        ↪  ctf, xs, ys, kx, ky, spars, ref=imref );
123
124     ## combine to complex object again
125     soreca = abs(orec) ./ max(max( abs( orec )) ) * 255; # scale orec to uint8
126     sorec = soreca .* exp( 1i .* angle(orec) );
127
128   endfor
```

B.3 Luminosity

B.4 Luminosity Calibration: kalibright.m

The following program `kalibright.m` performs our proposed brightness calibration, explained in Section 2.7.

```
#!/bin/octave
## calibrates the individual led's brightness of the led panel used for
## fourier ptychography. loads all lores images and corrects for
## respective exposure. calculates mean & sigma, and constructs a
## brightness normalisation matrix; either for brightfield when using
## notarget calibration only, or full led panel when using a diffuser
## target.
## @ moritz siegel
## \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## soft settings \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
global hd = "~/fourierptychographie" # parent directory
#global wd = "sets/2020_05_20_014735_brightness_r/512"
global wd = "sets/2020_06_02_113704_diffused_r/512"
global nn = ""; # additional luminosity suffix  (default: "")
dx = 0e-3; # /m, assumed dx misalignment (default: 0)
dy = 0e-3; # /m, assumed dy misalignment (default: 0)
dz = 0e-3; # /m, known real dz misalignment for plotting (default: 0)
roll = 0; # /deg, assumed roll misalignment (default: 0)
pitch = 0; # /deg, assumed pitch misalignment (default: 0)
yaw = 0; # /deg, assumed yaw misalignment (default: 0)
global diffused = false; # /bool, diffuser target: full led panel (true) / no
↪  target: bf only (false) (default: false)
global sparsing = false; # /bool, use sparsly sampling = ignore hot & cold
↪  pixels? (default: true)
global hot = 250; # /uint8, hot pixel (default: 255)
global cold = 5; # /uint8, cold pixel (default: 0)
## hard settings ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
global r = 1; # /bool, led color
global g = 0;
global b = 0;
global ledgap = 6e-3; # /m, between leds on panel (default: 6e-3)
global ledheight = 327e-3; #327e-3; # 135e-3 /m, dist panel-object (default:
↪  327e-3)
global wavelength = r*632e-9 + g*532e-9 + b*472e-9; # /m, wavelength of the
↪  led, 20nm bandwisth, for now! (default: r*632e-9 + g*532e-9 + b*472e-9)
```

```
global ccdpix = 2.2e-6; # /m, pixel size of the ccd, assuming quadratic pixel (default: 2.2e-6)
global na = 0.055; # numerical apertur of lens system  (default: 0.055)
global mag = 1.678; # /px, overall magnification of the optical (default: 1.678)
global xpb = 0.9; #/s, brightfield exposure, go measure (default: 0.9)
global xpd = 3; #/s, darkfield exposure, as hi as it gets (default: 3)
global led = 31; # size of led panel, wants to be odd, symmetric around ursprung (default: 31)
## end of settings: hands off ////////////////////////////////////////////
## ////////////////////////////////////////////////////////////////////////

## onwards, through the looking glass, into the dreamworld
global k0 = 2*pi/wavelength; # 1/m
global cutoff = na * k0;
global kmax = pi / ccdpix * mag; # 1/m

## in the shade
[xs ys] = skuiral( led**2 ); # spiralizing vector, centered at (16,16)
xv = linspace(-floor(led/2), floor(led/2), led ) * ledgap; # /m distance to center
global shade = sin( tan( na ) * ledheight );
[xm ym] = meshgrid( xv );
global brightfield = ( (xm.**2 + ym.**2) < shade.**2 ); # brightfield (1) or darkfield (0)?
global exposure = brightfield + ~brightfield * xpb/xpd; # exposure correction matrix

## load
global illuminate = false; # has to be, cause we do it now!
[ xs, ys, ims, ice, lava, am, sigma, spars ] = stakk( xs, ys );

## kgeodesics
lxn = size( ims{ xs(1), ys(1) }, 1 )
global dkx = 2*kmax / lxn;
[ kx0, ky0, x0, y0, z0 ] = krid( );
[ kx, ky, xm, ym, zm ] = krid( roll, pitch, yaw, dx, dy, dz );

## compile brightness normalisation matrix
if ( diffused == true ) # normalise full panel to full panel amplitude mean
    luminosity = mean( am( isfinite( am ) ) ) ./ am;
    luminosity( luminosity > 10 ) = 10; # upper bound on how much brighter is plausible
else # normalise bf val to bf amplitude mean
```

```
    luminosity = mean( am( isfinite( am ) & brightfield ) ) ./ am .*
    ↪  brightfield  + ~brightfield;
endif

## calibrate: patchwork & normalise
bright = round( shade/ledgap );
nice = nlava = nam = nsigma = nan( size( am ));
for tt = 1:length(xs)
    x = xs(tt); # get current x & y
    y = ys(tt);
    im = ims{ x, y };
    nim = im * luminosity( x, y ); # normalise brightness

    if ( abs(x-16) < bright+1 ) && ( abs(y-16) < bright+1 ) # only patch bf
    ↪  +1, the rest is black anyway
        xnp = floor( lxn/(2*bright+1) );
        xp = max(1, (x-16+bright)*xnp); # position in patchwork, bf only
        yp = max(1, (y-16+bright)*xnp);
        patch( yp:yp+xnp, xp:xp+xnp ) = im( yp:yp+xnp, xp:xp+xnp );
        npatch( yp:yp+xnp, xp:xp+xnp ) = nim( yp:yp+xnp, xp:xp+xnp );
    endif

    cool = im > cold;
    warm = im < hot;
    nice( x, y ) = 1 - mean( cool(:) ); # abundance of cold pixels
    nlava( x, y ) = 1 - mean( warm(:) ); # abundance of hot pixels
    nam( x, y )  = mean( nim( spars{ x, y } == 1 ) );
    nsigma( x, y ) = std( nim( spars{ x, y } == 1 ) );

    clear im nim
endfor
```

B.4.1 Luminosity Simulation: simlum.m

The following program `simlum.m` performs the alignment simulation explained in Section 2.7.1, based on B.4.

```
#!/bin/octave
## simulates the forward imaging process of Fourier ptychography.  create
## a deck of lores images via a coherent transfer function, simulating
## different illumination angles.  enters the matrix and retrieves phase
```

```
## information that god stored somewhere deep inside quantum mechanics.
## iteratively reconstructs a hires complex image.  estimates the impact
## of various misalignments on the reconstruction by respective deviation
## to reference image.  checks and balances are optimised for only one
## varying dof, though one could so test up to all six dof
## simultainously.
@ moritz siegel
## \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## soft settings \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
hd = "~/fourierptychographie" # parent directory
wd = "sims/alight"
aa = "wasch_512"; # amplitude image name
pp = "palme_512"; # phase image name
nn = ""; # additional name of simulation
scale = 3; # resolution scaling (per dimension)
global pic = 15; # 0<n<16, number lores input images considered in every
 ↪  direction.
global step = 1; # skip every step image? (default: 1)
densebrightfield = false; ## /bool, if step>1, still use step=1 in
 ↪  brightfield area? (default: true)
global ceps = 1e-3; # convergence reached when mean difference per px of
 ↪  succesive iterations is less than eps ... (default: 1)
global aeps = 1e-3; # convergence reached when discordance with lores images
 ↪  is less than eps ... (default: 1)
global loop = 1; # ... otherwise: loop maximum (default: 10)
dings = [ 0 logspace(-1,2,10) ]; # variation magnitude vector
dx = 0 # /m, assumed dx misalignment (default: 0)
dy = 0 # /m, assumed dy misalignment (default: 0)
dz = 0 # /m, known real dz misalignment for plotting (default: 0)
roll = 0 # /deg, assumed roll misalignment (default: 0)
pitch = 0 # /deg, assumed pitch misalignment (default: 0)
yaw = 0 # /deg, assumed yaw misalignment (default: 0)
global directional = true;# /bool, simulate directivity of 40 deg? (default:
 ↪  true)
global export = false; # /bool, export low-resolution images?
global sparsing = false; # /bool, use sparsly sampling = ignore hot & cold
 ↪  pixels? (default: true)
global hot = 245; # /uint8, hot pixel (if sparsly sampling) (default: 245)
global cold = 10; # /uint8, cold pixel (if sparsly sampling) (default: 10)
global pupilla = true; # /bool, correct pupil function interatively
 ↪  (embedded) beware of grid aliasing due to mislignement! (default: false)
global shifting = false;  # /bool, correct misaligment by importing
 ↪  correction matrix? (default: false)
```

```
global live = false; # /bool, create zeitraffer from recovery? expensive! (default: false)
global confidence = false; # plot confidence map? (very expensive!) (default: false)
phaseshift = true; # shift phase image for better comparison with imref?
## hard settings ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
global r = 1; # /bool, led color
global g = 0;
global b = 0;
global ledgap = 6e-3; # /m, between leds on panel (default: 6e-3)
global ledheight = 327e-3; #327e-3; # 135e-3 /m, dist panel-object (default: 327e-3)
global wavelength = r*632e-9 + g*532e-9 + b*472e-9; # /m, wavelength of the led, 20nm bandwisth, for now! (default: r*632e-9 + g*532e-9 + b*472e-9)
global ccdpix = 2.2e-6; # /m, pixel size of the ccd, assuming quadratic pixel (default: 2.2e-6)
global na = 0.055; # numerical apertur of lens system  (default: 0.055)
global mag = 1.678; # /px, overall magnification of the optical (default: 1.678)
global xpb = 0.9; #/s, brightfield exposure, go measure (default: 0.9)
global xpd = 3; #/s, darkfield exposure, as hi as it gets (default: 3)
global led = 31; # size of led panel, wants to be odd, symmetric around ursprung (default: 31)
## end of settings: hands off ////////////////////////////////////////////
## ////////////////////////////////////////////////////////////////////////

## simulate the high resoluiton complex image
IMAGE_PATH( strcat( hd, "/kulissen" ))
imrefa = double(imread( aa, "png" )); # pics better be png's!
imrefp = double(imread( pp, "png" ));
imrefp = 2*pi*imrefp./max(max(imrefp)) - pi; # phase ranges -pi to pi
imref = imrefa .* exp( 1i .* imrefp );
global hxn = size( imref, 2 );
skimref = fftshift( fft2( imref ) );

## onwards, through the looking glass, into the dreamworld
global k0 = 2*pi/wavelength; # 1/m
global cutoff = na * k0;
global kmax = pi / ccdpix * mag; # 1/m
global lxn = floor( hxn/scale );
global dkx = 2*kmax / lxn;
[kxc kyc] = meshgrid( -kmax:kmax/( (lxn-1)/2 ):kmax );
ctf = ( ( kxc .** 2 + kyc .** 2 ) < cutoff**2 ); # pupil function circ(kmax); no aberration
```

```
76
77  ## geodesics
78  [xs ys] = skuiral( (2*pic + 1)**2, step );
79  [ kx0, ky0, x0, y0, z0 ] = krid( );
80  [ kx, ky, xm, ym, zm ] = krid( roll, pitch, yaw, dx, dy, dz );
81
82  ## in the shade
83  global pum = ( (kx.**2 + ky.**2) < (cutoff/dkx)**2 ); # brightfield (1) or
    ↪  darkfield (0)?
84  global exposure = pum + ~pum * xpb/xpd; # exposure correction matrix
85
86  ## create: lores ipsus
87  skorec = fftshift(fft2( imref ));
88  lava = zeros( size( kx ));
89  ice = zeros( size( kx ));
90  for ll = 1 : numel( xs ) # generate lores images
91      x = xs(ll); # get current x & y
92      y = ys(ll);
93      skxc = (hxn+1)/2 + kx( y, x ); # geometric center (between pixels!)
94      skyc = (hxn+1)/2 + ky( y, x );
95      skyl = floor( skyc - (lxn-1)/2 ); # crop window, pixel dims
96      skyh = floor( skyc + (lxn-1)/2 );
97      skxl = floor( skxc - (lxn-1)/2 );
98      skxh = floor( skxc + (lxn-1)/2 );
99      kim = (lxn/hxn)^2 * skorec(skyl:skyh,skxl:skxh) .* ctf;
100     im = ifft2(ifftshift( kim ));
101     im = uint8( abs( im ) ); # write lores image to stack resembling sensor
        ↪  data in uint8
102     ims{ x, y } = im;
103 endfor
104
105 ## directivity 40 deg
106 if ( directional == true )
107     [mx my] = meshgrid( linspace( -15*ledgap, 15*ledgap, led ) );
108     dis = sqrt( mx.**2 + my.**2 );
109     alpha = atand( dis / ledheight );
110     directivity = cosd( alpha*3 ); # empirical led directivity of 40 deg
111 else
112     directivity = ones( size ( kx ) );
113 endif
114
115 ## variate & recover high resolution image under varying misalignment
116 for dd = 1 : numel( dings );
117     close all # close figures to be able to redraw
```

```
    ding = dings( dd ); # generate wrong assumtion of led positions

    ## add uint8 rain
    rain = ding * randn( size ( kx ) );

    ## variate
    am = nan( size( kx ) );
    sigma = nan( size( kx ) );
    for ll = 1 : numel( xs ) # generate lores images
        x = xs(ll); # get current x & y
        y = ys(ll);
        im = ims{ x, y };
        im = im + rain( x, y );
        im = im .* directivity( x, y );
        im = uint8( im );
        ims{ x, y } = im;
        warm = im < hot;
        cool = im > cold;
        ice( x, y ) = 1 - mean( cool(:) ); # abundance of cold pixels
        lava( x, y ) = 1 - mean( warm(:) ); # abundance of hot pixels
        if ( sparsing == true )
            spars{ x, y } = warm || cool;
        else
            spars{ x, y } = ones( size( im ) );
        endif
        am( x, y )  = mean( im( spars{ x, y } != 0 ) );
        sigma( x, y ) = std( im( spars{ x, y } != 0 ) );
    endfor

    ## recover
    [ orec, skorec, pupil, convergence, discordance, quality ] = rekover(
    ↪  ims, ctf, xs, ys, kx, ky, spars, ref=imref );
    ## combine to complex object again
    soreca = abs(orec) ./ max(max( abs( orec )) ) * 255; # scale orec to
    ↪  uint8
    sorec = soreca .* exp( 1i .* angle(orec) );
endfor
```

Bibliography

[22sm22, 2018] 22sm22 (2018). Optical setup for fourier ptychography, `https://en.wikipedia.org/wiki/File:Optical_setup_for_Fourier_ptychography.png`.

[Cooley and Tukey, 1965] Cooley, J. and Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

[Dong et al., 2014] Dong, S., Horstmeyer, R., Shiradkar, R., Guo, K., Ou, X., Bian, Z., Xin, H., and Zheng, G. (2014). Aperture-scanning fourier ptychography for 3d refocusing and super-resolution macroscopic imaging. *Opt. Express*, 22(11):13586–13599.

[Eaton et al., 2020] Eaton, J. W., Bateman, D., Hauberg, S., and Wehbring, R. (2020). *GNU Octave version 6.1.0 manual: a high-level interactive language for numericalcomputations.*

[Eckert et al., 2018] Eckert, R., Phillips, Z. F., and Waller, L. (2018). Efficient illumination angle self-calibration in fourier ptychography. *Appl. Opt.*, 57(19):5434–5442.

[Fienup, 1982] Fienup, J. R. (1982). Phase retrieval algorithms: a comparison. *Appl. Opt.*, 21(15):2758–2769.

[Gerchberg and Saxton, 1972] Gerchberg, R. W. and Saxton, W. O. (1972). A practical algorithm for the determination of the phase from image and diffraction plane pictures. *Optik*, 35(2):237.

[Ou et al., 2013] Ou, X., Horstmeyer, R., Yang, C., and Zheng, G. (2013). Quantitative phase imaging via fourier ptychographic microscopy. *Opt. Lett.*, 38(22):4845–4848.

[Randall, 2011a] Randall, M. (2011a). xkcd; a webcomic of romance, sarcasm, math, and language, `https://imgs.xkcd.com/comics/complex_conjugate.png`.

[Randall, 2011b] Randall, M. (2011b). xkcd; a webcomic of romance, sarcasm, math, and language, `https://imgs.xkcd.com/comics/machine_learning.png`.

[Siegel, 2021a] Siegel, M. (2021a). Fourierptychographie, `https://github.com/imrahilias/fourierptychography`.

[Siegel, 2021b] Siegel, M. (2021b). A programmable annular illumination source for a fourier ptychographic microscope, `https://github.com/imrahilias/fourierptychography`.

[Siegel et al., 2021] Siegel, M., Traxler, L., and Ginner, L. (2021). Robustness of fourier ptychographic imaging to variation of system parameters. *Electronic Imaging*, 2021(15):166–1–166–8.

[Sun et al., 2018] Sun, J., Zuo, C., Zhang, J., Fan, Y., and Chen, Q. (2018). High-speed fourier ptychographic microscopy based on programmable annular illuminations. *Scientific Reports*, 8(1):7669.

[Yeh et al., 2015] Yeh, L.-H., Dong, J., Zhong, J., Tian, L., Chen, M., Tang, G., Soltanolkotabi, M., and Waller, L. (2015). Experimental robustness of fourier ptychography phase retrieval algorithms. *Opt. Express*, 23(26):33214–33240.

[Zhang et al., 2019] Zhang, S., Zhou, G., Wang, Y., Hu, Y., and Hao, Q. (2019). A simply equipped fourier ptychography platform based on an industrial camera and telecentric objective. *Sensors*, 19(22):4913.

[Zheng, 2016] Zheng, G. (2016). *Fourier Ptychographic Imaging*. 2053-2571. Morgan & Claypool Publishers.

[Zheng et al., 2013] Zheng, G., Horstmeyer, R., and Yang, C. (2013). Wide-field, high-resolution fourier ptychographic microscopy. *Nature Photonics*, 7(9):739–745.

[Zhou et al., 2018] Zhou, A., Wang, W., Chen, N., Lam, E. Y., Lee, B., and Situ, G. (2018). Fast and robust misalignment correction of fourier ptychographic microscopy for full field of view reconstruction. *Opt. Express*, 26(18):23661–23674.

List of Tables

List of Figures

List of Algorithms

Acronyms

AOI area of interest. 21, 38

BF bright-field. 16

CCD charge-coupled device. 10,

CMOS complementary metal-oxide-semiconductor. 10,

DC zero-order term of Fourier spectrum. 28

DF dark-field. 30

DFT discrete Fourier transform. 10

DOF degrees of freedom. 25

FFT fast Fourier transform. 10, 29

FOV field of view. 20, 38

FP Fourier ptychography.

FPM Fourier ptychographic microscopy. 9

FT Fourier transform. 10

FWHM full width half maximum. 20, 21

GS Gerchberg-Saxton. 9, 11, 14

HDR high-dynamic-range. 24

LED light emitting diode. 13

MAD median absolute deviation. 43, 46

NA numerical aperture. 18

OTF optical transfer function. 14

PSF point spread function. 14

RANSAC random sample consensus. 30

RGB red green blue.

RMS root mean square. 16

RMSE root mean square error. 16, 17

SBP space–bandwidth product.

SMD surface-mount-device. 19, 21

STD sample standard deviation. 43, 46

Symbols

A amplitude of the high resolution image in real space $\mathbb{R}$.

D largest dimension of the light source.

E^2 squared error.

I Intensity of the high resolution image in real space $\mathbb{R}$.

I_{led} intensity of a light source.

M dimension of the space $\mathbb{R}^M, \mathbb{C}^M$.

N number of elements (images in image stack).

O complex object of the high resolution image in real space $\mathbb{R}$.

R_γ yaw rotation matrix, rotations about the z-axis.

R_ψ pitch rotation matrix, rotations about the y-axis.

R_ρ roll rotation matrix, rotations about the x-axis.

$\Delta\lambda$ spatial coherence (FWHM).

$\Delta\theta$ shift/rotation expressed as change in illumination angle.

Δr allowed radius deviation of the pupil function in low resolution Fourier space.

Δx lateral shift of the illumination source in x-axis.

Δy lateral shift of the illumination source in y-axis.

Δz lateral shift of the illumination source in z-axis.

Λ attenuation of a light source.

Φ phase of the high resolution image in real space $\mathbb{R}$.

α arbitrary misalignments for simulation.

$\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_n$ position of the found circles in low resolution Fourier space.

$\mathbf{k}$ M-dimensional spatial frequency coordinate.

$\mathbf{r}$ M-dimensional spatial coordinate.

β slope of the horizontal and vertical lines of the estimated grid.

$\boldsymbol{\Lambda_b}$ bright-field luminosity correction coefficients vector; Λ_n of all n bright-field images.

$\boldsymbol{\Lambda}$ luminosity correction coefficients vector; Λ_n of all n images.

$\boldsymbol{\iota}$ image vector; all images ι_n.

$\boldsymbol{\mu}$ mean intensity vector; μ_n of all n images.

δ directivity of a light source.

ℓ iteration counter.

η number of elements (pixels) of the low resolution object.

γ yaw, angle of rotations about the z-axis.

$\hat{A}$ amplitude of the high resolution image in Fourier space $\mathbb{C}$.

$\hat{I}$ Intensity of the high resolution image in Fourier space $\mathbb{C}$.

$\hat{O}$ complex object of the high resolution image in Fourier space $\mathbb{C}$.

$\hat{P}$ pupil function in high resolution Fourier space $\mathbb{C}$.

$\hat{\Phi}$ phase of the high resolution image in Fourier space $\mathbb{C}$.

$\hat{\imath}$ Intensity of the low resolution image in Fourier space $\mathbb{C}$.

$\hat{\phi}$ phase of the low resolution image in Fourier space $\mathbb{C}$.

$\hat{a}$ amplitude of the low resolution image in Fourier space $\mathbb{C}$.

$\hat{o}$ complex object of the low resolution image in Fourier space $\mathbb{C}$.

$\hat{p}$ pupil function in low resolution Fourier space $\mathbb{C}$.

$\imath$ Intensity of the low resolution image in real space $\mathbb{R}$.

λ wavelength.

λ_0 wavelength, of the incident light.

$\mathcal{F}^{-\infty}$ inverse Fourier transform.

$\mathcal{F}$ Fourier transform.

$\mathcal{N}$ normal distribution.

$\mathcal{O}$ order of magnitude.

μ arithmetic mean of a distribution.

A area of interest (AOI).

i imaginary unit.

ν magnitude of noise for luminosity simulations.

ν_{min} lowest possible frequency.

ω_N primitive root of unity.

ϕ phase of the high resolution image in real space $\mathbb{R}$.

ψ pitch, angle of rotations about the y-axis.

ρ roll, angle of rotations about the x-axis.

σ dimensions of a gaussian blur kernel.

σ^2 variance of a distribution.

$\theta, \theta_x, \theta_x$ illumination angle, respective projections on x-,y-axes.

ε, **eps** smallest possible number, convergence criterion.

ξ arbitrary variable.

ζ number of elements (pixels) of the high resolution object.

ζ, χ arbitrary complex objects.

a amplitude of the high resolution image in real space $\mathbb{R}$.

d minimum resolvable distance.

d_F Fraunhofer distance.

f, g, h arbitrary functions.

k_0 wave number of the incident light.

k_x, k_y, kx, ky, kx, ky two-dimensional spatial frequency coordinates.

k_{co} spatial cutoff frequency of the optical system.

l length, e.g. l_{ccd}, pixel-size of the imaging detector.

m arbitrary index.

mag magnification of an optical system.

n image stack index.

n_b size of the bright-field area of $n_b \times n_b$ LEDs.

o complex object of the high resolution image in real space $\mathbb{R}$.

r estimated radius of the pupil function in low resolution Fourier space.

res resolution of the microscope.

s pseudo-random normal distributed variable.

w diameter of the circular aperture (resembling a circular emitting area).

x, y two-dimensional spatial coordinates.

z spatial coordinate, distance of the object from the source.

NA numerical aperture.

Operators

$*$ convolution.

$\mathcal{F}$ Fourier transform.

$\mathcal{F}^{-1}$ inverse Fourier transform.

$\star$ autocorrelation, cross-correlation.

Index