a

Projektarbeit / Student Project

Dual Channel Single Molecule Localisation Microscopy

concluded at the

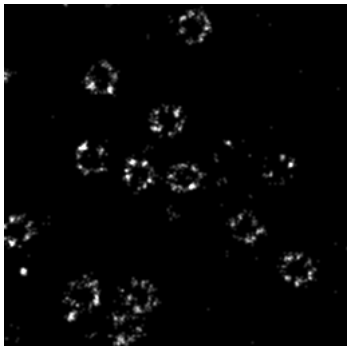
Technische Universität (Tu) Wien

advised by

Lukas Velas

of

Moritz Siegel



Vienna
June 23, 2021

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

Contents

Chapter 1

Introduction

we want two color dyed SMLM data in sync, how to get there?

Single Molecule Localisation Microscopy (SMLM) is a technique of fitting a full Point Spread Functions (PSF) to a stack of images containing reasonably spaced fluorescence signals.

To be able to accurately estimate the 3 dimensional location of the fluorescent molecule, the PSF must be known quite well. Simply put: If one knows the shape of a point source in varying degrees of defocus, one can guess the defocus and thus the z coordinate of the fluorescence molecule.

1.1 Flouro

Flouro

Chapter 2

Methods

2.1 SMLM

SMLM

2.2 TIRF

TIRF

2.3 dSTORM

dSTORM

2.3.1 Gloxy Buffer

Single channel SMLM may employ the same buffer for both excitation wavelengths, in our case at 645 nm (red) respective at 488 nm (blue). After its central ingredients glucose and oxidase, the buffer we used throughout this paper for all single channel measurements unless noted otherwise is called Gloxy buffer:

Gloxy buffer concentration

- 50 mmol β-MercaptoEthylamine hydrochloride (MEA, Sigma-Aldrich).
- 10 vol% of a 250 g L^{-1} solution of glucose.
- 0.5 mg ml^{-1} glucose oxidase.
- 40 mg ml^{-1} catalase (Sigma-Aldrich).
- in PBS, pH 7.6 .

2.3.2 OxEA Buffer

Dual channel Fluorescence microscopy poses novel challenges to find a proper dSTORM buffer, that works for both excitation wavelengths—thus two distinct fluorophores AF647 and AF488. The buffer composition we used is based on cit, and called OxEA after its main ingredients OxyFlour and (β-Mercapto)Ethylamine:

OxEA buffer concentration

- 50 mmol β-MercaptoEthylamine hydrochloride (MEA, Sigma-Aldrich).
- 3 vol% OxyFlourTM (Oxyrase Inc., Mansfield, Ohio, U.S.A.).
- 20 vol% of 60% sodium DL-lactate solution (L1375, Sigma-Aldrich).
- in PBS, pH adjusted to 8–8.5 with NaOH.

OxEA buffer protocol

For about 1 mL of OxEA buffer we used the amounts shown in Table 2.1, to obtain above listed concentrations.

pH

The pH of the OxEA buffer is checked using both broad range pH testing strips, and a digital pH meter, to be between pH 7 and pH 8.

Table 2.1: Ingredients used for preparation of OxEA buffer for dual channel dSTORM buffer.

Order	Ingredient	Store	Vol / µL
1	Ultra pure H_2O		600
2	10 M NaOH		20
3	10× PBS		100
4	60% DL-lactate	fridge	200
5	1 M MEA	freezer	50
6	OxyFluor	freezer	30

2.4 PSF

PSF

2.5 Zernike

zern

2.6 Dual Channel Transforms

transform

Chapter 3

Results

3.1 Flouro

sample image

3.1.1 dSTORM

2d npc images

3.2 Single Molecule Localisation

It is possible to use different excitations to simultaneously measure different fluorescence markers, but for that the Point Spread Functions (PSF) has to known for each wavelength. So the PSFs are estimated both for red and blue laser light, with the Phase Retrieval program by [Jesacher et al 21?]; each on a stack of 50000 STORM images of 100 nm 100nm? beads stained with fluorescence dyes for both red (645 nm) and blue (488 nm) laser light.

3.2.1 Correction Collar

To further complicate things, the new Olympus 1.5 NA objective comes with a Correction Collar to compensate for aberrations—which naturally vary slightly for both color channels. In order to find the best Correction Collar setting of the Olympus 1.5 NA objective for SMLM, the Point Point Spread Functions (PSF) is computed for each of the three settings of the correction collar $\{0.13, 0.17, 0.19\}$; using the program by [Jesacher et al 21?].

3.2.2 Zernike Modes

The Zernike modes of the PSF are shown for each of the three correction collar settings $\{0.13, 0.17, 0.19\}$, each for blue channel respective red channel in Figure 3.1 respective Figure 3.2.

For convenience the same results are additionally shown grouped by the the three correction collar settings $\{0.13, 0.17, 0.19\}$, now for both red and blue channel in Figure 3.3, Figure 3.4 and Figure 3.5.

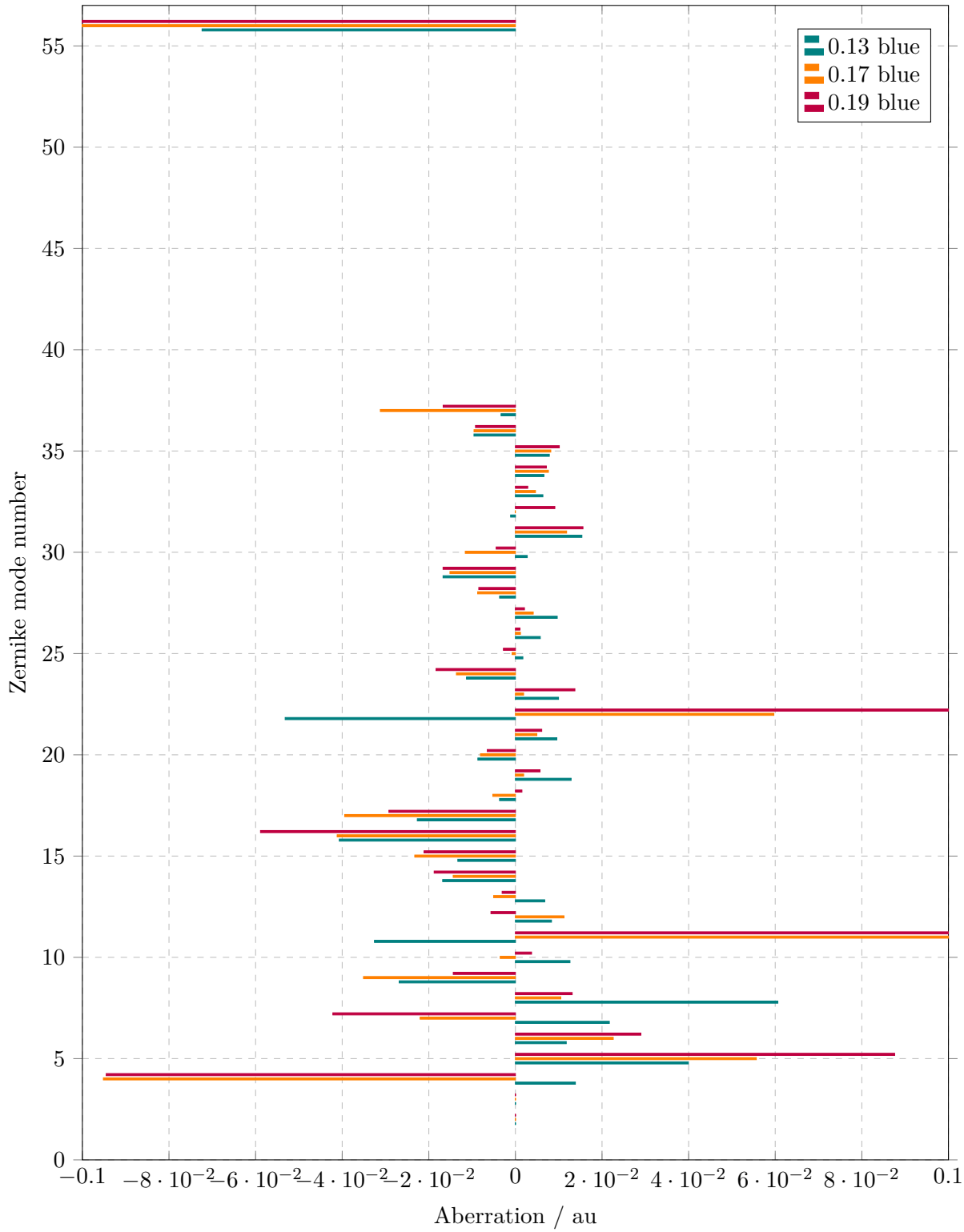


Figure 3.1: Blue channel Zernike modes $\{1 \ldots 37, 56\}$ *versus aberrations of PSF model via phase retrieval, for all three correction collar settings (0.13, 0.17, 0.19).*

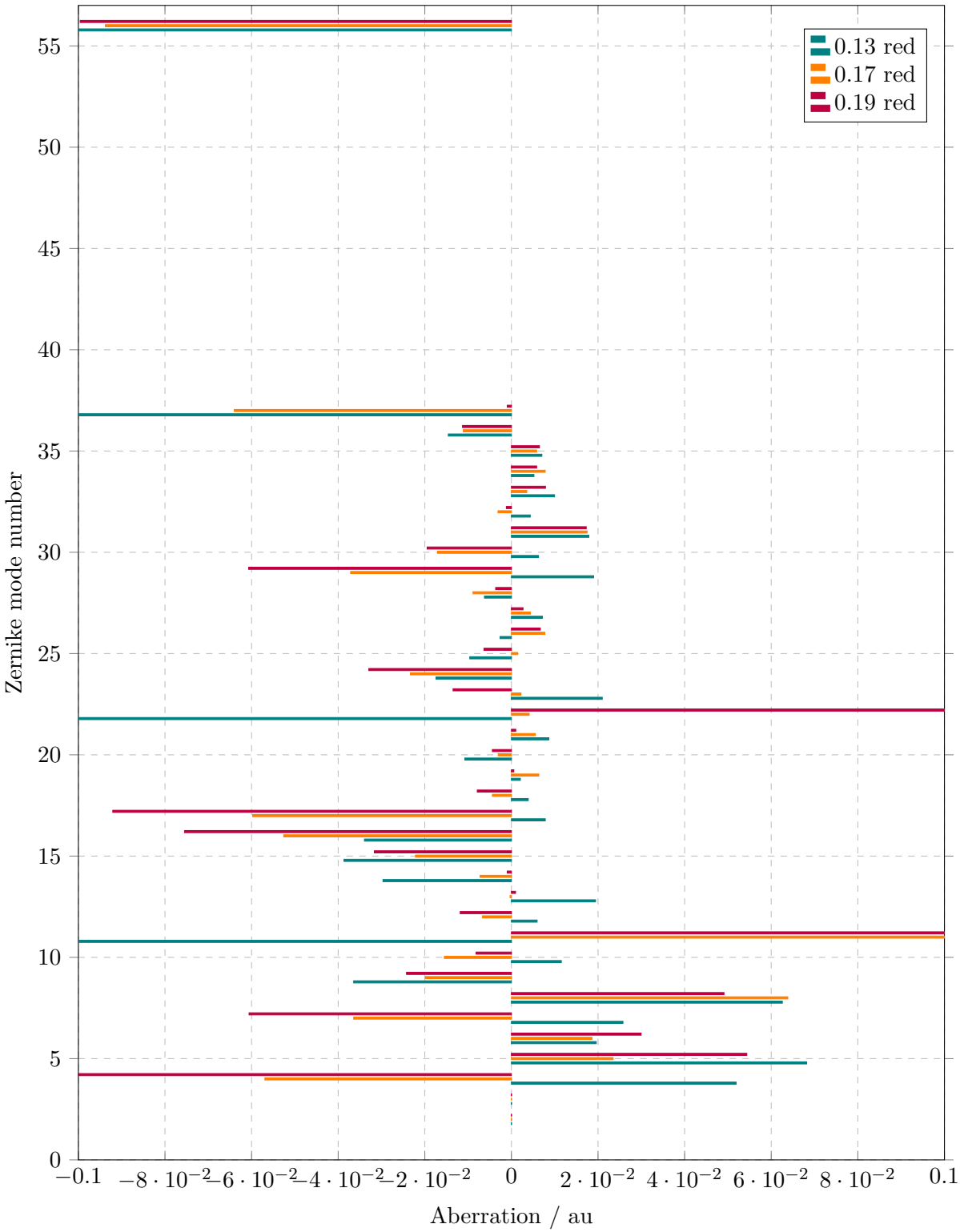


Figure 3.2: Red channel Zernike modes $\{1 \dots 37, 56\}$ *versus aberrations of PSF model via phase retrieval, for all three correction collar settings (0.13, 0.17, 0.19).*

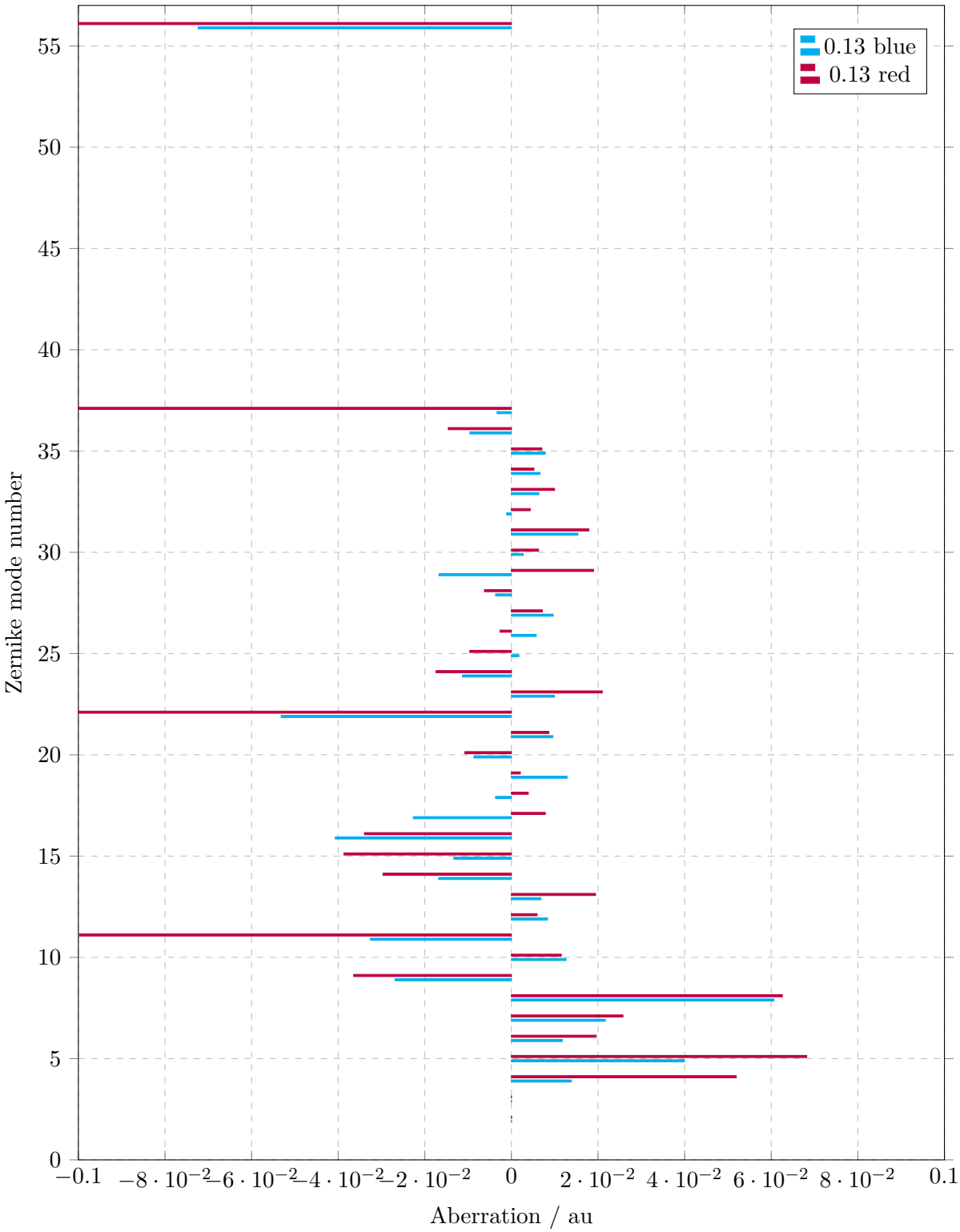


Figure 3.3: Red and blue channel Zernike modes $\{1 \ldots 37, 56\}$ *versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.13.*

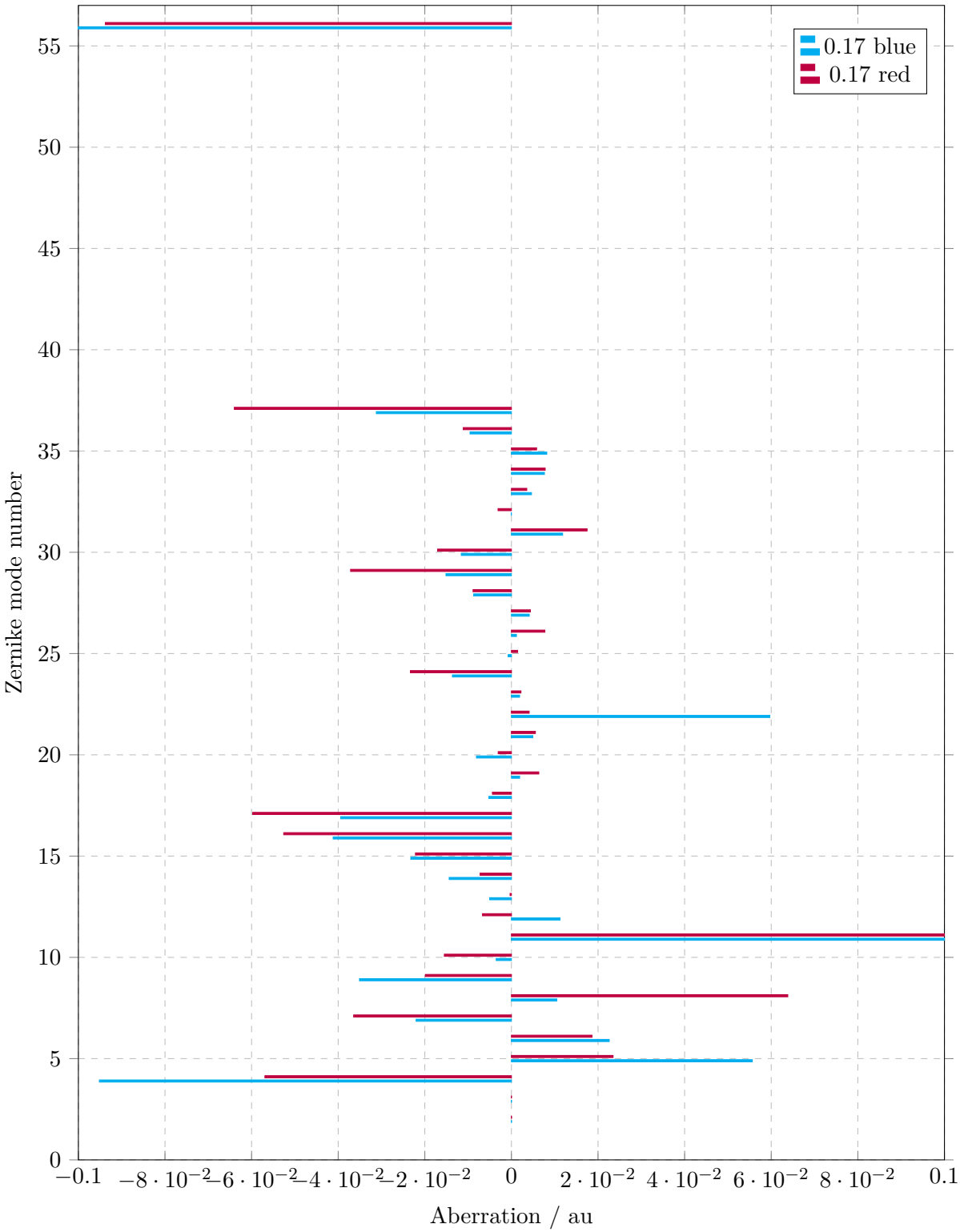


Figure 3.4: Red and blue channel Zernike modes $\{1 \ldots 37, 56\}$ *versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.17.*

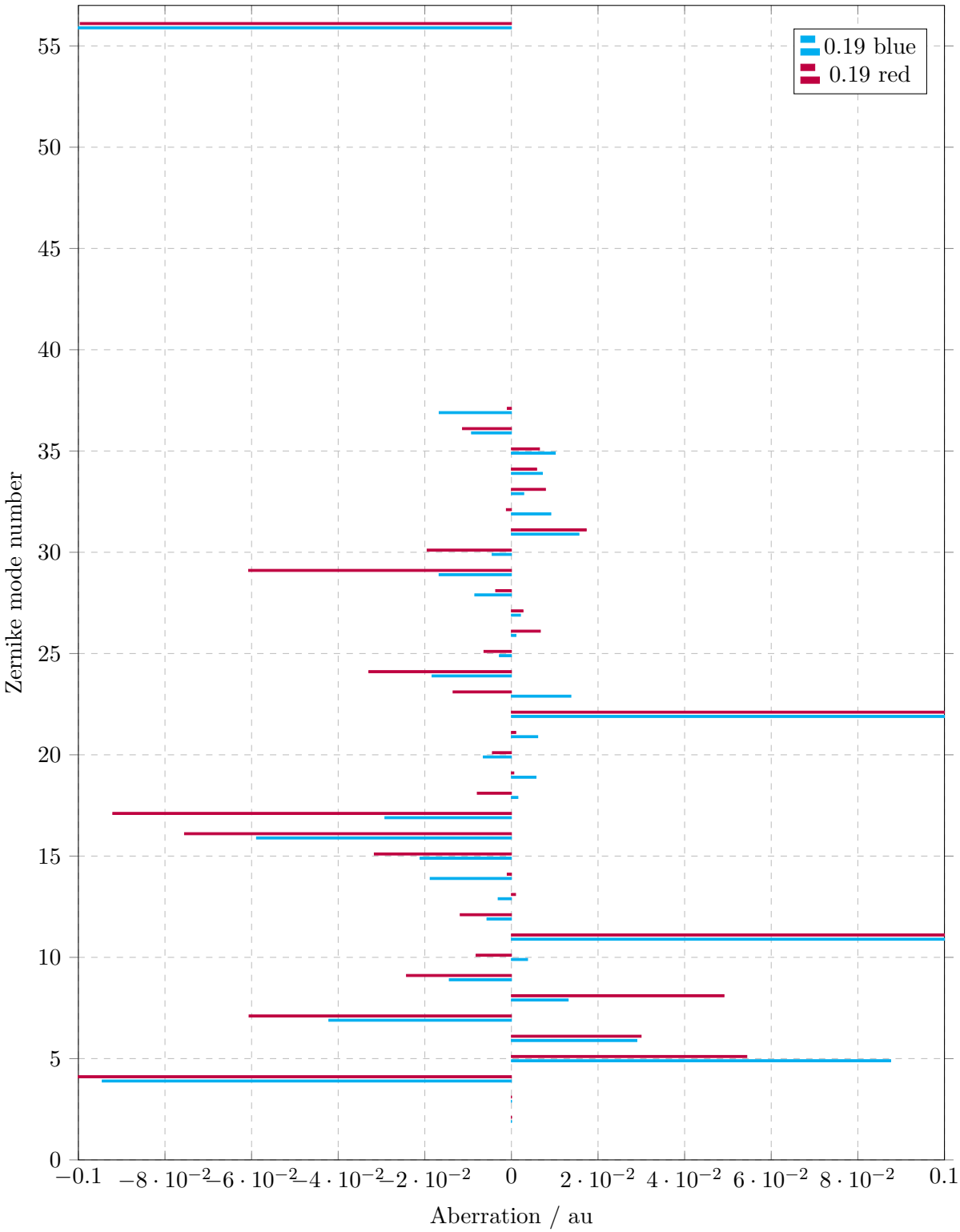


Figure 3.5: Red and blue channel Zernike modes $\{1 \ldots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.19.

3.2.3 Cramer Rao Lower Bound

In order to find the best Correction Collar setting of the Olympus 1.5 NA objective for SMLM, the Cramer Rao Lower Bound (CRLB) is computed using the program by [Jesacher et al 21?], with prior estimated Point Spread Functions (PSF) via phase retrieval [Jesacher et al 21?].

All CRLBs are computed at a defocus position of -500 nm, in steps of 5 nm from 0 to 500 nm. For all Estimations we assume identical arbitrary signal strength (2500) respective background (100).

The Estimated Cramer Rao Lower Bound for different correction Collar settings (variing linestyles for 0.13, 0.17, 0.19) of the Olympus 1.5 NA objective; for X, Y and Z axis (top, middle, and bottom); for both red and blue channel (colors) is shown in Figure 3.6.

Additionally plots grouped by X, Y and Z axis are shown in Figure 3.7, as well as grouped by different correction Collar settings in Figure 3.8.

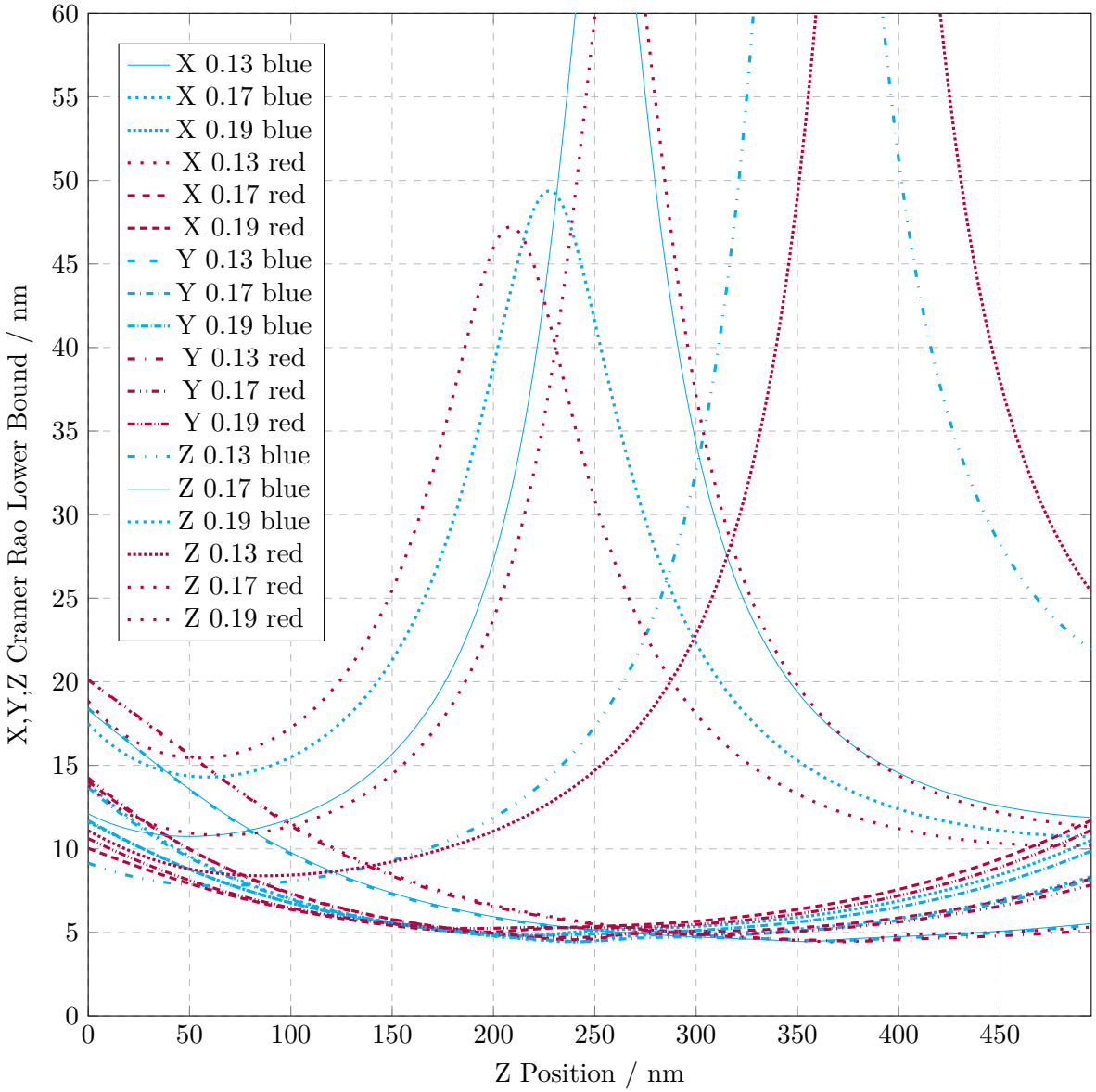


Figure 3.6: Estimated Cramer Rao Lower Bound for different correction Collar settings (variing linestyles for 0.13, 0.17, 0.19) of the Olympus 1.5 NA objective; for X, Y and Z axis (top, middle, and bottom); for both red and blue channel (colors).

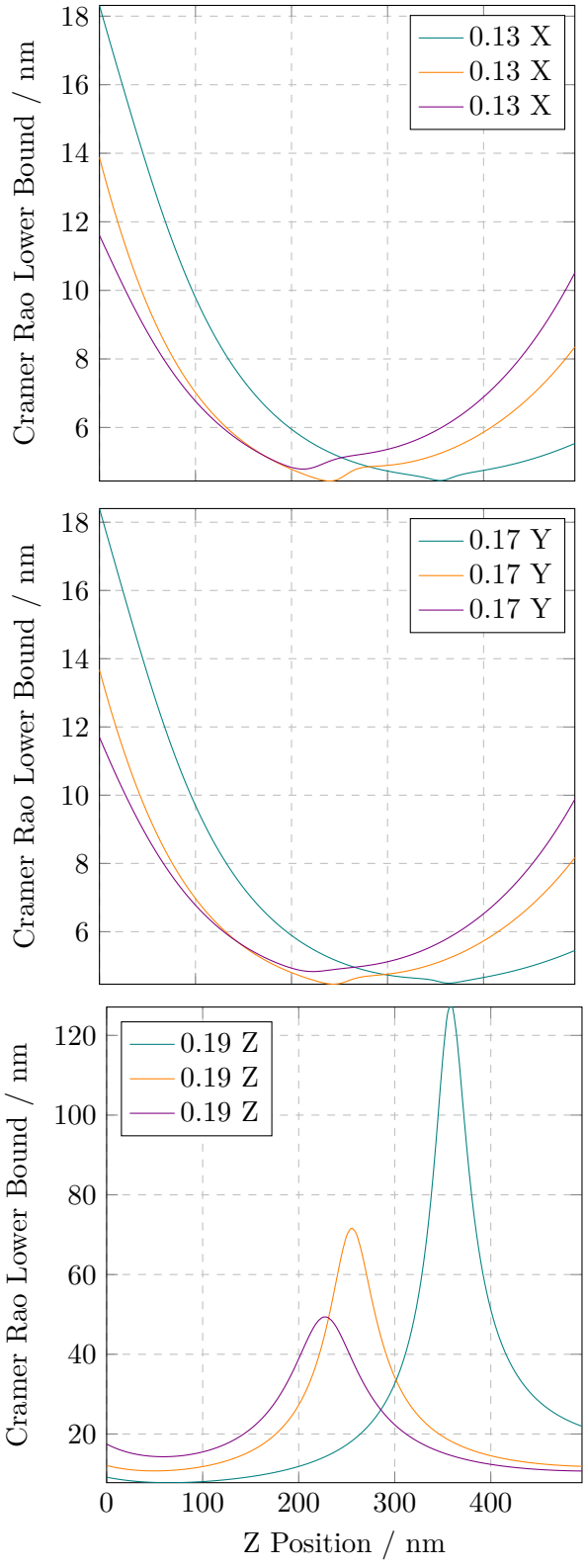


Figure 3.7: Estimated Cramer Rao Lower Bound for different correction Collar settings (teal: 0.13, orange: 0.17, purple: 0.19) of the Olympus 1.5 NA objective; grouped by X, Y and Z axis (top, middle, and bottom).

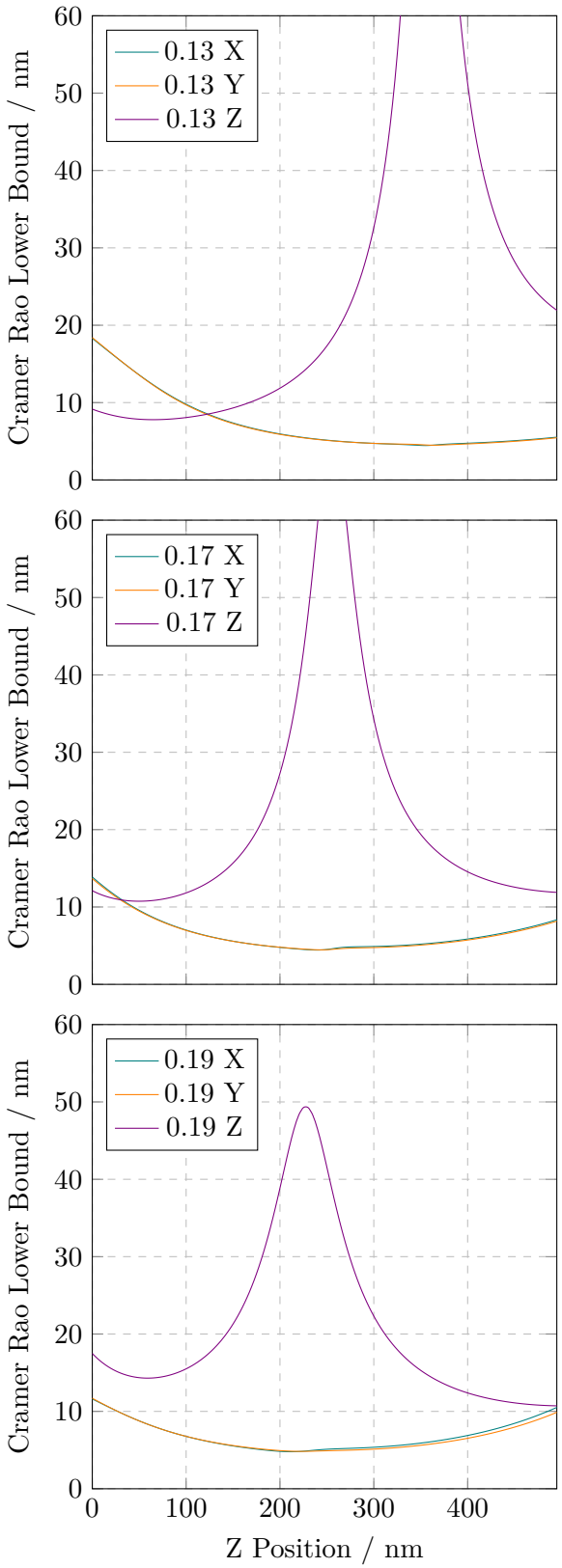


Figure 3.8: Estimated Cramer Rao Lower Bound for X, Y and Z axis (teal, orange and purple lines); grouped by different correction Collar settings (top: 0.13, middle: 0.17, bottom: 0.19) of the Olympus 1.5 NA objective.

3.3 Dual Channel: Simulation

Dual Channel: Simulation

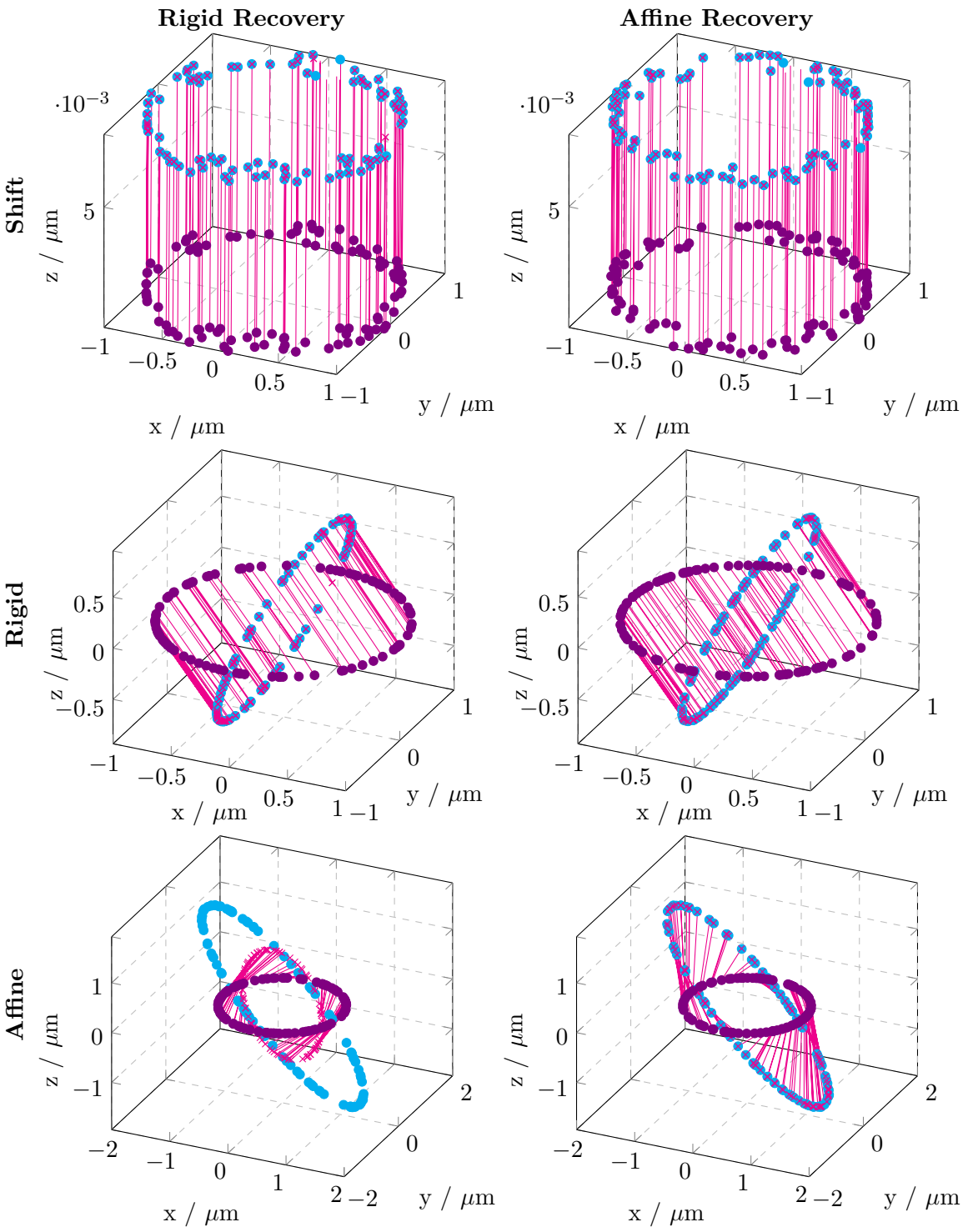


Figure 3.9: Demonstration of the recovery of rigid (left) respective affine (right) transformations, via recovering localisations of simulated two channel SMLM data; transformation (magenta) from red channel (violet) to blue channel (cyan); for the use cases of translation (top), rigid transformation (middle) and affine transformation (bottom). Obviously a rigid transformation may not correctly reconstruct an affine transformed data set (bottom left).

3.4 Dual Channel: NPC

Dual Channel: NPC

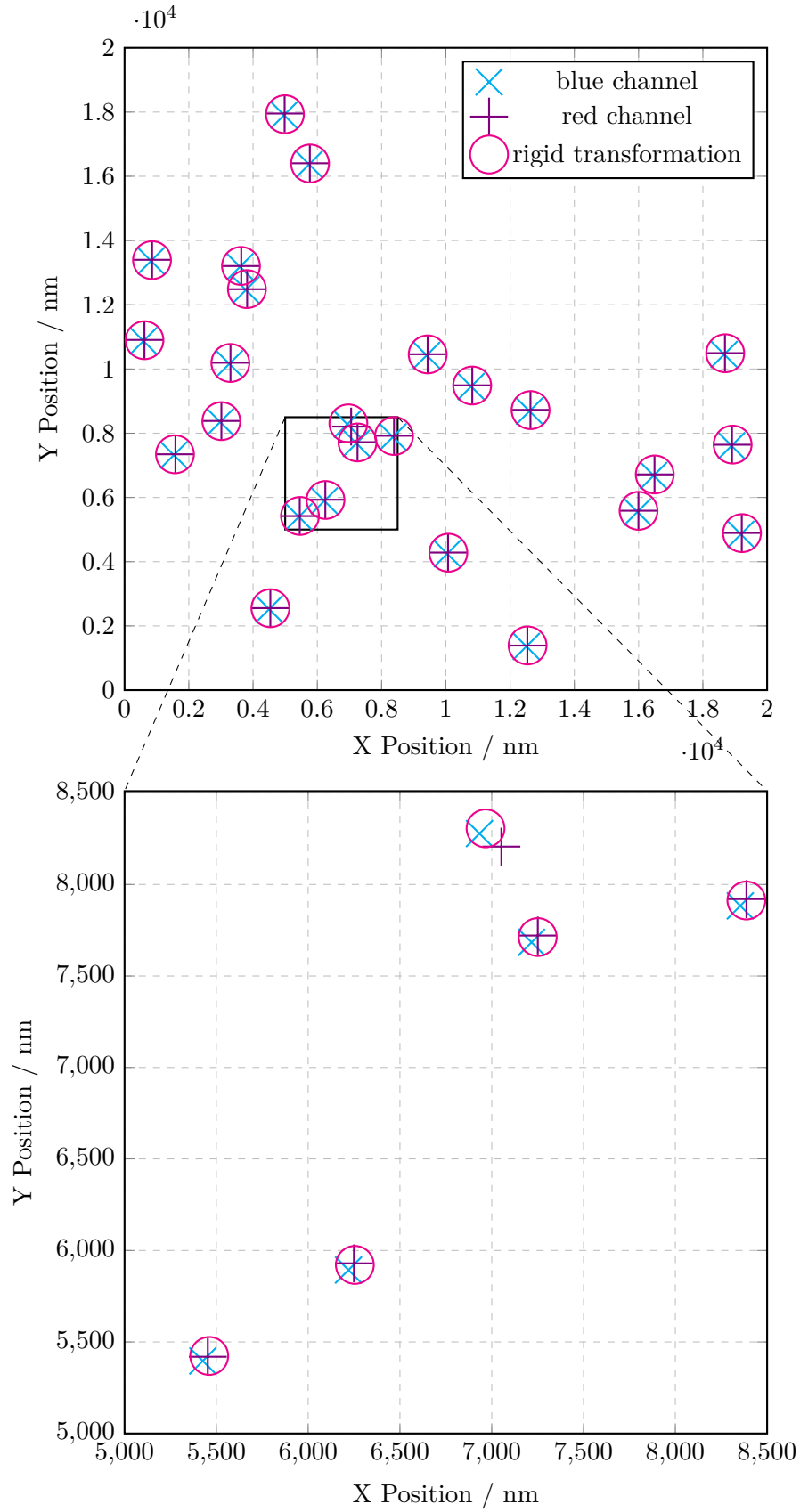


Figure 3.10: Demonstration of a rigid transformation of the localisations (magenta ∘) from blue channel (blue ×) to red channel (violet +); the transformed blue channel localisations mostly align well with the red channel localisations.

3.5 SMLM Analysis

Applying the SMLM described in Section sec for the stack of microscopy images, one obtains a list of localisations comprised of position (`x,y,z`), photon count (`n`), and a fit parameter (`fit`). Where the `sans serif` typeset letters refer to the variables in the code listed in this chapter.

In this section I want to describe our data analysis pipeline, in order to cut down the massive amount of initial localisations—in our example case over 130k—to distill it to the most meaningful conclusions.

As a proof of work we used NPCs as sort of a well defined biological test target. As these are used frequently for the purpose of validating a new method, it would be nice to quickly evaluate of *how good are the NPCs resolved*. The Section sec is thus dedicated to find some metric for *best resolved* NPCs.

This analysis is performed in *Python*, and is freely available in my Git repository git; both as a plain python file (.py) as well as in the format of a *Jupyter Notebook* (.ipynb).

For the sake of readability we omit the code sections generating the shown plots, as they are mostly redundant. Of course the full code is available online.

npc image

appendix

3.5.1 Import

Here we import the used packages and the data file we obtain from running the SMLM algorithm.

3.5.2 Drift Correction

The *drift* of the setup over time can be estimated using *ImageJ*, and is then imported as `drift` drift. In the following block the drift correction is applied to all the 130k localisations, shown in Figure 3.11 as 3d plot of all the initial 370k drift corrected localisations.

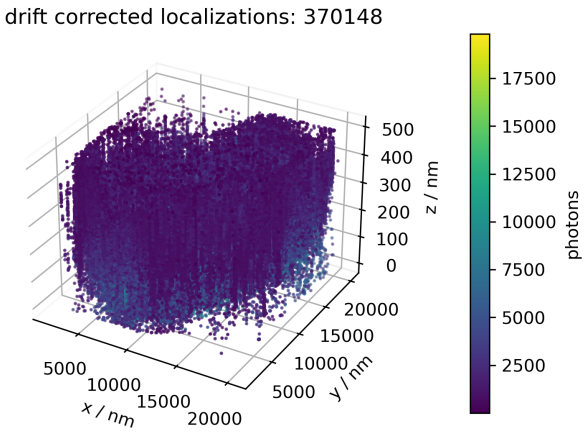


Figure 3.11: 3d plot of all 370148 drift corrected localisations, color coded by photon count.

3.5.3 Photon Counts

As a preliminary step it might prove useful to look at the photon count statistics, shown in Figure 3.12. Here we can easily see what the supposed intensity of a single molecule is; the peak, in our case about 2000. Those localisations below may be considered noise, those far above are probably overlapping or stacked molecules, so their intensities add up.

3.5.4 Filter

In this first filter we limit the dataset to the more meaningful points; like those with intensities between 2k and 7k. Also since we defocussed for 500 m, only those `z` values between 0 and 499 can be considered realistic. Here 0 means directly attached to the glass substrate, so negative values would be *inside* the glass substrate; and thus need to be discarded as unplausible. The dataset could be filtered by

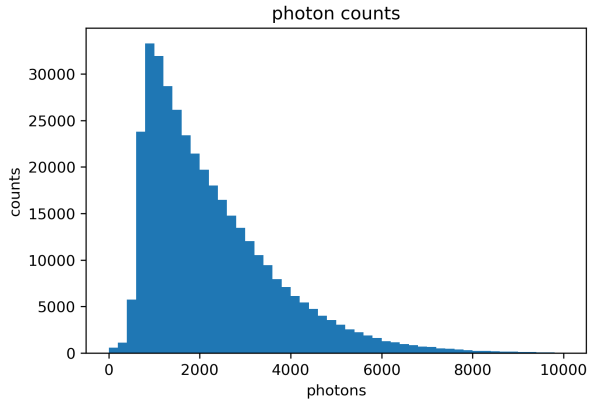


Figure 3.12: histogram of the photon count for the localisations.

`min_fit`, but to our findings this does not contribute much.

Figure 3.13 shows a 3d plot of the remaining 154k localisations after we apply the filter, thus effectively shrinking down our exemplary data set by about 40%.

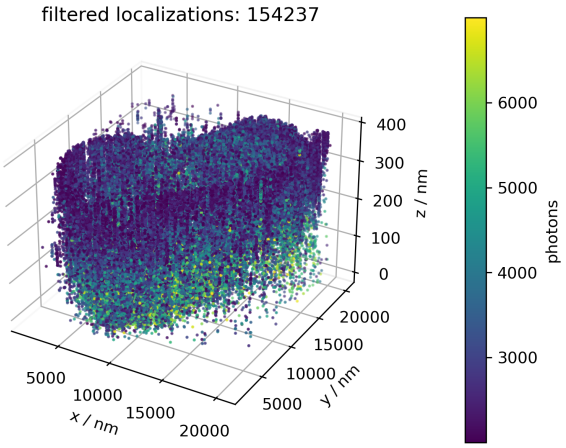


Figure 3.13: 3d plot of all 154237 filtered localisations, color coded by photon count.

3.5.5 Track Particles

The 50k frames we analyse here are taken with exposure of 20 m s, and 10 m s between consecutive exposures. Depending on the laser intensity and the buffer composition the bright state has a specific half-life. This leads to one exemplary molecule being *on* for some 30 m s (one frame) while some other is on for 60 m s; and so appears in two consecutive frames. Since the molecule in those two frames essentially is the same, we can *track* it over time: effectively averaging the location if present in multiple frames, thus reducing the amount of localisations while at the same time increasing their precision.

The parameters `sr` denotes the *search range*, how far apart two consecutive localisations are still considered *one* particle, this has to be adjusted based on the physical setup considering vibrations and the like.

Figure 3.14 shows a 3d plot of the remaining 98k localisations, after we track the particles. So the tracking step further shrinks down our exemplary data set by about 60%.

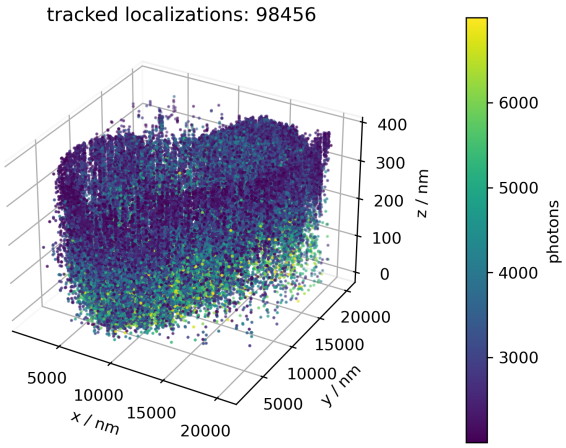


Figure 3.14: 3d plot of all 98456 filtered localisations, color coded by photon count.

3.6 SMLM Analysis: NPC

We use NPCs as a test target for our method, thus we have some additional knowledge, like

the geometry of each individual NPC: they comprise two stacked tori, each about 150 m diameter (in x,y direction), and 150 m apart (in z direction).

Now we can group our dataset with close to 100 thousand localisations to clusters of roughly this size in x,y (set `dim=2`). Note that for the sake of completeness, we include the possibility of clustering in 3d to spheres in x,y,z (set `dim=3`), but mind that x,y and z precision most often greatly varies (see Section sec).

The parameter `min_samples` denotes the minimum amount of constituents a cluster must have to be considered such.

Figure 3.14 shows a 3d plot of the localisations of 658 identified clusters, omitting all the other 35489 localisations as noise. So the clustering step further shrinks down our exemplary data set by about 30%.

3.6.1 Clustering

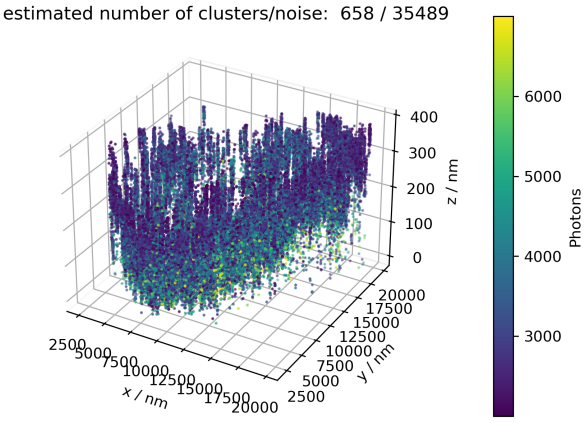


Figure 3.15: 3d plot of the localisations of the 658 identified clusters (omitting 35489 localisations as noise, due to not belonging to a cluster), color coded by photon count.

3.6.2 Cluster Analysis

For all the localisations within each of these identified clusters, we now derive the centroid position (`xmean,ymean,zmean`), with standard derivation (`xvar,yvar,zvar`). This enables the classification of the within-cluster distribution of localisations, alas how well they represent the anticipated NPC geometry.

To obtain this *quality*, we compose both the quantity `ringness`, denoting how well the cluster shapes a ring in x,y direction; as well as the quantity `twofold`, denoting how well the cluster shapes two stacked tori in z direction, basically forming a camel-curve in z direction. To break this down to one scalar each, we compute the root mean square (RMS) of the deviations of each localisations radius from the cluster centre (in x,y direction) from the known NPC radius (Lines 20–24). The quantity `twofold` is similarly comprised of a (RMS) deviation of each localisations z value from the cluster centre (in z direction) from the known NPC height. The mentioned parameters are thus called `npc_radius`, respective `npc_height`.

Figure 3.16 shows a broad overview of the location of the cluster centres (not the localisations within), color coded by photon count. This should be considered more of a short sanity check than a profound analysis.

3.6.3 Select Clusters

Now we possess all the information needed to evaluate the list of clusters; for example to sort for the lets say 10 most *ringlike* clusters. Or, by setting `sort = 'ringness + twofold'`, we may find the 10 *best* clusters in terms of both `ringness` and `twofold`—weighted equally—which would comprise the 10 *overall best*, thus *most NPC like* clusters.

For sake of completeness we also include the x,y and z variances, even if they do not comprise a very meaningful parameter in the par-

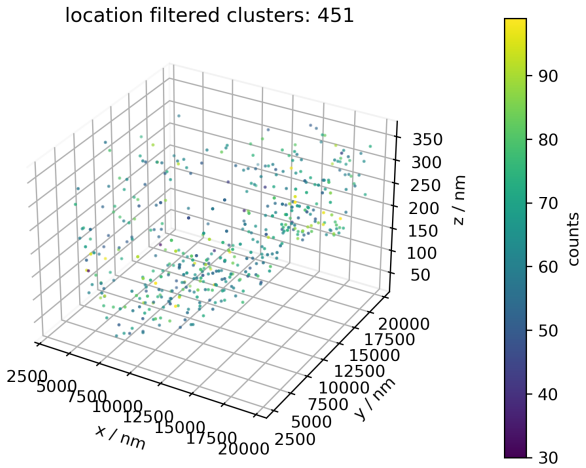


Figure 3.16: 3d plot of the positions of the 451 filtered clusters, color coded by photon count.

ticular case due to the NPCs geometry.

Figure shows a 3d plot of the localisations within the 10 *best* clusters, color coded by photon count.

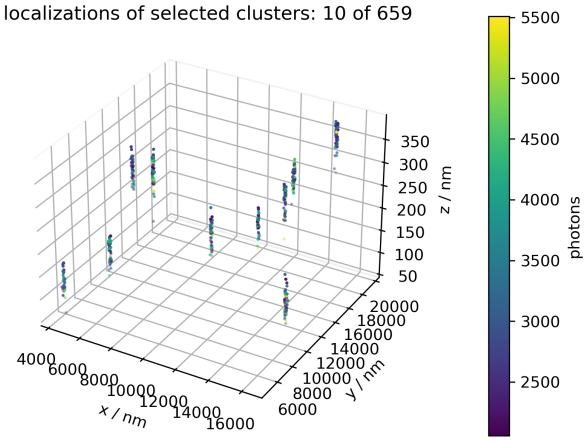


Figure 3.17: 3d plot of the localisations within the 10 best *clusters, color coded by photon count.*

3.6.4 X,Y,Z Histograms

As a further sanity check, we plot the histograms for selected clusters (`plot_cluster`), in x,y and z direction; to figure out if the sorting did work effectively—thus the higher sorted clusters are indeed *better* examples of NPCs.

Figure 3.18 shows exemplary histograms of the x,y (right) respective z distribution (left) of the localisations within the *best* cluster (top), the *worst* (bottom) and one in between.

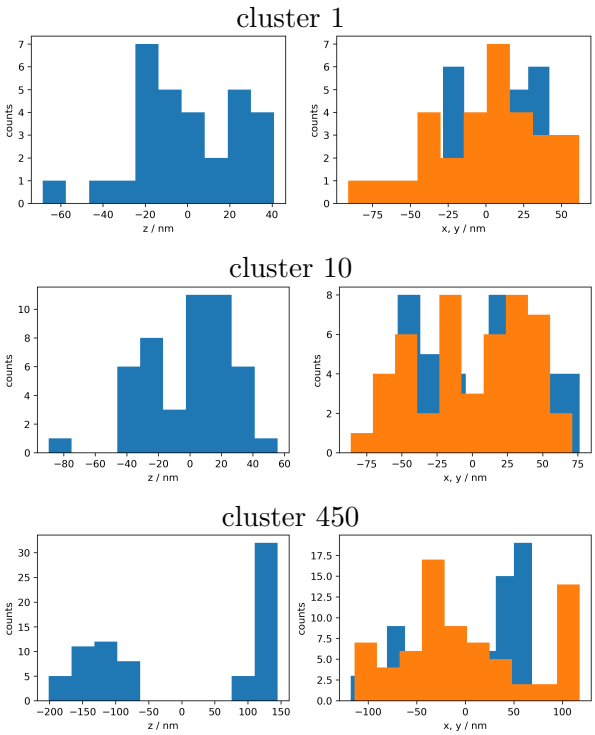


Figure 3.18: histogram of the x,y (right) respective z distribution (left) of the localisations within the best *cluster (top), the* worst *(bottom) and one in between.*

Chapter 4

Discussion

4.1 SMLM

4.1.1 Phase Retrieval

The image stacks look ok, even if not all are perfectly symmetrical–some errors are to be expected.

The estimated Zernike modes of the PSF are mostly plausible; based on the modes and the order of magnitude.

This is further backed by the comparison of the results grouped by the three correction collar settings $\{0.13, 0.17, 0.19\}$, in Figure 3.3, Figure 3.4 and Figure 3.5. It is quite obvious, that the results for both red and blue channel are in the same order of magnitude—if not quite similar—for most of the Zernike modes, as suspected by theory zern.

Yet the phase Retrieval program gave the error *high residual error* for both the red and the blue channel when using the correction collar settings $\{0.17, 0.19\}$, is this a serious problem?

how can we avoid it?

4.1.2 Correction Collar

Based on the Figures 3.7 and Figure 3.8, and considering the fact—that we are interested in moderately defocusing (up to say 250 nm)—one might conclude, that the most preferable setting for the correction collar is 0.13.

Yet the recommended setting for our microscope setup is 0.17! Which is backed by Lukas, based on the PSF stacks: One would have guessed the 0.17 is more sensitive to z since it changes much more with different z positions than 0.13.

Which is the one we should use?

4.2 SMLM Analysis

The analysis performed in Section 3.5 is well suited to greatly reduce the localisations, just by omitting unplausible data and noise; in our example from initially 370k to below 100k, or to about 26%.

4.3 SMLM Analysis: NPC

4.3.1 Automation

The fully automatic evaluation of the NPC *quality*, shown in Section 3.6 surfaced mixed results. The key problems being for one, that

there are quite many parameters involved; and secondly that the analysis proved very susceptible to changes of most of these parameters.

A quick glance at the histograms in Section 3.6.4 shows, that the cluster considered to be the *best* does not look better in fact, than most of the other clusters. We may well consider this approach failed.

Nevertheless we might have gained some valuable insight, into why this approach doesn't work. Reasons for this may be any and all of the following, for some of which we may suggest possible improvements.

numbers The clusters consist of very few localisations each, statistical analysis of so few elements have to be considered shaky. More constituents within each cluster would probably make this analysis more reliable.

quality The definitions of the two quality entities *ringness* and *twofold*, might not be derived well using RMS. A more sophisticated approach to quantify the clusters deviations to our known NPC geometry might work better, such as fitting the histogram in z direction to a Gaussian.

weighting The equal weighting of *ringness* and *twofold* quantities are canceling each other; Some clusters might look very *ring-like*, but are not at all stacked on top of each other, and vica versa. Unequal weighting based on empirical fine tuning might lesser this problem, but will most likely not solve it.

4.3.2 Secondary Filter

Some of the results of the NPC analysis might still prove themselves useful as a secondary filters. In reducing the clusters until only a few remain, effectively also finds the *best*.

high variance One might want to exclude clusters with an overly high variance, as these might be in fact two NPCs too close to each other to be accounted separately by the clustering algorithm.

low variance Quite similarly we might exclude those clusters showing extremely low variance, under suspicion of being well concentrated—yet noise, not representing a NPC at all.

spread Likewise we may exclude clusters spreading far in z, due to our knowledge of the NPC heigth, as well as in x,z due to the NPCs well defined radius.

In the end, possessing a list of the clusters position allows for a much easier way to plot some of them manually, than to zoom in on a plot of thousands of localisations repeatedly.

Chapter 5

Conclusion

5.1 Phase Retrieval

The Zernike modes for both red and blue channel are estimated via phase retrieval of in-focus measurements of beads at various depths. This yields a full PSF model to be used for de-focus 3d SMLM.

5.2 Correction Collar

The best setting of the Correction Collar for the Olympus 1.5 NA objective is shown to be 0.13. Here we find the preferable compromise between x,y precision and z precision, in the regime between about 0 and 250 µm.

5.3 dSTORTM Buffer

.1 SMLM Analysis: NPC

.1.1 Import

```
#@markdown ##import core
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import scipy.io
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler

#@markdown ##import jupyter
#%matplotlib ipympl
#%matplotlib widget
#%matplotlib interactive
% matplotlib inline
import trackpy
#import sdt
#from sdt import io, chromatic, multicolor, brightness

## local
wd = 'data/210422_npc_red_defocus/'
data = pd.read_csv( wd + 'cell1_tr1000_def500.csv',
header = None,
names=["x", "y", "z", "n", "bg","fit","id","frame"] )
```

.1.2 Drift Correction

```
#@markdown ## import & scale drift
#@markdown > set **magnification** via factor in drift.

drift = pd.read_csv( wd + 'day2_cell1_driftValues.csv')

drift['Y2']=drift['Y2']*146.6
drift['Y3']=drift['Y3']*146.6
drift['X2']=np.round(drift['X2'])
drift['X3']=np.round(drift['X3'])

#@markdown ## apply drift correction

for i in range(len(drift)-1):
```

```
fr=data[(data['frame']>=drift['X2'].iloc[i]) &
(data['frame']<drift['X2'].iloc[i+1])]
fr['y']=fr['y']-drift['Y2'].iloc[i]
fr['x']=fr['x']-drift['Y3'].iloc[i]
data[(data['frame']>=drift['X2'].iloc[i]) &
(data['frame']<drift['X2'].iloc[i+1])]=fr
```

.1.3 Photon Counts

```
#@markdown ## Check: photon counts
#@markdown > set `max_photons` accordingly (default: 10000).
max_photons = 10000 #@param {type:"slider", min:0, max:40000, step:1000}

fig = plt.figure()
plt.hist( data['n'], bins=50, range=( 0, max_photons ) )
```

.1.4 Filter

```
#@markdown ## filter
#@markdown > set `min_photons` and `max_photons` accordingly (default: 2000 < photons < 7
#@markdown > set `min_z` and `max_z` accordingly (default: 0 < z < 499).\
#@markdown > set `min_fit` accordingly (default: 6e6).
min_photons = 2000 #@param {type:"slider", min:0, max:40000, step:1000}
max_photons = 7000 #@param {type:"slider", min:0, max:40000, step:1000}
min_z = 0 #@param {type:"slider", min:0, max:500, step:1}
max_z = 384 #@param {type:"slider", min:0, max:500, step:1}
min_fit = 7192000 #@param {type:"slider", min:0, max:1e7, step:1000}

fdata = data[ ( data['n'] > min_photons ) &
( data['n'] < max_photons ) &
( data['z'] > min_z ) &
( data['z'] < max_z ) &
( data['fit'] < min_fit ) ]
```

.1.5 Track Particles

```
#@markdown ## track all in x,y
#@markdown > set `sr` to wanted search range (default: 50).\
#@markdown > set `mem` to wanted memory (default: 10).
sr = 50 #@param {type:"slider", min:0, max:100, step:1}
mem = 10 #@param {type:"slider", min:0, max:100, step:1}

linkedxy = trackpy.link_df( fdata,
```

```
    pos_columns = ["x","y","z"],
    search_range = sr,
    memory = mem )

    particles = linkedxy.groupby( "particle" ).aggregate( np.mean )
    std_pos = linkedxy.groupby( "particle" ).aggregate( 'std' )
    particles["length"] = linkedxy.groupby( "particle" ).apply( len )
    particles["z_std"] = std_pos['z'].copy()
    particles["x_std"] = std_pos['x'].copy()
    particles["y_std"] = std_pos['y'].copy()
```

.2 SMLM Analysis: NPC

.2.1 Clustering

```
    #@markdown ## compute dbscan
    #@markdown > set `dim = 2` for clustering in x and y (default:).\
    #@markdown > set `dim = 3` for experimental clustering in 3d; be aware that x,z and y pre
    #@markdown > set `eps` (default: 200).\
    #@markdown > set `min_samples` (default: 10).
    dim = "2" #@param [2, 3]
    eps = 100 #@param {type:"slider", min:0, max:500, step:10}
    min_samples = 50 #@param {type:"slider", min:1, max:100, step:1}

    alocalisations = localisations.to_numpy()
    alocalisations[ :, 0:2 ]

    db = DBSCAN( eps, min_samples ).fit( alocalisations[ :, 0:int( dim ) ] )
    core_samples_mask = np.zeros_like( db.labels_, dtype=bool )
    core_samples_mask[ db.core_sample_indices_ ] = True
    labels = db.labels_
    localisations[ "cluster" ] = labels

    # count clusters (ignore noise if present)
    n_clusters_ = len( set( labels ) ) - ( 1 if -1 in labels else 0 )
    n_noise_ = list( labels ).count( -1 )

    #print('Estimated number of clusters: %d' % n_clusters_)
    #print('Estimated number of noise points: %d' % n_noise_)

    nlocalisations = localisations.loc[ localisations['cluster'] == -1 ]
    clocalisations = localisations.loc[ localisations['cluster'] != -1 ]
```

.2.2 Cluster Analysis

```
#@markdown ## analyse clusters
#@markdown > set `npc_radius` to NPC radius /nm (default: 50).\
#@markdown > set `npc_radius` to NPC haight /nm (default: 150).
npc_radius = 50 #@param {type:"slider", min:0, max:500, step:1}
npc_height = 25 #@param {type:"slider", min:0, max:500, step:1}

clabels = set(labels)
cnames = [ "counts", "xmean", "ymean", "zmean", "nmean", "xvar", "yvar", "zvar",
"nvar", "label", "ringness", "twofold" ]
clusters = pd.DataFrame( index = clabels, columns = cnames, dtype="float64" )
clusters[ "label" ] = clabels

for k in clabels:
tmp = localisations.loc[ localisations['cluster'] == k ]
clusters.loc[ k, "counts" ] = len( tmp )
for label in [ "x", "y", "z", "n" ]:
clusters.loc[ k, label+"mean"  ] = np.mean( tmp.loc[ :, label ] )
clusters.loc[ k, label+"var"  ] = np.var( tmp.loc[ :, label ] )

## xy: radius (distance to centroid)
rad = np.sqrt( ( tmp.loc[ :, "x" ] - clusters.loc[ k, "xmean" ] )**2 +
( tmp.loc[ :, "y" ] - clusters.loc[ k, "ymean" ] )**2 )
## xy: radius rms deviation from NPC radius
clusters.loc[ k, "ringness"  ] = np.sqrt( sum( ( rad - npc_radius )**2 ) )

## z: radius (distance to centroid)
rad = abs( tmp.loc[ :, "z" ] - clusters.loc[ k, "zmean" ] )
## z: radius rms deviation from NPC radius
clusters.loc[ k, "twofold"  ] = np.sqrt( sum( ( rad - npc_height )**2 ) )

#@markdown ## filter clusters
#@markdown > set `count_threshold` to min elements (counts) in cluster (default: 30)\
#@markdown > set `diameter_threshold` to max x,y (radius) deviation of cluster from NPC d
#@markdown > set `twofold_threshold` to max z (radius) deviation of cluster from NPC heig
#@markdown > set `xyvar_threshold` to wanted x,y variance (default: 1e4)\
#@markdown > set `zvar_threshold` to wanted z variance (default: 1e4)
count_threshold = 100 #@param {type:"slider", min:0, max:200, step:1}
diameter_threshold = 500 #@param {type:"slider", min:0, max:500, step:10}
twofold_threshold = 1000 #@param {type:"slider", min:0, max:1000, step:10}
xyvar_threshold = 100000 #@param {type:"slider", min:0, max:1e5, step:1e3}
zvar_threshold = 100000 #@param {type:"slider", min:0, max:1e5, step:1e3}

```

```
fclusters = clusters[ ( clusters['counts'] < count_threshold ) &
( clusters['ringness'] < diameter_threshold ) &
( clusters['twofold'] < twofold_threshold ) &
( clusters['xvar'] < xyvar_threshold ) &
( clusters['yvar'] < xyvar_threshold ) &
( clusters['zvar'] < zvar_threshold ) ]
```

.2.3 Select Clusters

```
#@markdown ## select best clusters & plot localisations
#@markdown > set `show_clusters` to wanted number of best clusters (default: 100).\
#@markdown > set `sort` to sort the best clusters (default: ringness + twofold).
show_clusters = 10 #@param {type:"slider", min:0, max:1000, step:1}
sort = 'ringness + twofold' #@param [ "ringness + twofold", "twofold", "ringness", "xyvar

if sort == "xvar": # sort by variance
sclusters = fclusters.sort_values( "xvar" )
elif sort == "xyvar": # sort by x and y variance using least squares
sclusters = fclusters.loc[
( fclusters.xvar ** 2 + fclusters.yvar ** 2 ).sort_values().index ]
elif sort == "ringness": # sort by ringness (deviation to ringness)
sclusters = fclusters.sort_values( "ringness" )
elif sort == "twofold": # sort by ringness (deviation to ringness)
sclusters = fclusters.sort_values( "twofold" )
elif sort == "ringness + twofold": # sort by ringness (deviation to ringness)
sclusters = fclusters.loc[
( fclusters.twofold ** 2 + fclusters.ringness ** 2 ).sort_values().index ]

show_clusters = min( show_clusters, len( sclusters ) )
selected_clusters = sclusters["label"].iloc[ 0:show_clusters ]

fig = plt.figure()
ax = fig.add_subplot( projection = '3d' )
for clus in selected_clusters:
flocalisations = localisations[ ( localisations['cluster'] == clus ) ]
ff = ax.scatter( flocalisations['x'],
flocalisations['y'],
flocalisations['z'],
s=1 ,c = flocalisations['n'] )
```

.2.4 X,Y,Z Histograms

```
#@markdown ## Check: z distribution
#@markdown > set `plot_cluster` to wanted cluster (default: 0).
plot_cluster = 3  #@param {type:"slider", min:0, max:100, step:1}
plot_cluster = min( plot_cluster, len( sclusters ) -1 )

tmp = localisations.loc[ localisations['cluster'] ==
sclusters.loc[ sclusters.index[ plot_cluster ],
"label" ] ]

z = tmp.z - np.mean( tmp.z )

fig, axes = plt.subplots(2, 1, figsize=(4, 7) ) # figsize=(4, 12)

axes[0].hist( tmp.z - np.mean( tmp.z ) )
axes[0].set_xlabel('z / nm')
axes[0].set_ylabel('counts')
#axes[0].set_title( 'z' )

axes[1].hist( tmp.x - np.mean( tmp.x ) )
axes[1].hist( tmp.y - np.mean( tmp.y ) )
```

List of Figures