

a

PROJEKTARBEIT / STUDENT PROJECT

DUAL CHANNEL SINGLE MOLECULE LOCALISATION MICROSCOPY

concluded at the

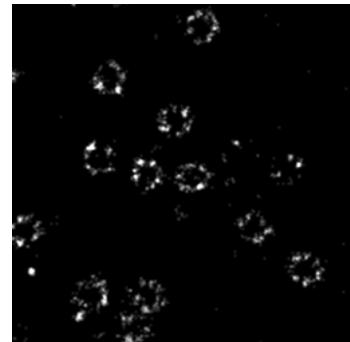
TECHNISCHE UNIVERSITÄT (TU) WIEN

advised by

LUKAS VELAS

of

MORITZ SIEGEL



Vienna
June 23, 2021

moritz.siegel@tuwien.ac.at
<https://github.com/imrahiliyas/biophysics>

Contents

1	Introduction	3
1.1	Fluoro	3
2	Methods	4
2.1	SMLM	4
2.2	TIRF	4
2.3	dSTORM	4
2.3.1	Gloxy Buffer	4
2.3.2	OxEA Buffer	4
2.4	PSF	5
2.5	Zernike	5
2.6	Dual Channel Transforms	5
3	Results	6
3.1	Fluoro	6
3.1.1	dSTORM	6
3.2	Single Molecule Localisation	6
3.2.1	Correction Collar	6
3.2.2	Zernike Modes	6
3.2.3	Cramer Rao Lower Bound	12
3.3	Dual Channel Transformation	15
3.4	Dual Channel: NPC	17
3.5	SMLM Analysis	17
3.5.1	Import	19
3.5.2	Drift Correction	19
3.5.3	Check Photon Counts	20
3.5.4	Filter	21
3.5.5	Track Particles	22
3.6	SMLM Analysis: NPC	23
3.6.1	Clustering	23
3.6.2	Cluster Analysis	24
3.6.3	Select Clusters	26
3.6.4	X,Y,Z Histograms	27

4 Discussion	30
4.1 SMLM	30
4.1.1 Phase Retrieval	30
4.1.2 Correction Collar	30
5 Conclusion	31
5.1 Phase Retrieval	31
5.2 Correction Collar	31
5.3 dSTORTM Buffer	31
List of Figures	32

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.

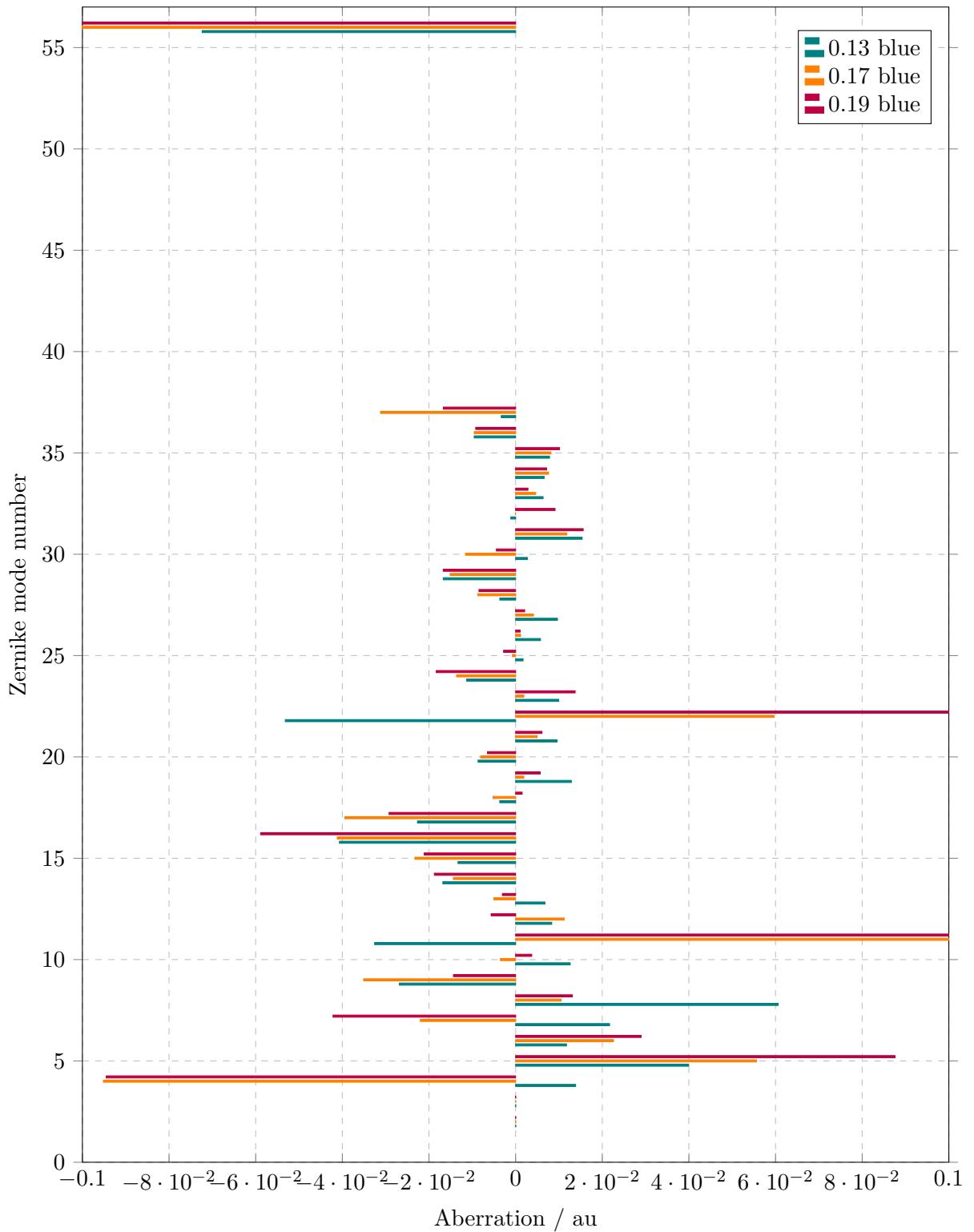


Figure 3.1: Blue channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for all three correction collar settings (0.13, 0.17, 0.19).

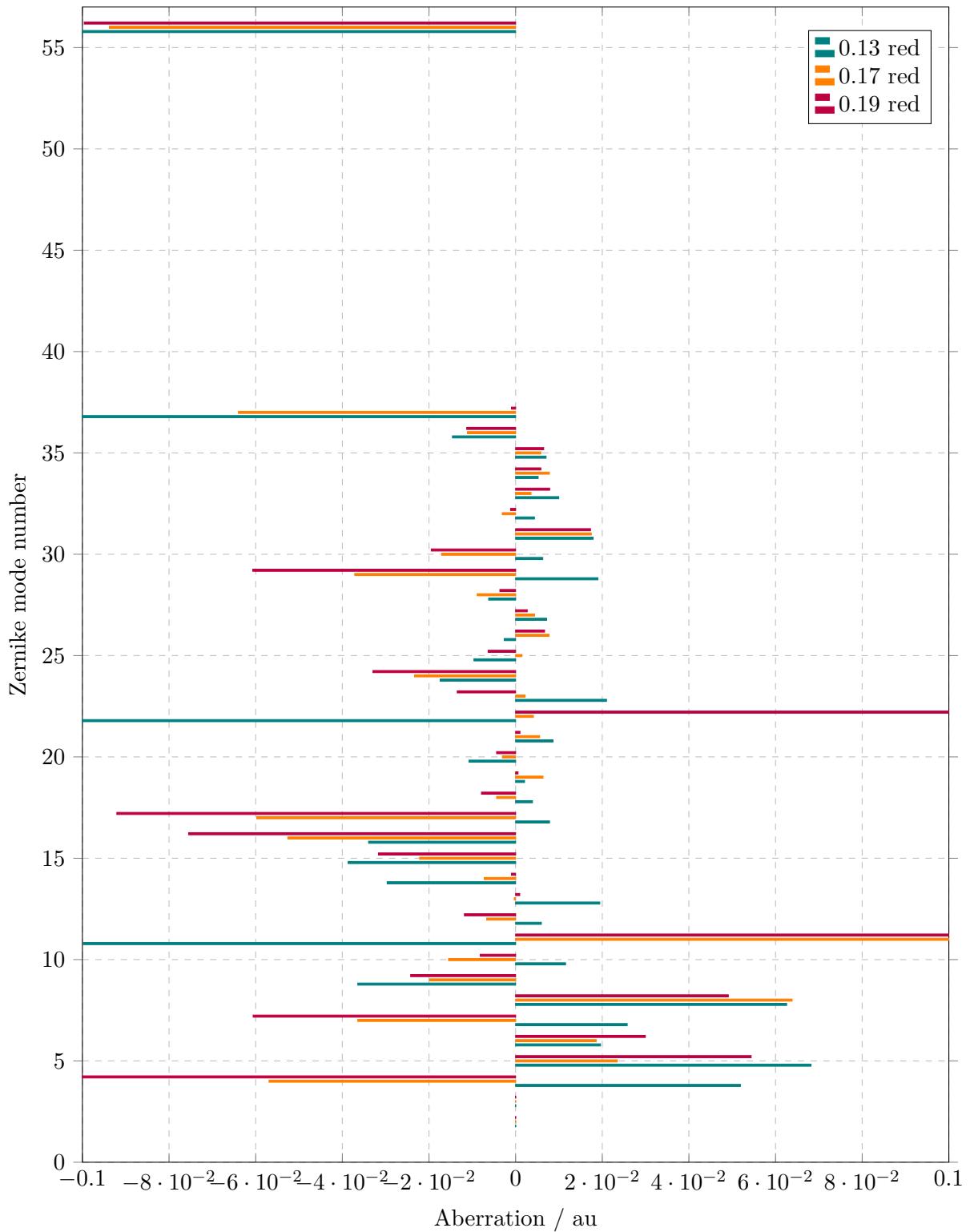


Figure 3.2: Red channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for all three correction collar settings (0.13, 0.17, 0.19).

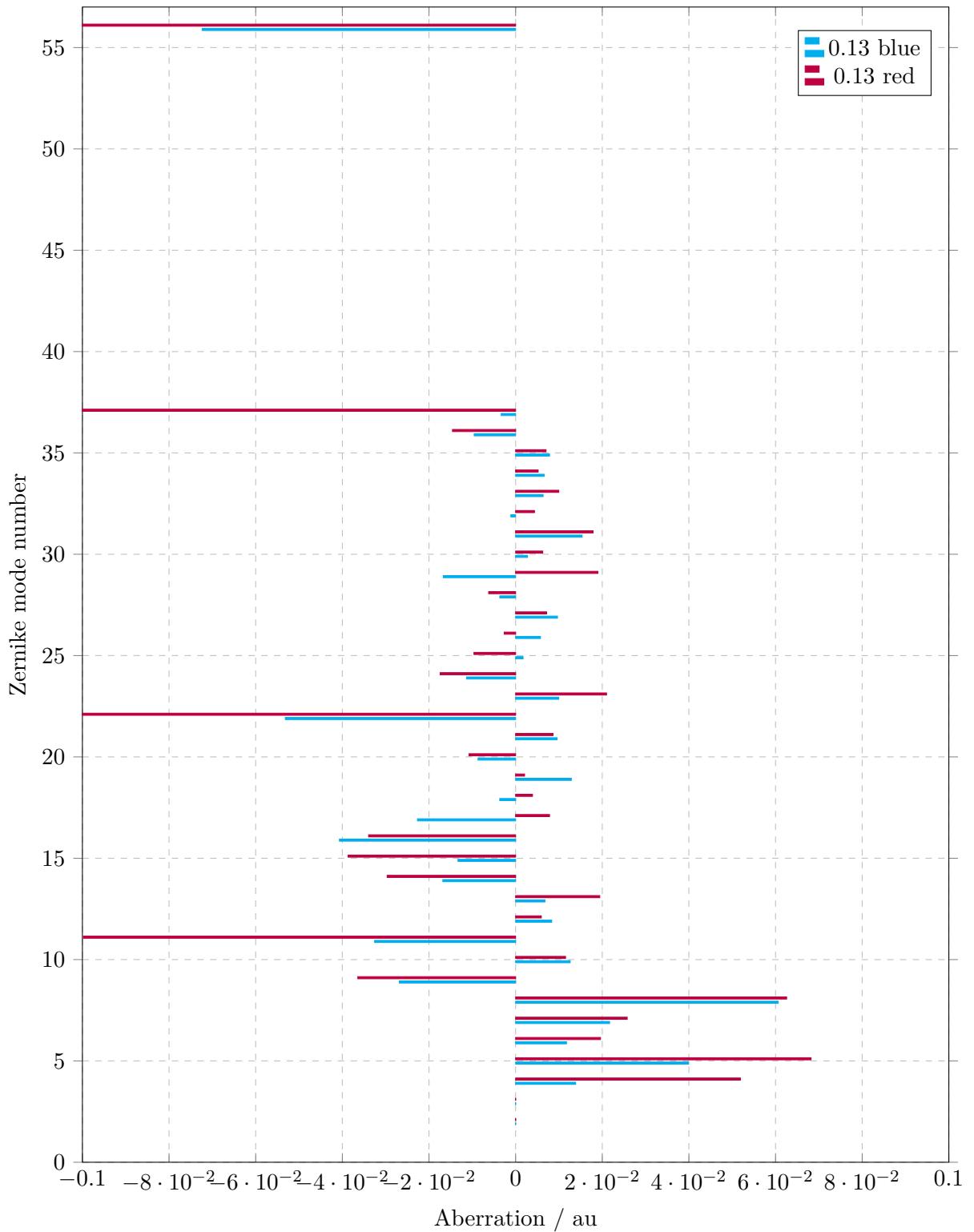


Figure 3.3: Red and blue channel Zernike modes {1 … 37, 56} versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.13.

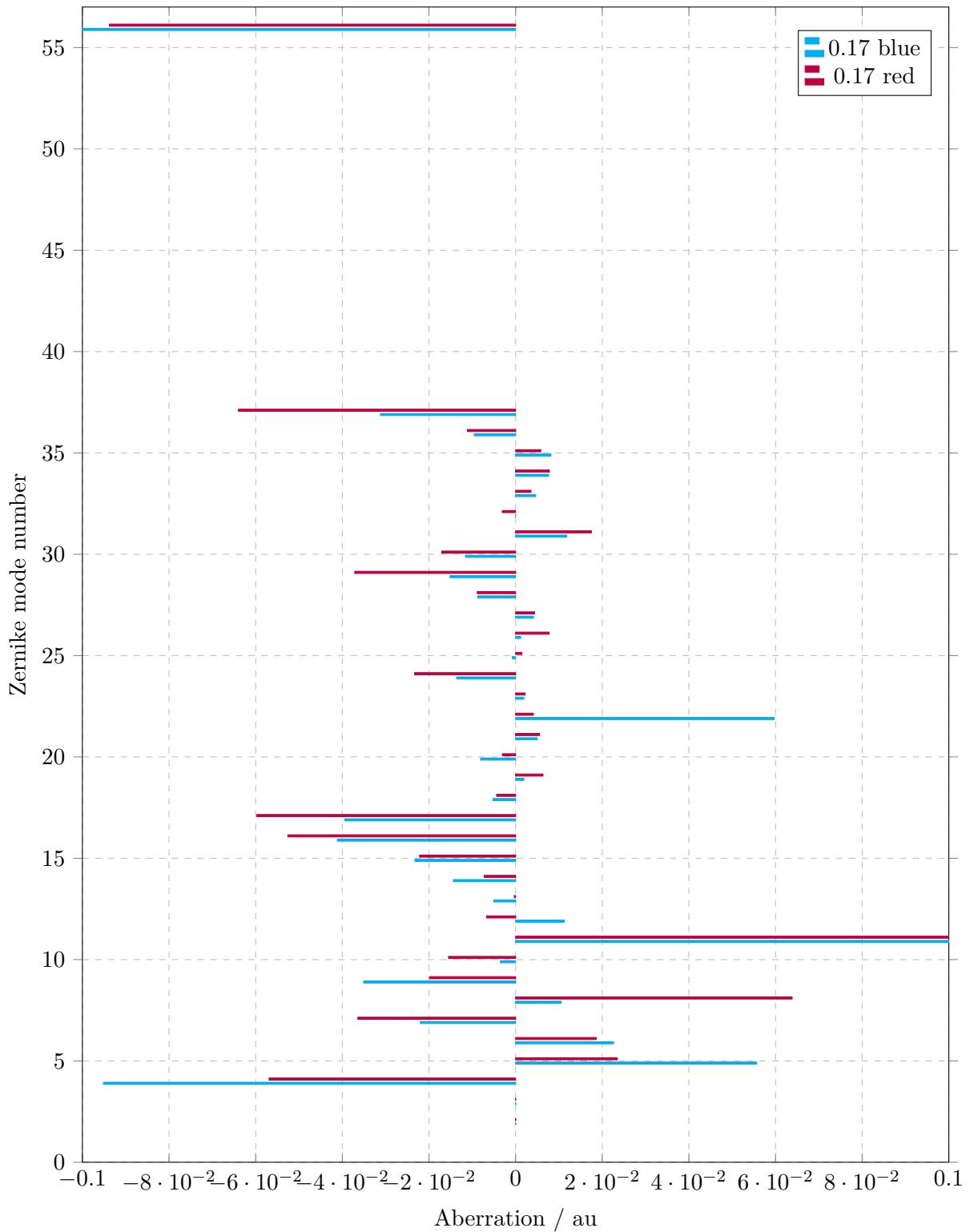


Figure 3.4: Red and blue channel Zernike modes {1...37,56} versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.17.

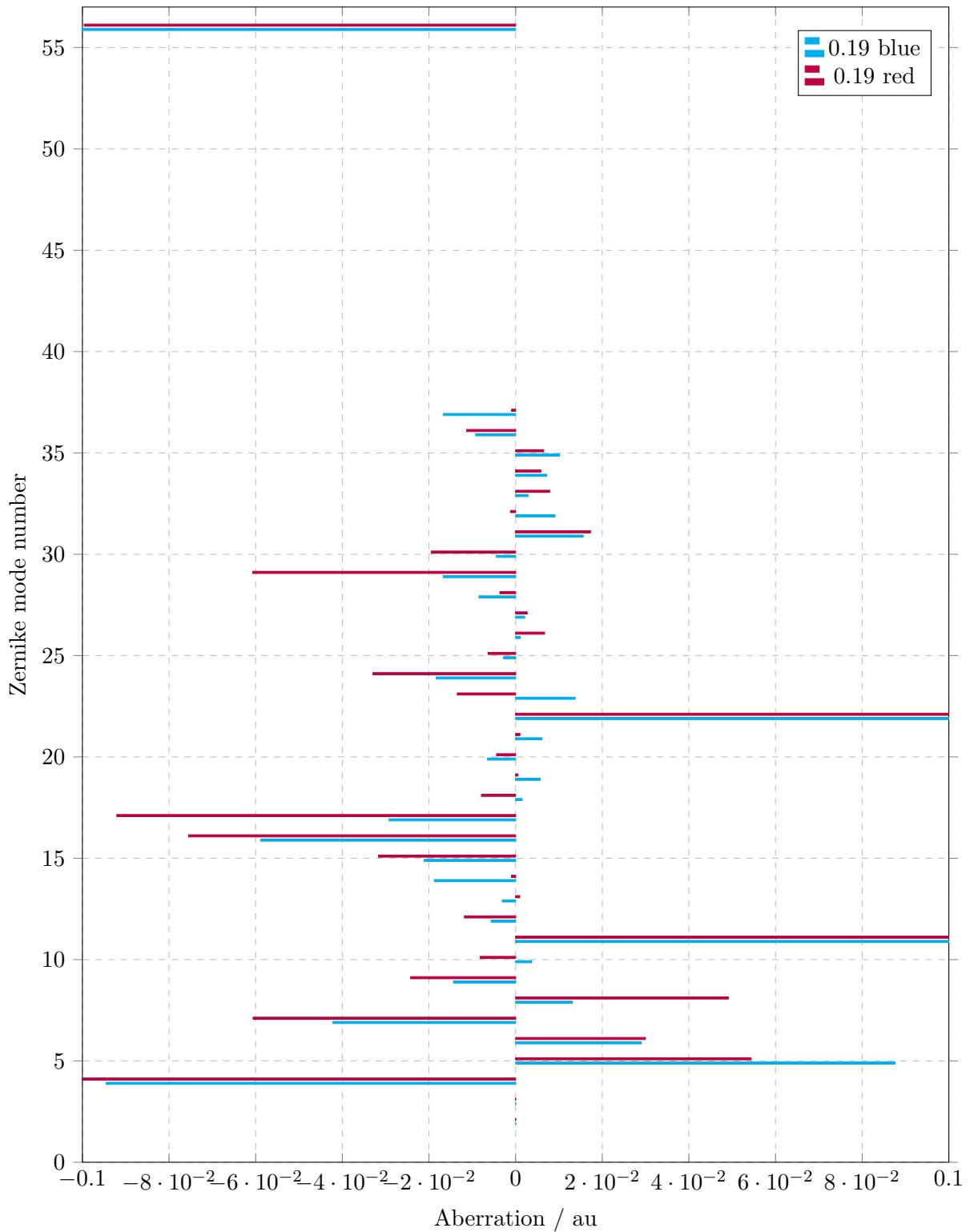


Figure 3.5: Red and blue channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.19.

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 (varying 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.

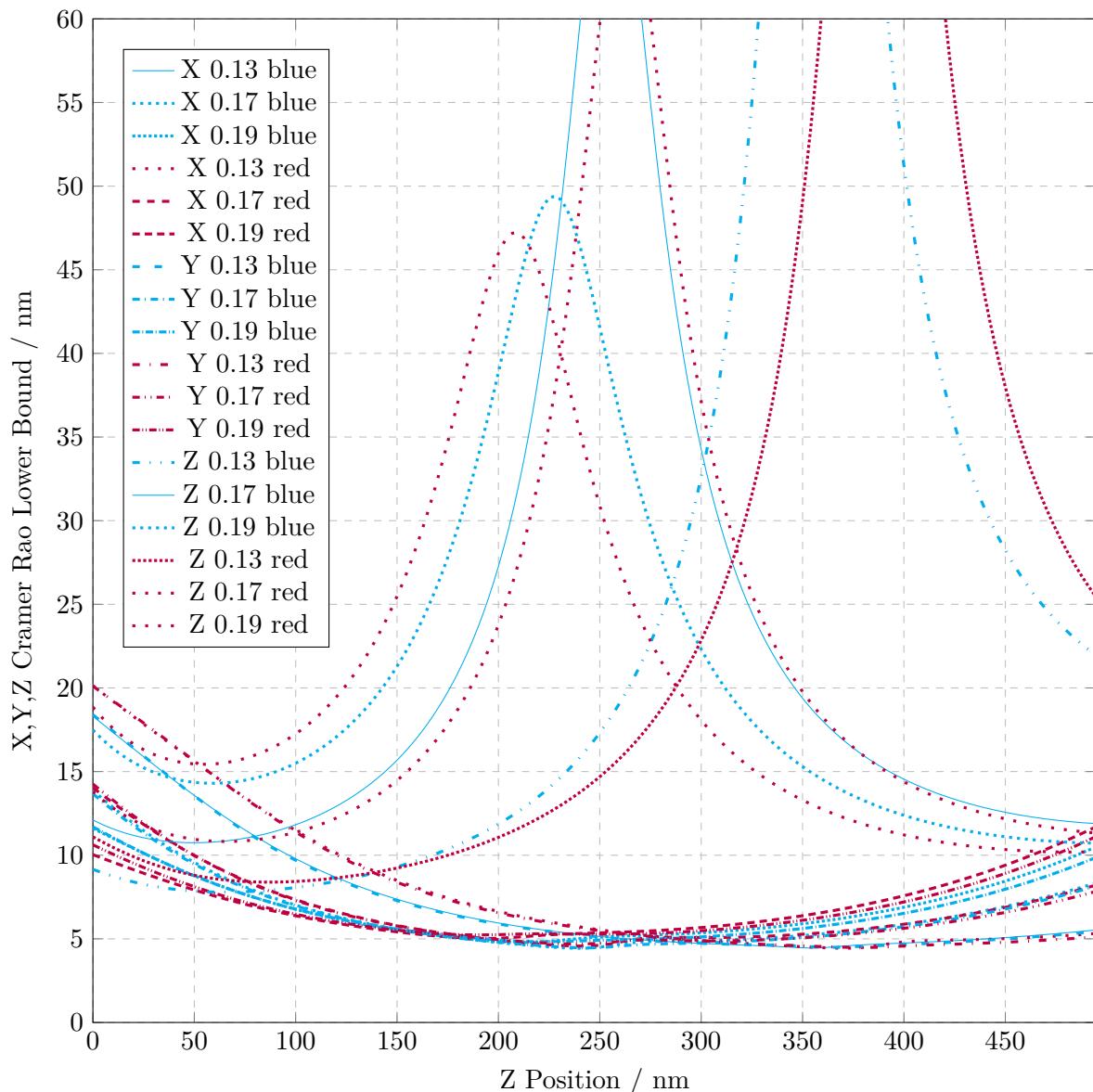


Figure 3.6: Estimated Cramer Rao Lower Bound for different correction Collar settings (varying line styles for 0.13, 0.17, 0.19) of the Olympus 1.5 NA objective; for X, Y and Z axis (top, middle, and bottom); for both red and blue channel (colors).

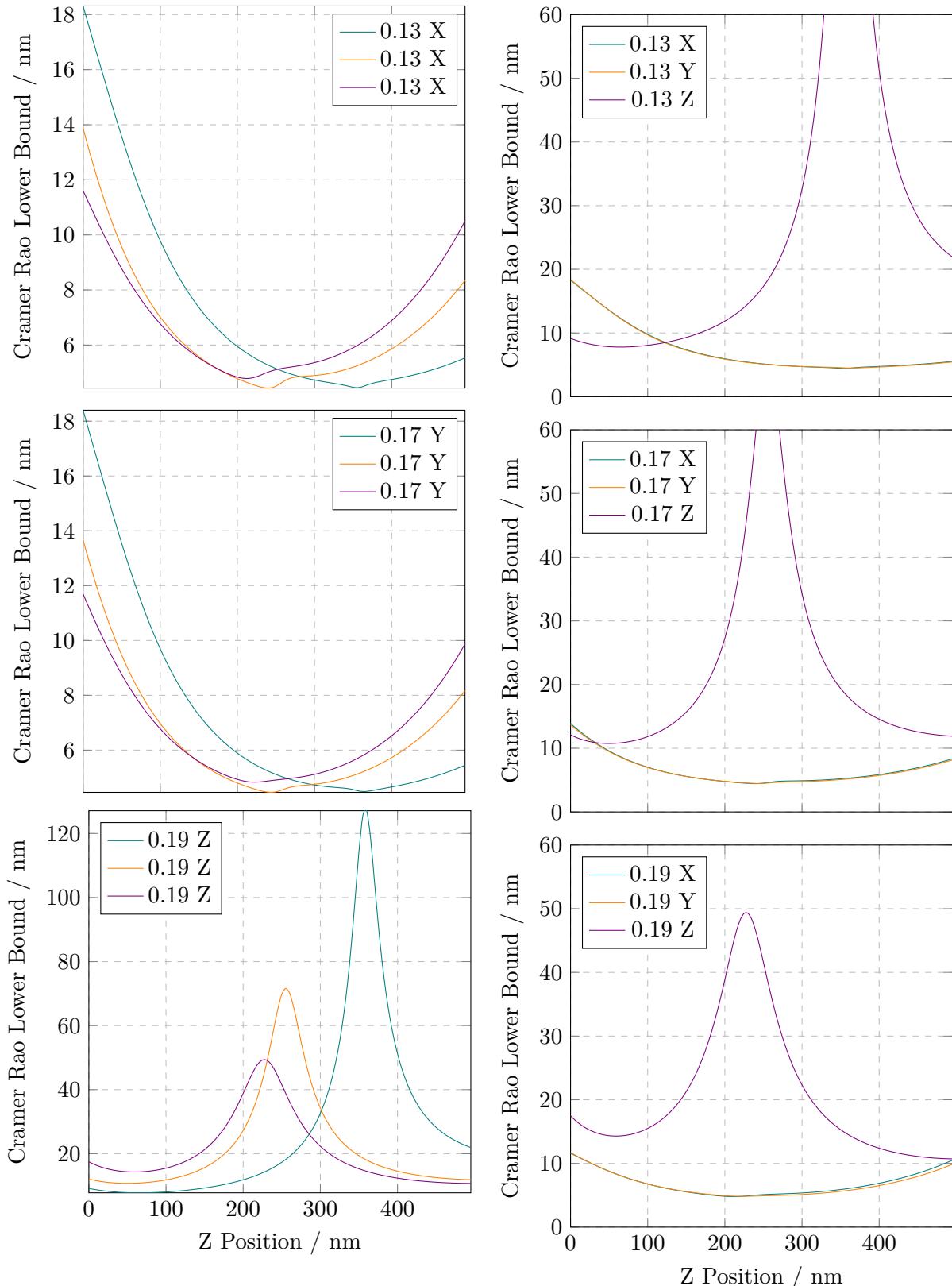


Figure 3.7: Estimated Cramer Rao Lower Bound for different correction Collar settings¹⁴ (teal: 0.13, orange: 0.17, purple: 0.19) of the Olympus 1.5 NA objective; grouped by X, Y and Z axis (top, middle, and bottom).

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 Transformation

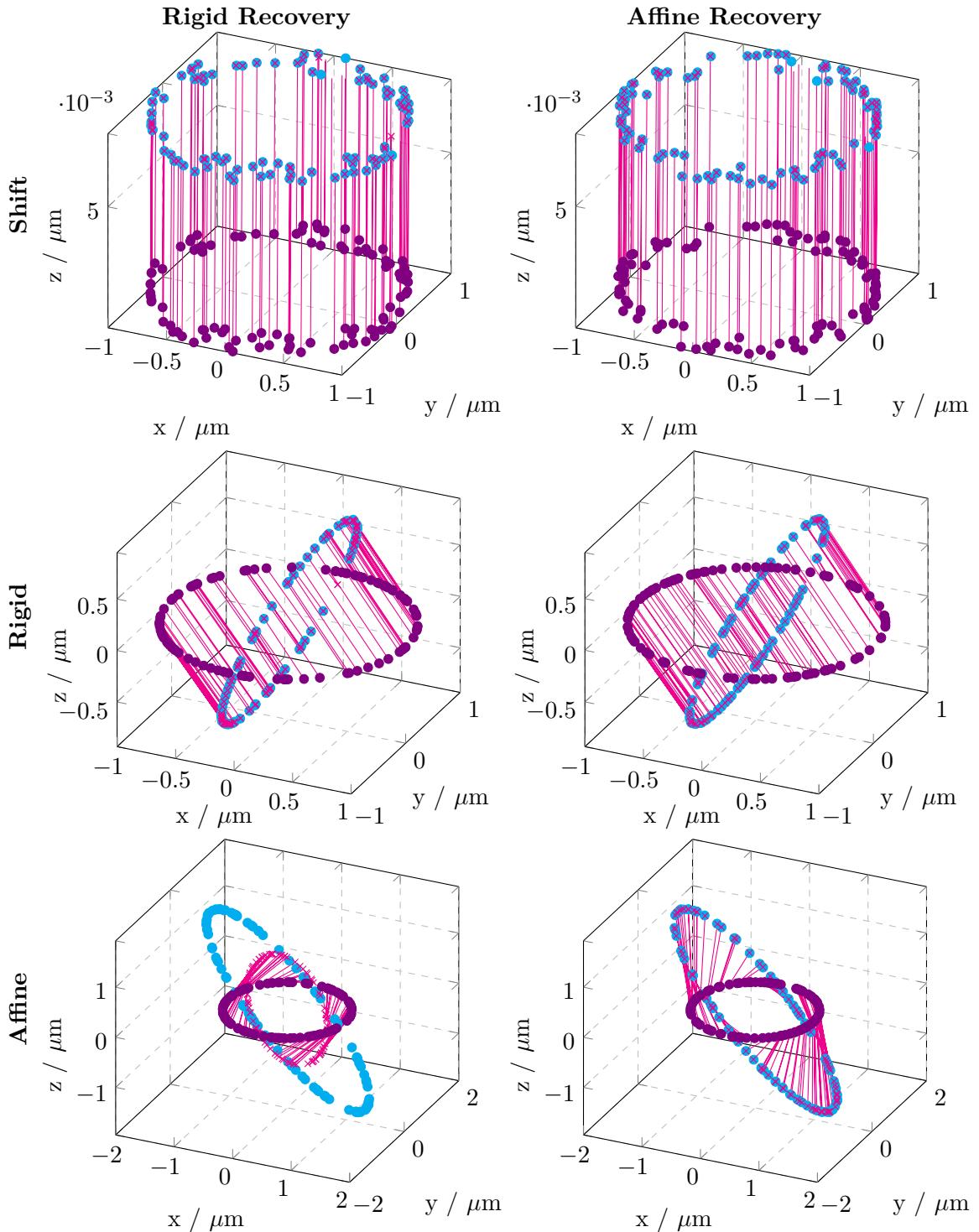


Figure 3.9: Demonstration of the recovery of rigid (left) respective affine (right) transformations, via recovering localisations of simulated two channel SMLM data; transformation (magenta) from red channel (violet) to blue channel (cyan); for the use cases of translation (top), rigid transformation (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

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).

[npc image](#)

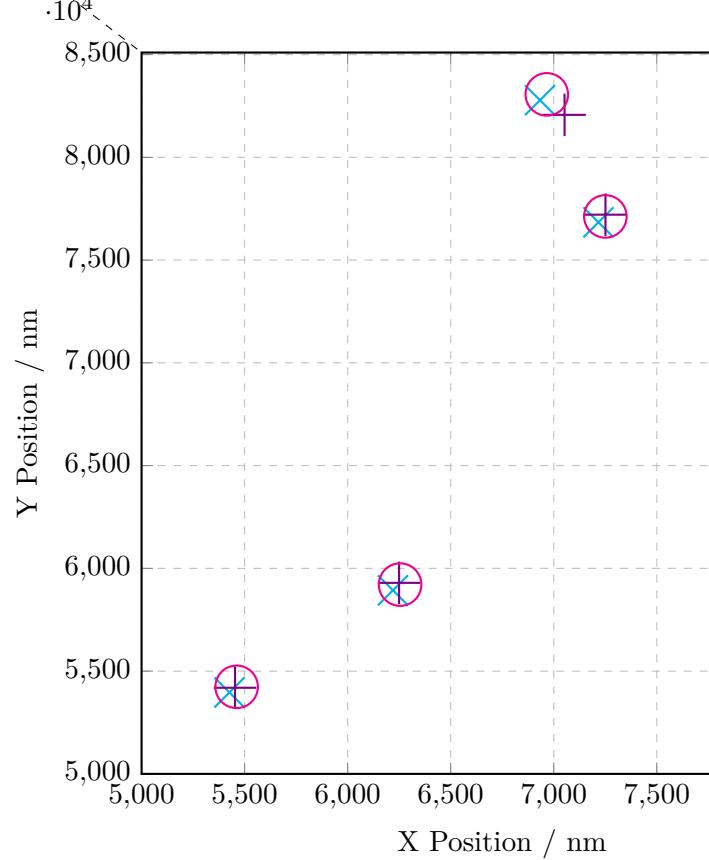
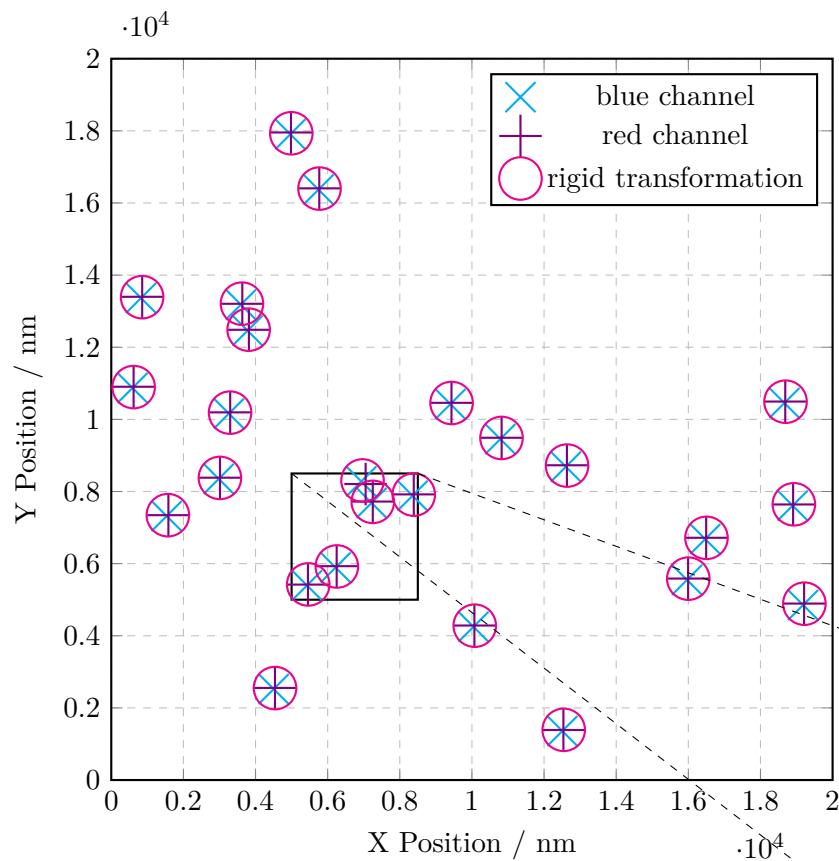


Figure 3.10: Demonstration of a rigid transformation of the localisations (magenta \circ) from blue channel (blue \times) to red channel (violet $+$); the transformed blue channel localisations mostly align well with the red channel localisations. 18

3.5.1 Import

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

```
1  #@markdown ##import core
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import numpy as np
5  import matplotlib as mpl
6  from mpl_toolkits.mplot3d import Axes3D
7  import scipy.io
8  from sklearn.cluster import DBSCAN
9  from sklearn import metrics
10 from sklearn.datasets import make_blobs
11 from sklearn.preprocessing import StandardScaler
12
13 #@markdown ##import jupyter
14 #%%matplotlib ipympl
15 #%%matplotlib widget
16 #%%matplotlib interactive
17 %matplotlib inline
18 import trackpy
19 #import sdt
20 #from sdt import io, chromatic, multicolor, brightness
21
22 ## local
23 wd = 'data/210422_npc_red_defocus/'
24 data = pd.read_csv( wd + 'cell1_tr1000_def500.csv',
25                     header = None,
26                     names=["x", "y", "z", "n", "bg","fit","id","frame"] )
```

3.5.2 Drift Correction

The *drift* of the setup over time can be estimated using *ImageJ*, and is then imported as `drift`. In the following block the drift correction is applied to all the 130k localisations.

```
1  #@markdown ## import & scale drift
2  #@markdown > set **magnification** via factor in drift.
3
4  drift = pd.read_csv( wd + 'day2_cell1_driftValues.csv')
5
6  drift['Y2']=drift['Y2']*146.6
7  drift['Y3']=drift['Y3']*146.6
```

```

8  drift['X2']=np.round(drift['X2'])
9  drift['X3']=np.round(drift['X3'])
10
11 #@markdown ## apply drift correction
12
13 for i in range(len(drift)-1):
14     fr=data[(data['frame']>=drift['X2'].iloc[i]) &
15             (data['frame']<drift['X2'].iloc[i+1])]
16     fr['y']=fr['y']-drift['Y2'].iloc[i]
17     fr['x']=fr['x']-drift['Y3'].iloc[i]
18     data[(data['frame']>=drift['X2'].iloc[i]) &
19             (data['frame']<drift['X2'].iloc[i+1])]=fr

```

drift corrected localizations: 370148

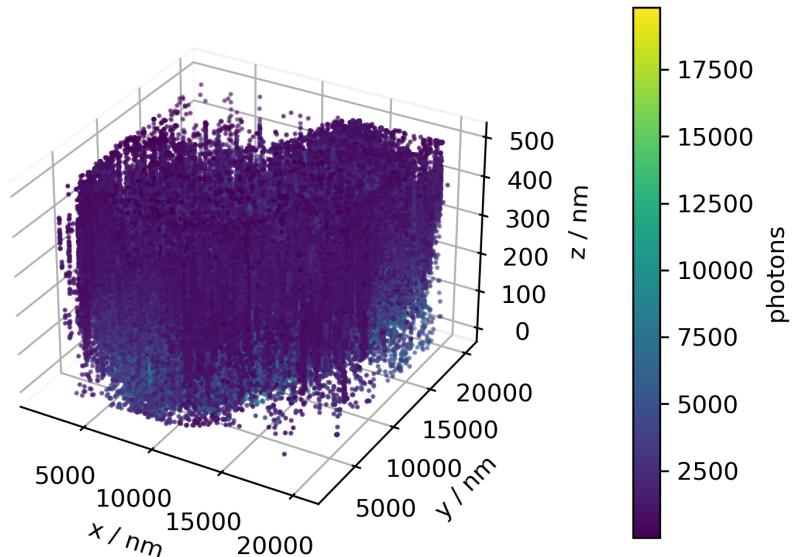


Figure 3.11: 3d plot of all 370148 drift corrected localisations, color coded by photon count.

3.5.3 Check Photon Counts

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

```

1  #@markdown ## Check: photon counts
2  #@markdown > set `max_photons` accordingly (default: 10000).

```

```

3 max_photons = 10000 #@param {type:"slider", min:0, max:40000, step:1000}
4
5 fig = plt.figure()
6 plt.hist( data['n'], bins=50, range=( 0, max_photons ) )

```

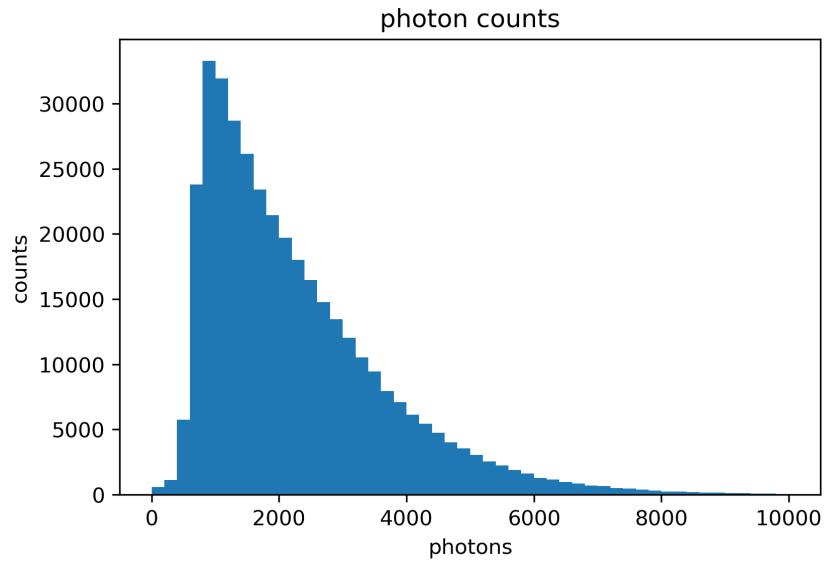


Figure 3.12: histogram of the photon count for the localisations.

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

```

1 #@markdown ## filter
2 #@markdown > set `min_photons` and `max_photons` accordingly (default: 2000 < photons < 7000)
3 #@markdown > set `min_z` and `max_z` accordingly (default: 0 < z < 499). \
4 #@markdown > set `min_fit` accordingly (default: 6e6).
5 min_photons = 2000 #@param {type:"slider", min:0, max:40000, step:1000}
6 max_photons = 7000 #@param {type:"slider", min:0, max:40000, step:1000}
7 min_z = 0 #@param {type:"slider", min:0, max:500, step:1}
8 max_z = 384 #@param {type:"slider", min:0, max:500, step:1}
9 min_fit = 7192000 #@param {type:"slider", min:0, max:1e7, step:1000}
10

```

```

11 fdata = data[ ( data['n'] > min_photons ) &
12             ( data['n'] < max_photons ) &
13             ( data['z'] > min_z ) &
14             ( data['z'] < max_z ) &
15             ( data['fit'] < min_fit ) ]

```

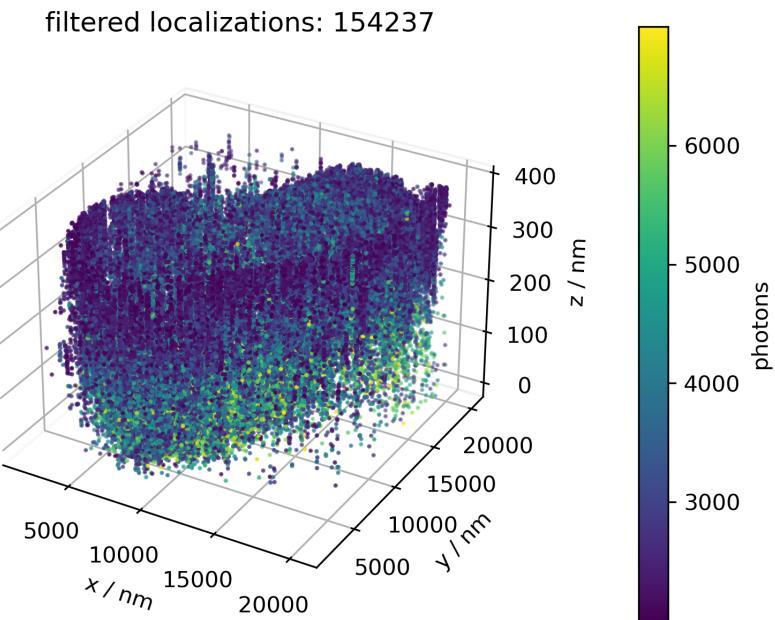


Figure 3.13: 3d plot of all 154237 filtered localisations, color coded by photon count.

3.5.5 Track Particles

```

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

6
7  linkedxy = trackpy.link_df( fdata,
8      pos_columns = ["x", "y", "z"],
9      search_range = sr,
10     memory = mem )

11
12 particles = linkedxy.groupby( "particle" ).aggregate( np.mean )
13 std_pos = linkedxy.groupby( "particle" ).aggregate( 'std' )
14 particles["length"] = linkedxy.groupby( "particle" ).apply( len )

```

```

15 particles["z_std"] = std_pos['z'].copy()
16 particles["x_std"] = std_pos['x'].copy()
17 particles["y_std"] = std_pos['y'].copy()

```

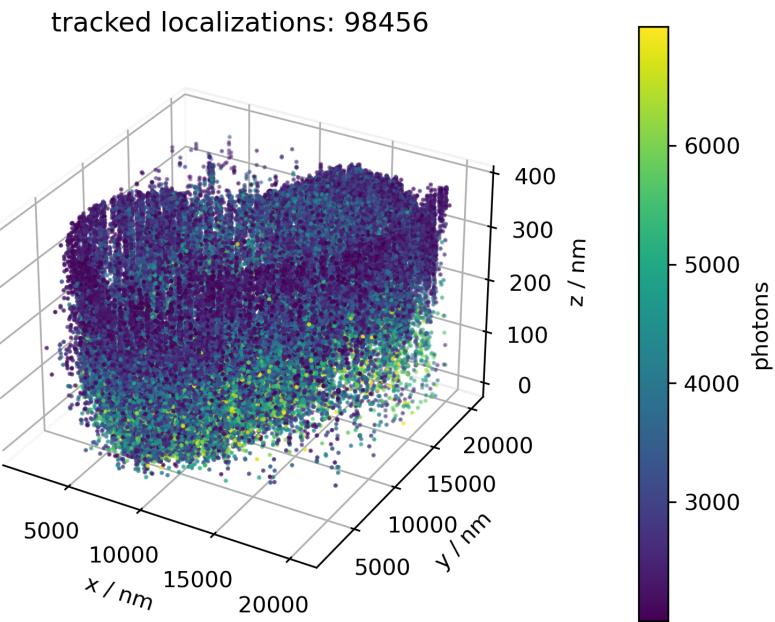


Figure 3.14: 3d plot of all 98456 filtered localisations, color coded by photon count.

3.6 SMLM Analysis: NPC

3.6.1 Clustering

```

1 #@markdown ## compute dbscan
2 #@markdown > set `dim = 2` for clustering in x and y (default:).\ \
3 #@markdown > set `dim = 3` for experimental clustering in 3d; be aware that x,z and y preci
4 #@markdown > set `eps` (default: 200).\ \
5 #@markdown > set `min_samples` (default: 10).
6 dim = "2" #@param [2, 3]
7 eps = 100 #@param {type:"slider", min:0, max:500, step:10}
8 min_samples = 50 #@param {type:"slider", min:1, max:100, step:1}
9
10 alocalisations = localisations.to_numpy()
11 alocalisations[ :, 0:2 ]
12
13 db = DBSCAN( eps, min_samples ).fit( alocalisations[ :, 0:int( dim ) ] )

```

```

14 core_samples_mask = np.zeros_like( db.labels_, dtype=bool )
15 core_samples_mask[ db.core_sample_indices_ ] = True
16 labels = db.labels_
17 localisations[ "cluster" ] = labels
18
19 # count clusters (ignore noise if present)
20 n_clusters_ = len( set( labels ) ) - ( 1 if -1 in labels else 0 )
21 n_noise_ = list( labels ).count( -1 )
22
23 #print('Estimated number of clusters: %d' % n_clusters_)
24 #print('Estimated number of noise points: %d' % n_noise_)
25
26 nlocalisations = localisations.loc[ localisations['cluster'] == -1 ]
27 clocalisations = localisations.loc[ localisations['cluster'] != -1 ]

```

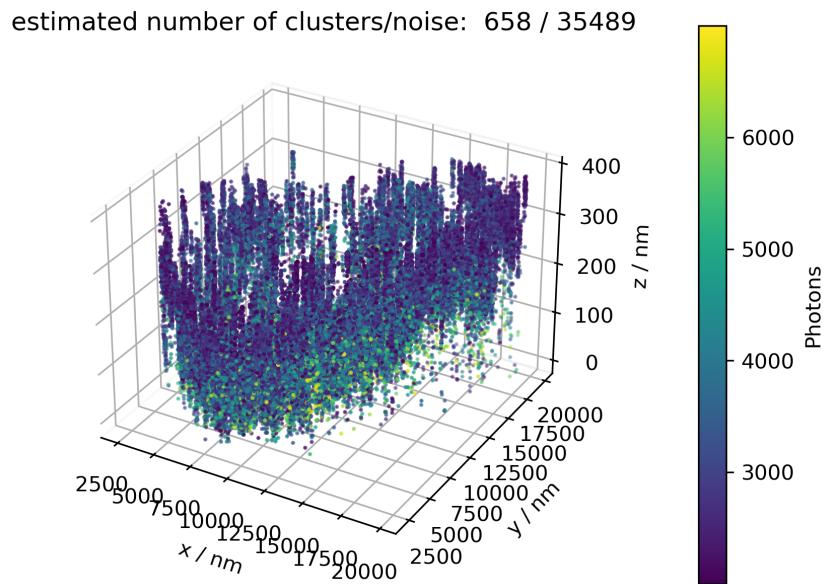


Figure 3.15: 3d plot of the localisations of the 658 identified clusters (omitting 35489 localisations as noise, due to not belonging to a cluster), color coded by photon count.

3.6.2 Cluster Analysis

```

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

```

```

6  clabels = set(labels)
7  cnames = [ "counts", "xmean", "ymean", "zmean", "nmean", "xvar", "yvar", "zvar",
8             "nvar", "label", "ringness", "twofold" ]
9
10 clusters = pd.DataFrame( index = clabels, columns = cnames, dtype="float64" )
11 clusters[ "label" ] = clabels
12
13 for k in clabels:
14     tmp = localisations.loc[ localisations['cluster'] == k ]
15     clusters.loc[ k, "counts" ] = len( tmp )
16     for label in [ "x", "y", "z", "n" ]:
17         clusters.loc[ k, label+"mean" ] = np.mean( tmp.loc[ :, label ] )
18         clusters.loc[ k, label+"var" ] = np.var( tmp.loc[ :, label ] )
19
20     ## xy: radius (distance to centroid)
21     rad = np.sqrt( ( tmp.loc[ :, "x" ] - clusters.loc[ k, "xmean" ] )**2 +
22                     ( tmp.loc[ :, "y" ] - clusters.loc[ k, "ymean" ] )**2 )
23     ## xy: radius rms deviation from NPC radius
24     clusters.loc[ k, "ringness" ] = np.sqrt( sum( ( rad - npc_radius )**2 ) )
25
26     ## z: radius (distance to centroid)
27     rad = abs( tmp.loc[ :, "z" ] - clusters.loc[ k, "zmean" ] )
28     ## z: radius rms deviation from NPC radius
29     clusters.loc[ k, "twofold" ] = np.sqrt( sum( ( rad - npc_height )**2 ) )
30
31 #@markdown ## filter clusters
32 #@markdown > set `count_threshold` to min elements (counts) in cluster (default: 30)\n
33 #@markdown > set `diameter_threshold` to max x,y (radius) deviation of cluster from NPC dia
34 #@markdown > set `twofold_threshold` to max z (radius) deviation of cluster from NPC height
35 #@markdown > set `xyvar_threshold` to wanted x,y variance (default: 1e4)\n
36 #@markdown > set `zvar_threshold` to wanted z variance (default: 1e4)
37 count_threshold = 100 #@param {type:"slider", min:0, max:200, step:1}
38 diameter_threshold = 500 #@param {type:"slider", min:0, max:500, step:10}
39 twofold_threshold = 1000 #@param {type:"slider", min:0, max:1000, step:10}
40 xyvar_threshold = 100000 #@param {type:"slider", min:0, max:1e5, step:1e3}
41 zvar_threshold = 100000 #@param {type:"slider", min:0, max:1e5, step:1e3}
42
43 fclusters = clusters[ ( clusters['counts'] < count_threshold ) &
44                         ( clusters['ringness'] < diameter_threshold ) &
45                         ( clusters['twofold'] < twofold_threshold ) &
46                         ( clusters['xvar'] < xyvar_threshold ) &
47                         ( clusters['yvar'] < xyvar_threshold ) &
48                         ( clusters['zvar'] < zvar_threshold ) ]

```

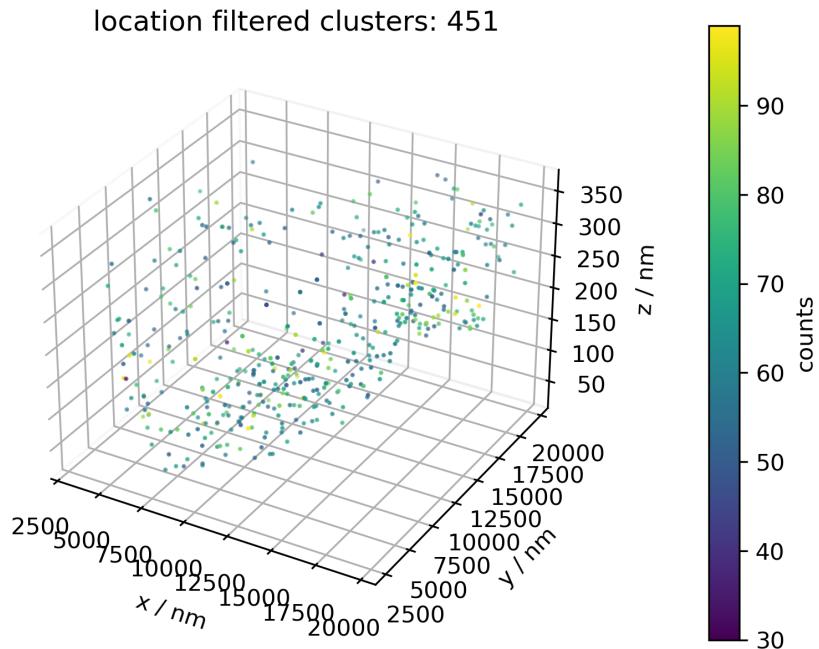


Figure 3.16: 3d plot of the positions of the 451 filtered clusters, color coded by photon count.

3.6.3 Select Clusters

```

1  # @markdown ## select best clusters & plot localisations
2  # @markdown > set `show_clusters` to wanted number of best clusters (default: 100). \
3  # @markdown > set `sort` to sort the best clusters (default: ringness + twofold).
4  show_clusters = 10 # @param {type:"slider", min:0, max:1000, step:1}
5  sort = 'ringness + twofold' # @param [ "ringness + twofold", "twofold", "ringness", "xyvar"
6
7  if sort == "xvar": # sort by variance
8      sclusters = fclusters.sort_values( "xvar" )
9  elif sort == "xyvar": # sort by x and y variance using least squares
10     sclusters = fclusters.loc[
11         ( fclusters.xvar ** 2 + fclusters.yvar ** 2 ).sort_values().index ]
12 elif sort == "ringness": # sort by ringness (deviation to ringness)
13     sclusters = fclusters.sort_values( "ringness" )
14 elif sort == "twofold": # sort by ringness (deviation to ringness)
15     sclusters = fclusters.sort_values( "twofold" )
16 elif sort == "ringness + twofold": # sort by ringness (deviation to ringness)
17     sclusters = fclusters.loc[
18         ( fclusters.twofold ** 2 + fclusters.ringness ** 2 ).sort_values().index ]
19
20 show_clusters = min( show_clusters, len( sclusters ) )

```

```

21 selected_clusters = sclusters["label"].iloc[ 0:show_clusters ]
22
23 fig = plt.figure()
24 ax = fig.add_subplot( projection = '3d' )
25 for clus in selected_clusters:
26     flocalisations = localisations[ ( localisations['cluster'] == clus ) ]
27     ff = ax.scatter( flocalisations['x'],
28                      flocalisations['y'],
29                      flocalisations['z'],
30                      s=1 ,c = flocalisations['n'] )

```

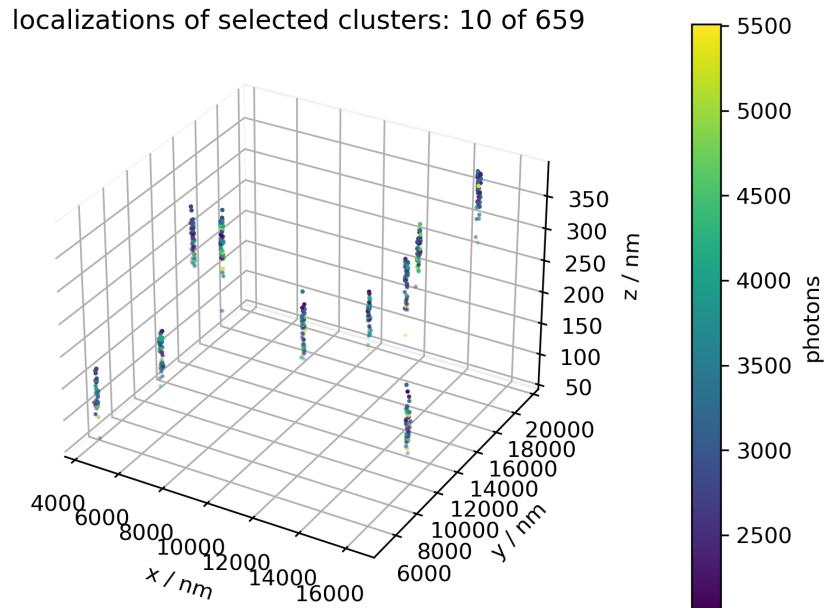


Figure 3.17: 3d plot of the positions of the 451 filtered clusters, color coded by photon count.

3.6.4 X,Y,Z Histograms

```

1 #@markdown ## Check: z distribution
2 #@markdown > set `plot_cluster` to wanted cluster (default: 0).
3 plot_cluster = 3 #@param {type:"slider", min:0, max:100, step:1}
4 plot_cluster = min( plot_cluster, len( sclusters ) -1 )
5
6 tmp = localisations.loc[ localisations['cluster'] ==
7                         sclusters.loc[ sclusters.index[ plot_cluster ],
8                           "label" ] ]

```

```
10 z = tmp.z - np.mean( tmp.z )
11
12 fig, axes = plt.subplots(2, 1, figsize=(4, 7)) # figsize=(4, 12)
13
14 axes[0].hist( tmp.z - np.mean( tmp.z ) )
15 axes[0].set_xlabel('z / nm')
16 axes[0].set_ylabel('counts')
17 #axes[0].set_title( 'z' )
18
19 axes[1].hist( tmp.x - np.mean( tmp.x ) )
20 axes[1].hist( tmp.y - np.mean( tmp.y ) )
21 axes[1].set_ylabel('counts')
22 axes[1].set_xlabel('x, y / nm')
23 #axes[1].set_title( 'x, y' )
```

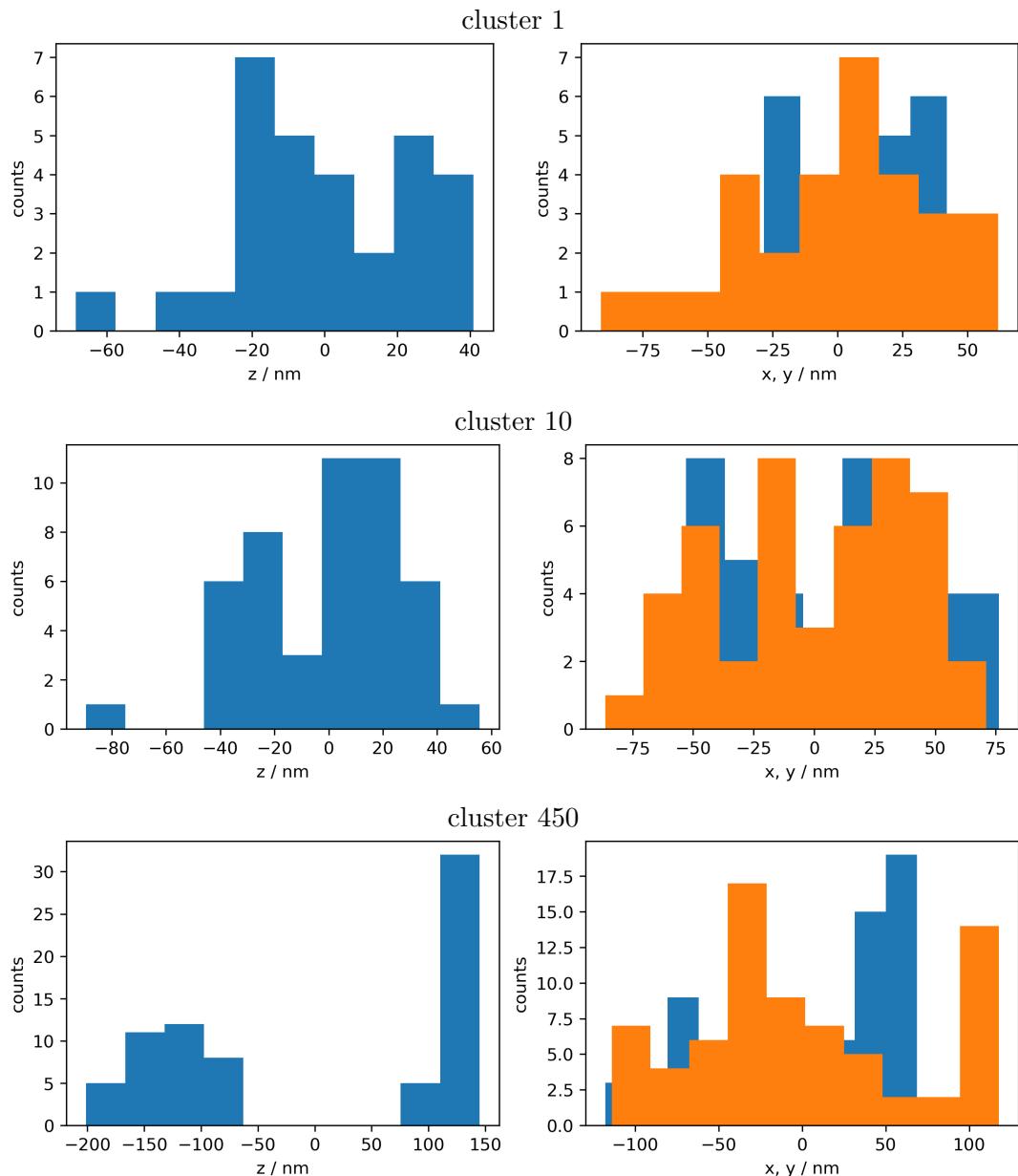


Figure 3.18: histogram of the x, y (right) respective z distribution (left) of the localisations within the *i*best cluster (top), the *i*worst (bottom) and one in between.

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 inter-

ested 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?

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 0.17. Here we find the preferable compromise between x,y precision and z precision, in the regime between about 0 and 250 μm .

5.3 dSTORM Buffer

List of Figures

3.1	Blue 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).	7
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).	8
3.3	Red and blue channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.13.	9
3.4	Red and blue channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.17.	10
3.5	Red and blue channel Zernike modes $\{1 \dots 37, 56\}$ versus aberrations of PSF model via phase retrieval, for correction collar setting of 0.19.	11
3.6	Estimated Cramer Rao Lower Bound for different correction Collar settings (varying 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).	13
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).	14
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.	14
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).	16
3.10	Demonstration of a rigid transformation of the localisations (magenta \circ) from blue channel (blue \times) to red channel (violet $+$); the transformed blue channel localisations mostly align well with the red channel localisations.	18
3.11	3d plot of all 370148 drift corrected localisations, color coded by photon count. .	20
3.12	histogram of the photon count for the localisations.	21
3.13	3d plot of all 154237 filtered localisations, color coded by photon count.	22
3.14	3d plot of all 98456 filtered localisations, color coded by photon count.	23
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.	24

3.16	3d plot of the positions of the 451 filtered clusters, color coded by photon count.	26
3.17	3d plot of the positions of the 451 filtered clusters, color coded by photon count.	27
3.18	histogram of the x,y (right) respective z distribution (left) of the localisations within the ibest cluster (top), the iworst (bottom) and one in between.	29