Alma Mater Studiorum · Università di Bologna

Corso di Laurea Magistrale in Matematica

Sparsity-enforcing optimization for 3D single-molecule localisation in genomic imaging

Tesi di Laurea Magistrale in Mathematical and Machine Learning Methods in Imaging

Relatrice:
Chiar.ma Prof.ssa
Serena Morigi

Presentata da:
Jacopo Betti

Correlatori:
Prof. Luca Calatroni
Dr.ssa Irene Farabella

Anno Accademico 2024-2025

Contents

Introduction

In recent years, advancements in genomic imaging have enabled the investigation of the 3D organization of chromosomes within the cell nucleus. Among the techniques developed for this purpose, Oligopaints-based Stochastic Optical Reconstruction Microscopy (STORM) [1] enables the acquisition of high-resolution images of specific genomic regions, which can be used to trace the chromatin path.

In particular, images obtained with Oligopaints-based STORM can be employed for Volumetric Chromatin Tracing, a technique that preserves the 3D topographical arrangement of genomic regions. This method requires capturing thousands of images for each region, identifying active fluorophores in them, and using their localized positions to reconstruct the chromatin path. However, accurate chromatin tracing relies on localization algorithms that identify emitters with nanometric precision, which is a technically challenging task. Thus, the development of advanced 3D single-molecule localization algorithms is an important problem in genomic imaging.

This thesis focuses on the mathematical modeling and algorithmic development for 3D single-molecule localization in STORM imaging. The work was carried out during an internship at the Italian Institute of Technology under the supervision of Dr. Irene Farabella, in collaboration with Prof. Luca Calatroni. In particular, a forward mathematical model of the STORM image acquisition process was developed, and based on this model, an image simulator for STORM data was implemented, enabling the generation of synthetic datasets for numerical experiments.

Furthermore, the localization problem was formulated as a regularized inverse problem, incorporating sparsity-promoting regularization terms to recover the 3D coordinates of emitters from STORM images. Numerical methods were implemented and evaluated, and existing 3D single-molecule localization algorithms and software packages, such as ImageJ, Fiji, and ThunderSTORM, were studied and compared.

This thesis is organized as follows:

- The first chapter introduces the biological and microscopy-related background relevant to the 3D localization problem, with a particular focus on genomic imaging

and 3D single-molecule localization microscopy;

- The second chapter presents the mathematical model of the STORM image acquisition process;
- The third chapter discusses the regularized inverse problem used for single-molecule localization in detail, including the sparsity-promoting regularization functions considered, the numerical methods used to solve the problem, and the underlying theoretical framework;
- The fourth chapter presents the datasets used for numerical experiments, the tested algorithms, the obtained results, and the validation metrics used to assess their performance.

Chapter 1

Introduction to SMLM for genomic imaging

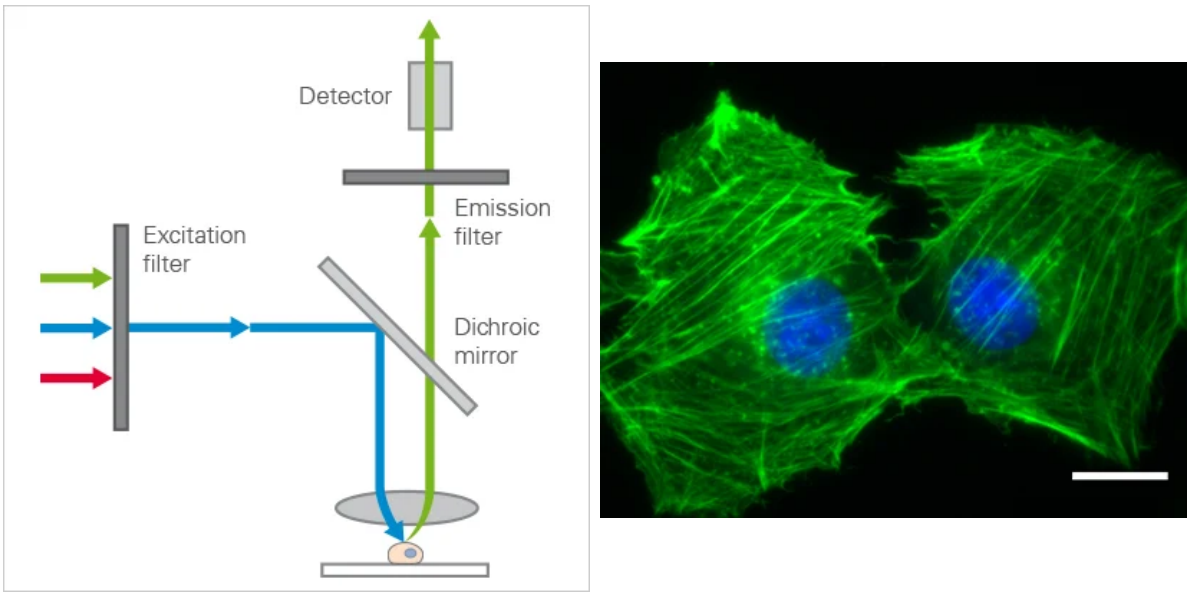
In this first chapter, we will introduce Single-Molecule Localization Microscopy (SMLM). In particular, we will discuss components relevant to the 3D localization problem; for example, how images of chromatin are obtained using fluorophores and how the 3D positions of these fluorophores can be recovered from 2D images.

1.1 SMLM for imaging

Microscopes are used to obtain images of samples, such as groups of cells. One type of microscope is the widefield microscope. Specimens are prepared according to the specific analysis, mounted on glass slides, and placed on the microscope stage. In fluorescence microscopy, images are acquired using a camera, lenses, a light source that emits light at a specific wavelength, and fluorophores within the sample. An example of image acquisition with a widefield microscope is shown in Fig 1.1a. These fluorophores are chemical compounds that absorb light at a particular wavelength and then emit light at a longer wavelength. When using a widefield microscope, we obtain only a single 2D image of the sample. In Fig 1.1b, we observe an image obtained with a widefield microscope. As we can see, it's hard to distinguish single emitters in this image. This problem is caused by light diffraction, a phenomenon that limits the resolution of the widefield microscope. For a microscope, the resolution is defined as the shortest distance between two points on a specimen that can still be distinguished as separate entities, and it is measured in nanometers. For widefield microscopes, the resolution is around 200-300 nm. If the distance between two emitters is less than the optical system resolution, we say they are unresolved; if it is greater than the resolution, we say they are resolved. In

Fig 1.2, we show examples of resolved and unresolved emitters.

If we want to localize single emitters in an image, we need special techniques. For this aim, SMLM methods have been developed.



(a) (b)

Figure 1.1: *Widefield microscopy*

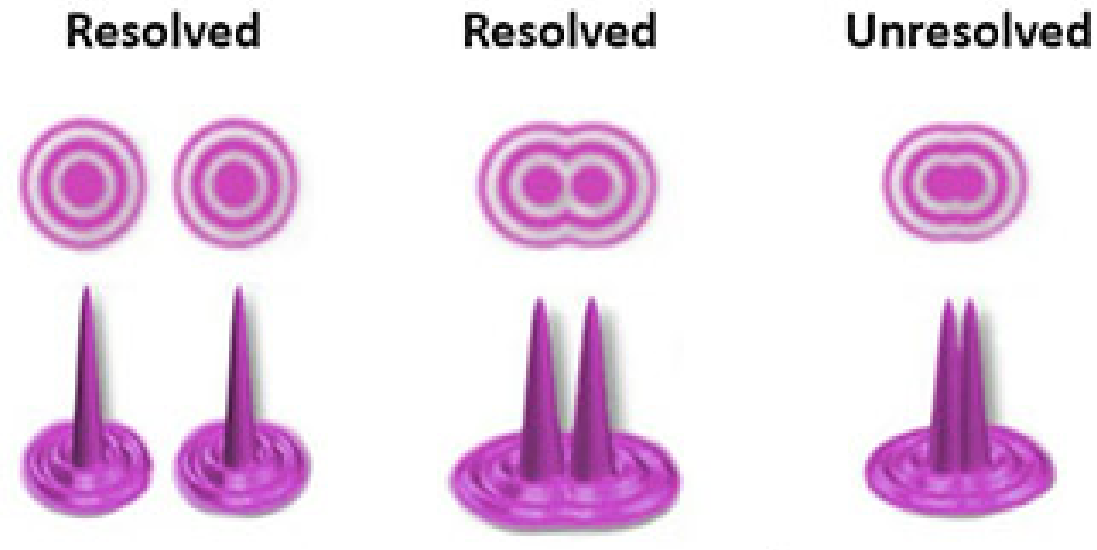


Figure 1.2: *Example of microscope resolution [3]*

1.1.1 Single-molecule localization microscopy techniques

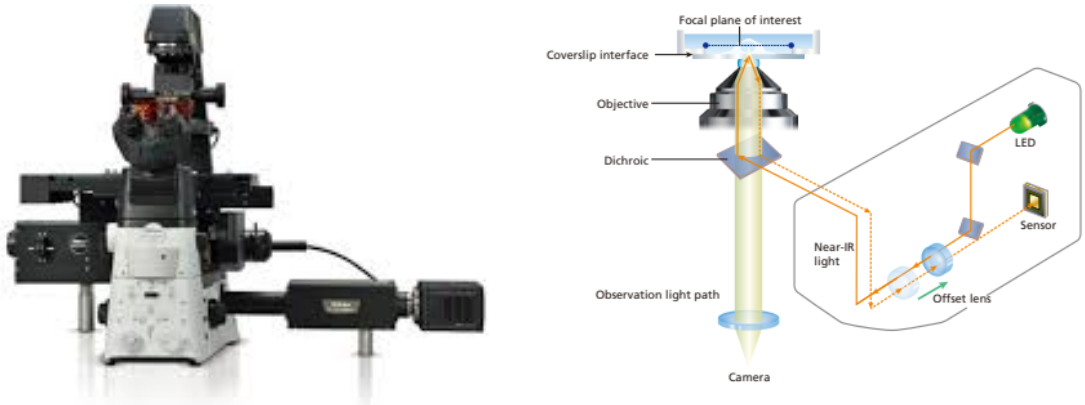
SMLM is a group of super-resolution techniques that bypass the diffraction limit of visible light, achieving a resolution of about 20 nm. In this group, two subsets are:

- Deterministic super-resolution methods [4], which rely on physically controlling the emission of active fluorophores;
- Stochastic super-resolution methods [5], which rely on alternating the fluorophores between an active state, in which they emit light, and a passive state, in which they remain dark;

The subset we are interested in is the second one. In particular, in this group, two techniques are:

- Stochastic Optical Reconstruction Microscopy (STORM) [5];
- Photoactivated Localization Microscopy (PALM) [6].

To use these techniques, specific microscopes are required. In particular, Nikon's N-STORM systems are designed for STORM imaging. In Fig 1.3a and Fig 1.3b, we can see the Nikon N-STORM Super Resolution System and how light travels inside the microscope.



(a) **(b)**

Figure 1.3: *Nikon N-STORM Super Resolution System [7]*

1.1.2 STORM

Applying STORM, small subsets of fluorophores are sequentially activated, one subset at a time. Each time a subset is activated, an image of it is captured, and the

emitters in the image are localized. To obtain one image, only a few milliseconds are needed. After capturing a photo of the emitters, they are deactivated, and another subset of fluorophores is activated. We repeat this process until every fluorophore has been illuminated.

Thus, using STORM yields a sequence of images, each captured in a specific time frame t_n, rather than a single image as in widefield microscopy. To reconstruct the object we want to analyze, we locate the emitters in all the photos and combine them into a single image.

Fig 1.4 shows the reconstruction of an object using a STORM movie, while Fig 1.5 illustrates 3 frames of a STORM movie. In Fig 1.6, the top row contains images obtained with a widefield microscope, while the bottom row contains images obtained with STORM imaging. In the widefield images, we cannot distinguish single emitters, while in the STORM images, we can.

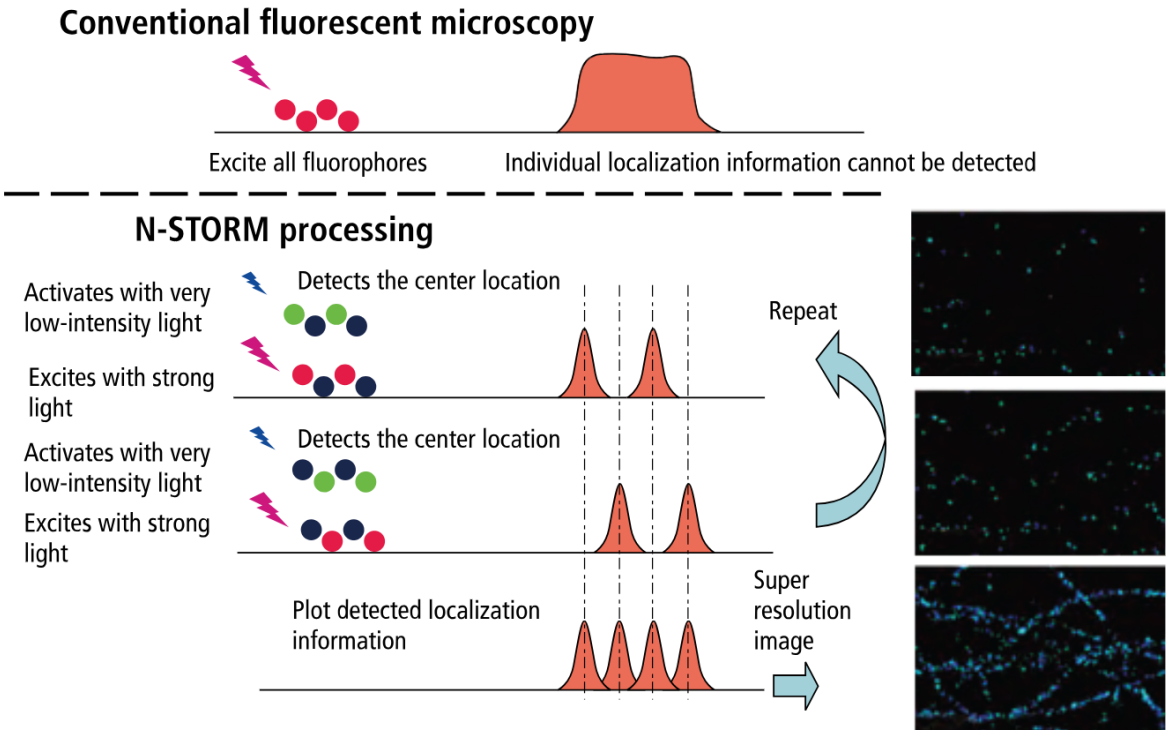


Figure 1.4: *Comparison between STORM imaging and conventional fluorescent microscopy [7]*

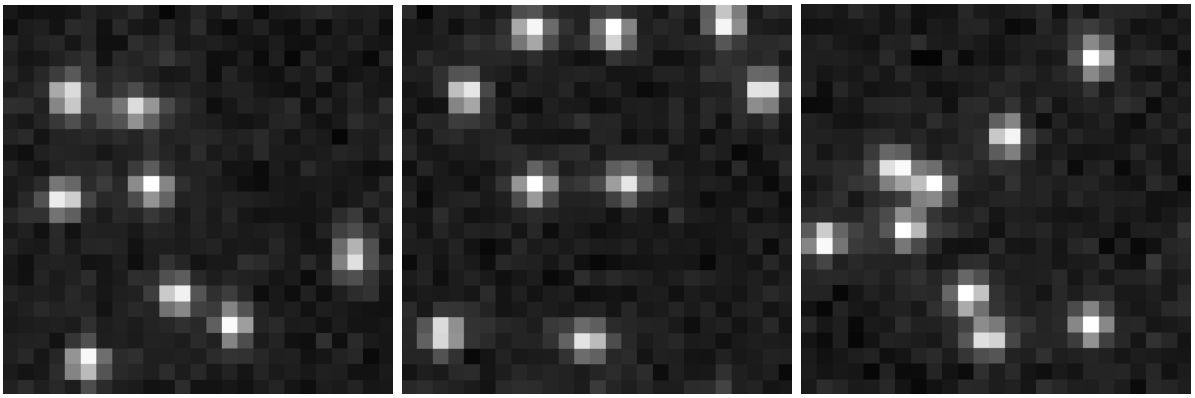


Figure 1.5: *Different STORM images*

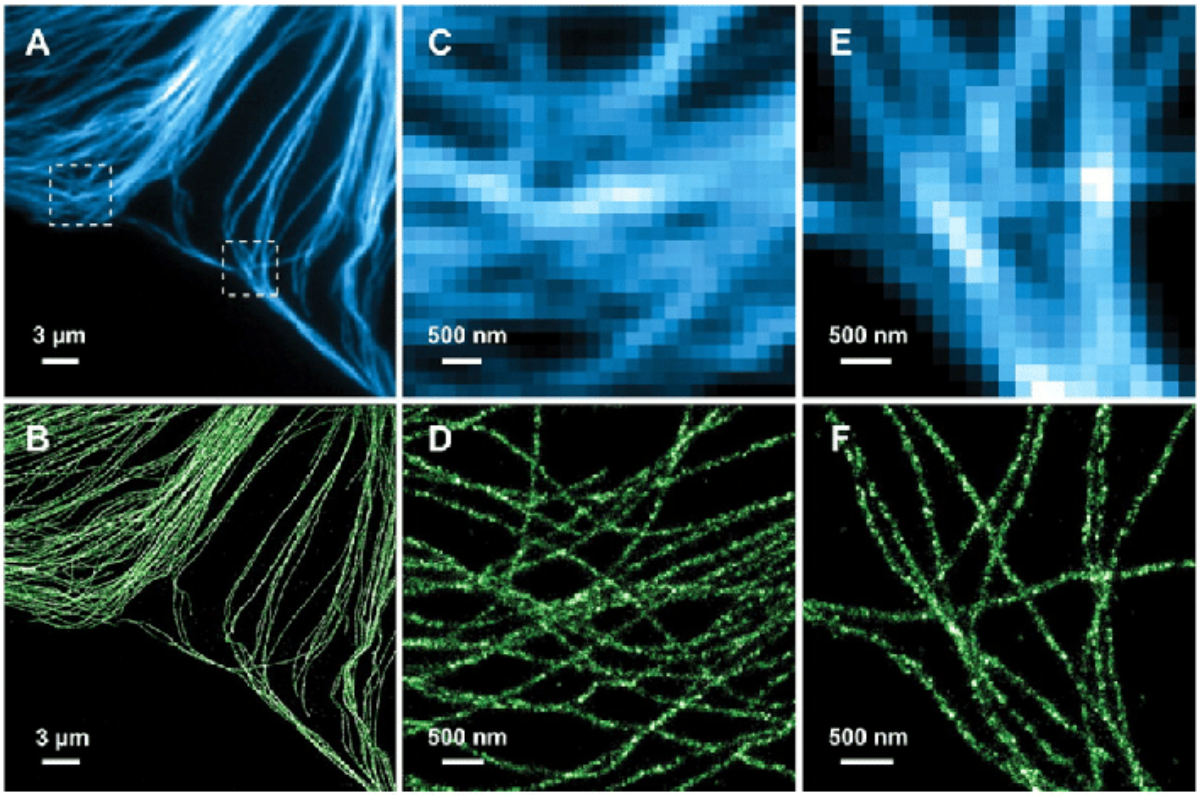


Figure 1.6: *Comparison between results obtained with STORM imaging and conventional microscopes [8]*

1.1.3 Oligopaint probes

The goal is to localize the emitters in images of chromatin, a mixture of DNA and proteins that forms the chromosomes in eukaryotic cells. In Fig. 1.7, we can see the chromatin packaged inside the nucleus. To obtain these images, one part of the image generation process involves the binding of chromatin and Oligopaint probes [9].

These probes are composed of computationally designed oligonucleotides (oligos), which are small sequences of nucleotides. Part of the probes comprises a short region of genomic homology that allows them to bind to specific parts of the chromosome. Genomic homology regions are specific sequences of nucleotides designed to bind to targeted genomic regions. How does this binding happen? The complementarity of nucleotide bases is used to bind to sections of the chromosome. These Oligopaint probes also have two arms at the ends of the genomic homology, called Mainstreet and Backstreet, which are nongenomic sequences 47 nucleotides long. These streets can be used for various purposes. Each Street consists of a universal primer, a region-specific barcode, and a 7-nucleotide sequence that, combined with the barcode, creates a binding site for a toehold oligo. Oligos called bridge oligos can bind to the streets, and these oligos are computationally designed to bind to fluorophores. The probes' barcode and bridge oligos are specific to each region of the chromosome that we want to analyze, allowing us to obtain images of each site independently. This is done to trace chromatin conformation, allowing the creation of a high-resolution 3D reconstruction of the chromatin. In Fig 1.8, we can see the structure of an Oligopaint probe.

After absorbing light at a specified wavelength, these fluorophores emit light, which is used to obtain images of the genomic regions. After the imaging round of the current genomic region is complete, toehold oligos are used to dislodge the bridge oligos of the current region. Then, the imaging round for the next region begins. These toehold oligos have a greater affinity for the site where the bridges bind to the probe than the bridges themselves. The bridges of the previous region are unbound, so that oligo bridges are bound only to the region currently being analyzed.

This method of obtaining genomic data is not always effective. In fact, the following can happen:

1. the fluorophores do not bind to anything, so we have unbound fluorophores floating in the nucleus;
2. the Oligopaint probes bind to the wrong parts of the chromosome or attach to other chromosomes;
3. the fluorophores bind to one another;

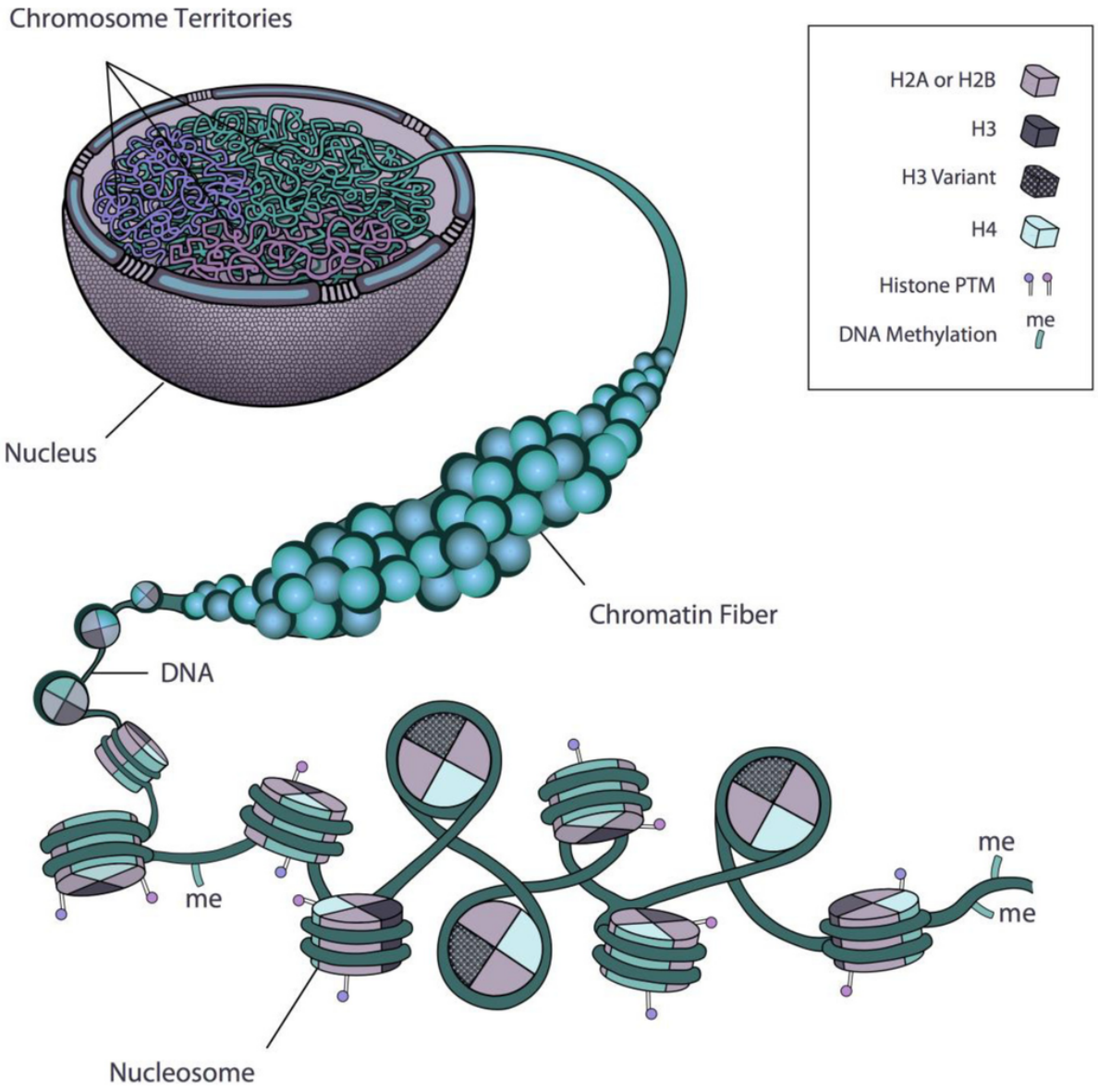


Figure 1.7: *Packaging of chromatin in the nucleus. Image from [10]*

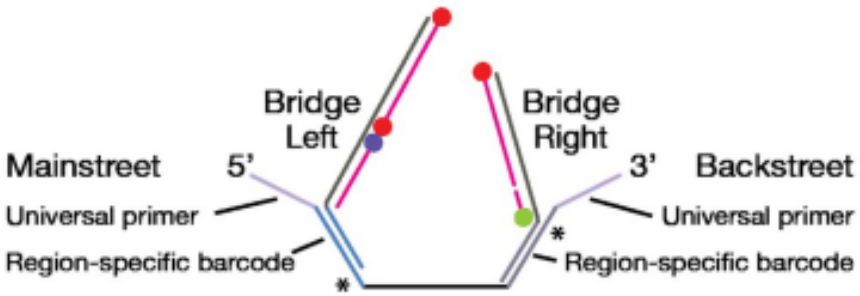


Figure 1.8: *Example of an Oligopaint probe. Image from [9]*

Thus, when fluorophores are activated, genomic data and noise are obtained.

1.2 Point spread function of the optical system

Having discussed how 2D images of the chromatin are obtained, we can now extend the discussion to how the 3D positions of the emitters in the frames can be recovered by modifying the optical system, yielding Point Spread Functions (PSFs) that encode the emitters' z-coordinates in the image. The PSF of an optical system is a mathematical function that represents how light emitted from a point spreads through the system, and it is correlated with the structure of the microscope.

For 2D and 3D SMLM, the data we acquire are movies of 2D images $\mathbf{Y}_{t_n} \in \mathbb{R}^{M_x \times M_y}$, with $t_n \in [0, T]$. Meanwhile, the samples analyzed are three-dimensional in both cases. However, for 2D SMLM, the microscopes used do not encode the emitters' z-positions in the image, so we can only recover the emitters' x- and y-coordinates. This is already a hard problem. For 3D SMLM, we can also recover the emitters' z-coordinates, which is even harder than estimating the x- and y-coordinates. Thus, for 2D SMLM, we reconstruct the emitters projected onto a 2D plane, whereas for 3D SMLM, we reconstruct the emitters in a 3D space.

1.2.1 Point Spread Function

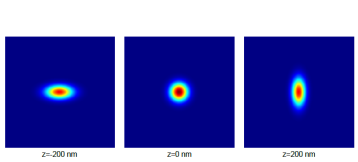
To make it possible to determine the z-coordinates for 3D SMLM, the following imaging techniques can be used:

- Astigmatic imaging [13] (Fig 1.9a), where an active fluorophore appears as an ellipse elongated vertically or horizontally, depending on the fluorophore's z-position relative to the focal plane of the optical system;

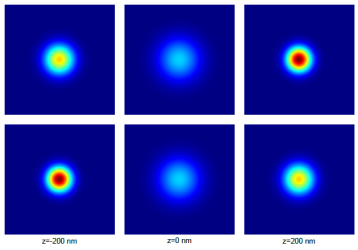
- Biplane imaging [14] (Fig 1.9b), in which light from a single active fluorophore is split into two paths to obtain two images. Thus, we also have two different focal planes. These focal planes are usually distant by a few hundred nm. The relative intensity difference of the fluorophore in the two images is used to determine its z-position relative to the focal plane of the optical system;

- Double-helix imaging [15] (Fig 1.9c), where an active fluorophore appears as two circles that rotate with the fluorophore's z-position relative to the focal plane of the optical system.

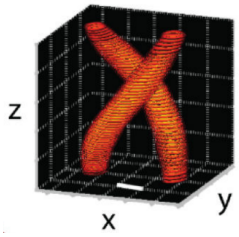
As we can see, all these methods consist of modifying the optical system to encode the z-coordinate of the active fluorophore in a 2D image. Thus, each method has particular PSFs.



(a) *Sections of astigmatic PSF [16]*



(b) *Sections of biplane PSFs [16]*



(c) *Double-helix PSF [17]*

Figure 1.9: *Examples of different PSFs*

Each method has pros and cons. For example, biplane and double imaging have a larger axial range than astigmatic imaging, which represents the z-distance over which the optical system encodes the z-position with acceptable precision. However, these two methods split the photon stream into two parts, resulting in lower image intensities, whereas this does not apply in the astigmatic case. The technique we are interested in is astigmatic imaging. For astigmatic imaging, a cylindrical lens is added to the optical system. If we know the relationship between the PSF distortions and the z-coordinates, we can infer the z-coordinates of the molecules based on the images we capture. In the center of Fig 1.10, we see a microscope with an astigmatic PSF, while in the right panel of Fig 1.10, we observe how emitters with different z-positions appear in an image.

1.2.2 Approximation of the PSF of the optical system

Having discussed how optical systems can be modified to obtain a PSF with properties that allow for 3D SMLM, we can now describe how the system PSF can be experimentally measured. For the PSF measurement, a few fluorescent beads are needed. These beads continuously emit light and are positioned to have the same z-position. Furthermore, they can also be moved along the z-direction.

Now, multiple images of these beads are computed at different z-positions. The z-distance traveled by the beads between images is constant and is called the z-step. Ultimately, we collect a sequence of images. During this process, the beads drift along the x and y directions. To handle this problem, drift-correction algorithms have been developed. In Fig. 1.11, we show different images of the fluorescent beads.

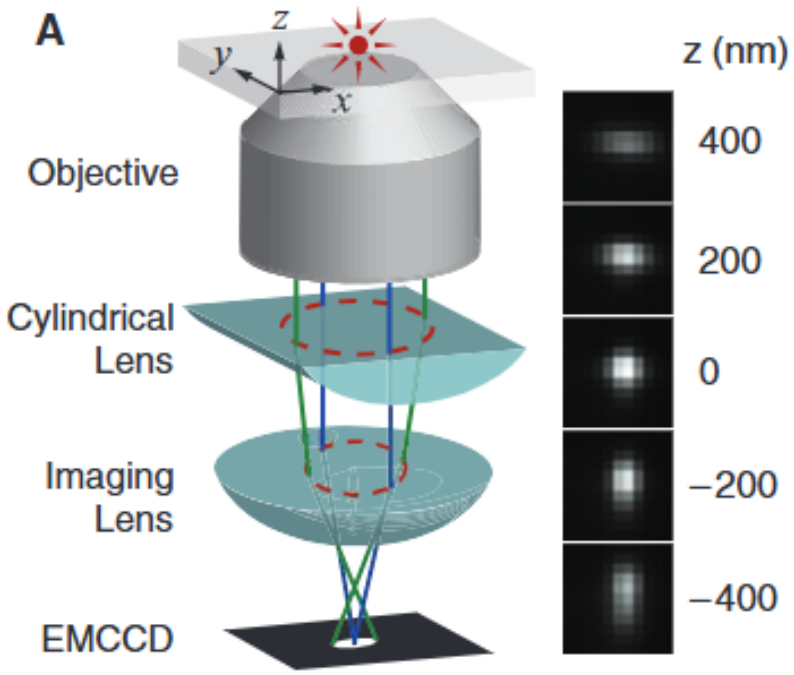


Figure 1.10: *Microscope with cylindrical lens. Image from [13]*

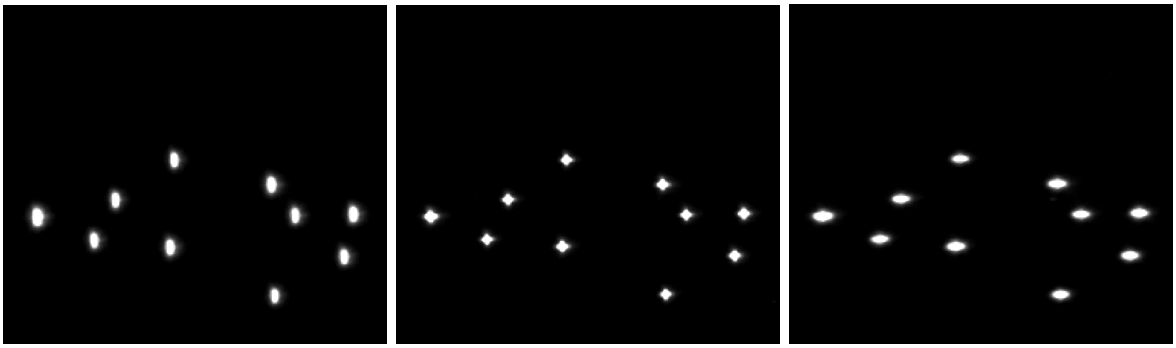


Figure 1.11: *Different images of the same set of beads at different z-positions*

Processing the sequence of images of the fluorescent beads, we can approximate the PSF of the optical system. This approximation is usually required for localization algorithms. There are two main approaches to approximating the PSF of an astigmatic imaging system:

- Gaussian fitting [18];
- Cubic spline fitting [19].

With Gaussian fitting, we assume that the PSF has the following form:

$$H(x,y,z) = \frac{1}{2\pi\sigma_x(z)\sigma_y(z)} e^{-\frac{(x-x_p)^2}{2\sigma_x^2(z)} - \frac{(y-y_p)^2}{2\sigma_y^2(z)}},$$

where $\sigma_x(z)$ and $\sigma_y(z)$ are functions that approximate the relationships between the PSF distortions and the z-coordinates of the molecules. Using the movie of the fluorescent beads, calibration curves are computed. In Fig 1.12, we observe an example of calibration curves. From these curves, we can determine $\sigma_x(z)$ and $\sigma_y(z)$.

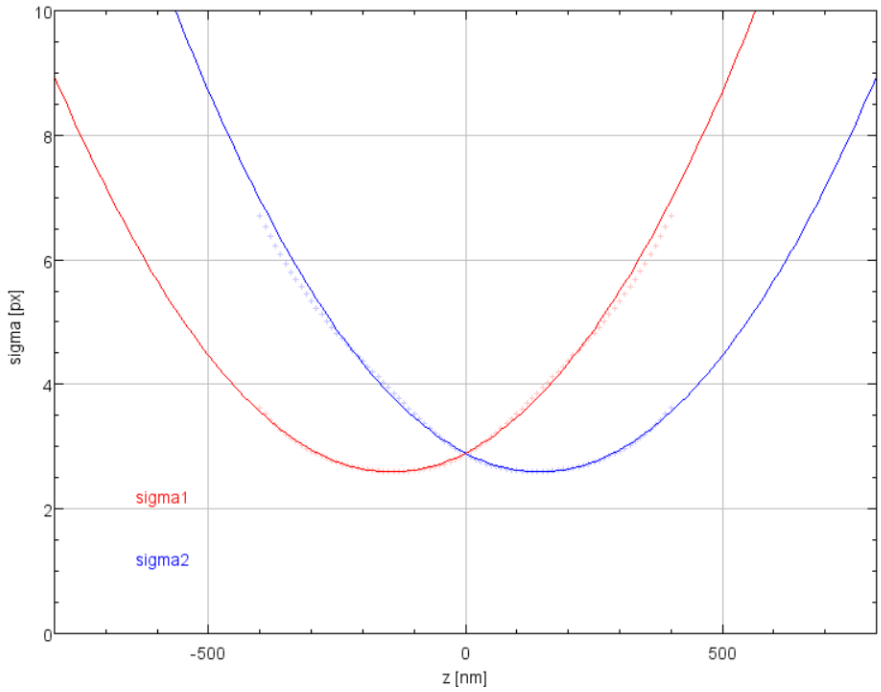


Figure 1.12: *Example of calibration curves*

With cubic spline fitting, we attempt to approximate the PSF using cubic splines. For the 3D case, the approximated PSF in a voxel is a trivariate cubic polynomial of the form:

$$f_{i,j,k}(x,y,z) = \sum_{m=0}^{3}\sum_{n=0}^{3}\sum_{o=0}^{3} a_{i,j,k,m,n,o} \left(\frac{x-t_i}{\Delta t}\right)^m \left(\frac{y-t_j}{\Delta t}\right)^n \left(\frac{z-u_k}{\Delta u}\right)^o,$$

with $t_i \leq x \leq t_{i+1}$, $t_j \leq y \leq t_{j+1}$, $u_k \leq z \leq u_{k+1}$, and $[t_i, t_{i+1}] \times [t_j, t_{j+1}]$ being the dimensions of a pixel, and $[t_i, t_{i+1}] \times [t_j, t_{j+1}] \times [u_k, u_{k+1}]$ being the dimensions of a voxel, which is the 3D version of a pixel.

The sequence of bead images is not sufficient to accurately approximate the PSF. Therefore, a cubic spline interpolation is used to obtain enough PSF measurements to compute the coefficients $a_{i,j,k,m,n,o}$. Since we have 64 coefficients $a_{i,j,k,m,n,o}$ per voxel,

we need 64 measurements per voxel. To obtain the coefficients $a_{i,j,k,m,n,o}$, we solve the interpolation problem:

$$f_{i,j,k}(x_m, y_n, z_o) = m_{i,j,k,m,n,o}, \qquad m, n, o = 0, \cdots, 3,$$

where $m_{i,j,k,m,n,o}$ are the interpolated measurements and (x_m, y_n, z_o) are the coordinates corresponding to the measurements. Thus, we obtain the PSF approximation for a certain voxel. This process is repeated for all voxels that make up the domain of our problem.

Compared with Gaussian fitting, this method yields better results. However, it is computationally more expensive.

1.2.3 Z-stacks

Astigmatic imaging works well for samples within a certain z-range. However, sometimes the objects are much taller than this range. In such cases, what can be done?

One option is to partition the domain containing the object of interest into sections and analyze each section independently. In particular, the volume can be divided into horizontal sections, called z-sections, such that each section has a z-range smaller than the range in which our PSF is well-defined. Then, images are obtained one section at a time. These sections overlap in certain parts to avoid leaving gaps in the sample analysis. If we capture images of each section, we end up with a movie for each section. All these movies together form a z-stack. The term 'z-stack' can also be used when discussing a sequence of fluorescent beads. In Fig. 1.14, we show how an emitter far from the focal plane appears. As we can see, inferring the emitter's coordinates is very difficult. In Fig. 1.13, we show a cell divided into different sections.

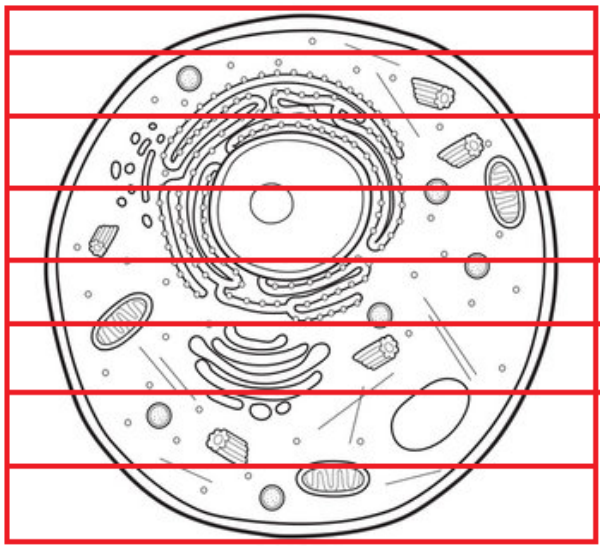


Figure 1.13: *Example of selected z-sections*

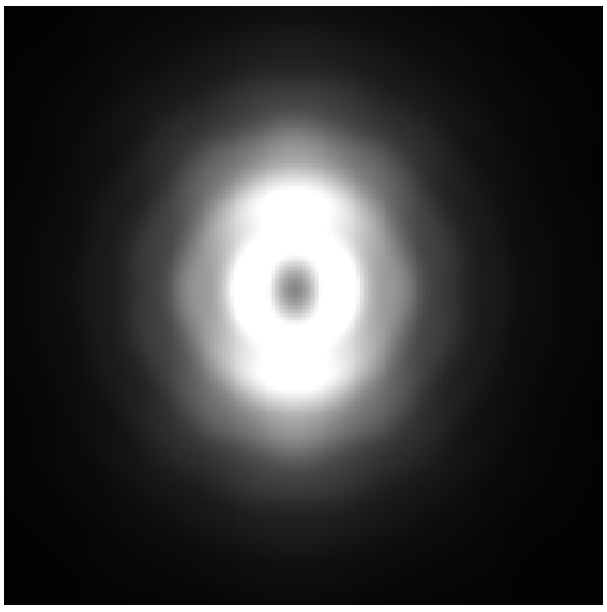


Figure 1.14: *PSF of an emitter too distant from the focal plane*

Chapter 2

Forward mathematical model for SMLM

In the previous chapter, we described the acquisition process for single-molecule localization microscopy in the 3D case. In this chapter, we will formulate a mathematical model for the image acquisition process of STORM images. In particular, we will describe how an image $\mathbf{Y}_{t_n}$ is acquired. The index t_n represents the discrete time at which the image was acquired, where $t_n \in [0, T]$.

2.1 Notations

From now on, we will use the following notations:

- $\mathbb{R}_{++} = (0, +\infty)$;
- $a, b \in \mathbb{R}$;
- $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n, n \in \mathbb{N}$ are vectors, and $a_i, b_i \in \mathbb{R}$ are their respective i-th elements.
- $\mathbf{A}, \mathbf{B}$ can be tensors or matrices. If $\mathbf{A} \in \mathbb{R}^{N_x \times N_y}$ is a matrix, $\mathbf{a}_i \in \mathbb{R}^{N_X}$ is the respective i-th column. If $\mathbf{H} \in \mathbb{R}^{N_x \times N_y \times N_z}$ is a tensor, $\mathbf{H}^i = \mathbf{H}(:, :, i) \in \mathbb{R}^{N_x \times N_y}$ is the i-th horizontal slice;
- Let $\mathbf{A}$ be a tensor or a matrix. $vec(\mathbf{A}) \in \mathbb{R}^n$ is $\mathbf{A}$ reshaped to be a vector, with n being the number of elements of $\mathbf{A}$;
- Given a tensor $\mathbf{H}$, $\mathbf{H}^T$ is a tensor such that $(\mathbf{H}^i)^T = (\mathbf{H}^T)^i$;

2.2 Image generation process

The image generation process can be divided into four steps, which are:

1. Volume domain discretization;
2. Blurring operator;
3. Downsampling operator;
4. Noise and background perturbation.

In the following, we will describe each of these steps.

2.2.1 Volume domain discretization $\mathbf{X}_{t_n}$

We start with a list of coordinates of emitters active at a time t_n, which exist inside a bounding box $\mathcal{B}$ in $\mathbb{R}^3$, $\mathcal{B} = [0, C] \times [0, W] \times [0, H]$. In a discrete setting, we must discretize the bounding box and approximate the emitters' coordinates. We discretize them using the z-step used for fluorescent beads and the pixel size we want for our final images, divided by the downsampling factor L used later. Let px be the width of a pixel in nanometers divided by L, py be the height of a pixel in nanometers divided by L, and zs be the z-step. Our discretized bounding box will be $[0, N_z] \times [0, N_x] \times [0, N_y]$, with:

$$\begin{aligned} N_z &= \frac{C}{zs}, \\ N_x &= \frac{W}{px \cdot L}, \\ N_y &= \frac{H}{py \cdot L}. \end{aligned}$$

Now, let (x_c, y_c, z_c) be the continuous coordinates of the emitters, the corresponding discretized coordinates (x_d, y_d, z_d) are:

$$\begin{aligned} x_d &= round\left(\frac{x_c}{px} - 0.5\right), \\ y_d &= round\left(\frac{y_c}{py} - 0.5\right), \\ z_d &= round\left(\frac{z_c}{zs}\right), \end{aligned}$$

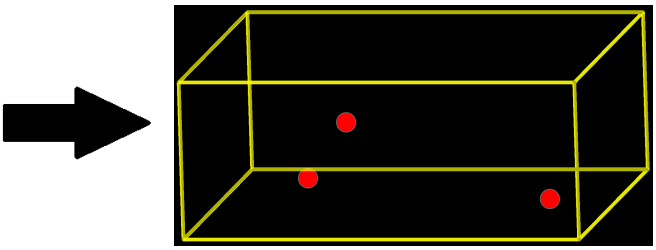
where $round(\cdot)$ rounds a value to its nearest integer. Due to later problems related to downsampling operations, we add -0.5 to our discretization formulas.

Assuming that an activated fluorophore emits 10000 photons, let $\mathbf{X}_{t_n} \in \mathbb{R}^{N_z \times N_x \times N_y}$. Then, $X_{t_n}(z_d, x_d, y_d) = 10000$ if (z_d, x_d, y_d) are the discretized coordinates of an emitter.

Otherwise, $\mathbf{X}_{t_n}(z_d, x_d, y_d) = 0$. From now on, we can also assume that $N_x = N_y = N$. Thus, we have $\mathbf{X}_{t_n} \in \mathbb{R}^{N_z \times N \times N}$. In Fig. 2.1, we can see the results of applying this discretization operation to three entries of a list of coordinates. For this figure, we used $N = 104$, $N_z = 41$, $px = py = 20$, $zs = 10$.

1135.291307628886	937.5374053597208	357.19166067720926
762.5595707292123	1068.335972968276	370.1892506731356
853.6359745541922	1372.7349962380522	380.55922406541276
745.8304878284507	1449.3758431324768	358.84023275360505
843.8270878297772	1316.2125487392968	352.8217902205879
762.1945288517345	1325.800489043986	382.8885127899848
873.5986969443366	1510.9342361898616	380.43321556130786
1006.4319138747304	1536.6374455014757	351.1043245349533
843.5483402603058	1427.8183008893593	362.7859137779883
1021.8493784265738	1572.896319852569	263.39625647271623
748.5322914789795	1510.1114209215102	363.1428779798116
1090.020721322122	1501.4471289101912	384.0334836051875
970.550670072731	1395.63119792189	396.92737817310353
1000.7619403234276	1538.7381815844728	262.8708397587229
976.6952536888966	1384.3131714898255	399.4943328355821
872.8226203369777	1329.1624292687757	369.87144668922565
924.5328939883292	1510.2085283508586	364.6388369930669
808.8701518050475	1367.1586475959066	266.6886011650781
1182.585436988997	1459.377776642652	297.60125129589414
1088.8702682625662	1241.3608455073645	223.84898358939336
1123.707389923682	1366.4336209721896	308.9994226966828

(a) *List of coordinates*



(b) $\boldsymbol{X}_{t_n}$

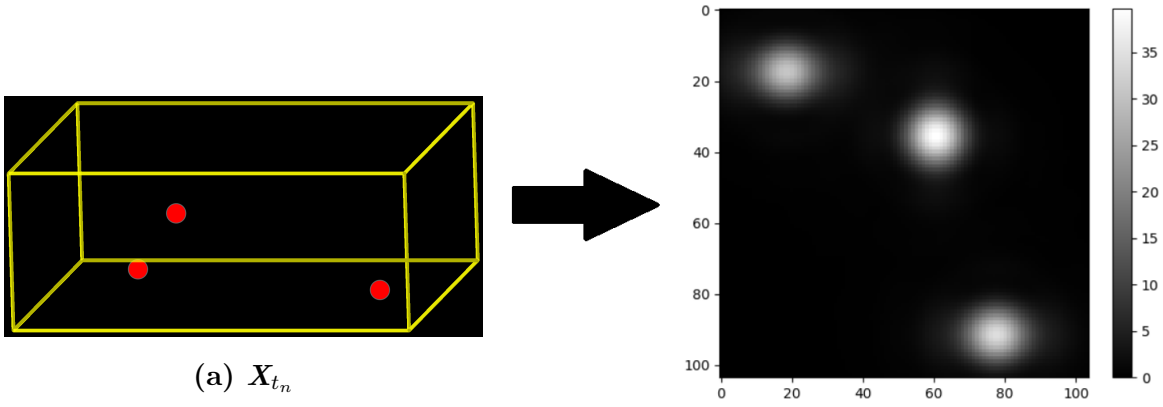
Figure 2.1: *Volume domain discretization*

2.2.2 Blurring operator H

In Chapter 1, we discussed approximating the optical system PSF. In particular, the approximations we discussed are continuous. We transform the continuous approximation into a discrete one using the z-step and the pixel size divided by L. Let $\mathbf{H} \in \mathbb{R}^{N_z \times N \times N}$ be the discrete PSF of the optical system. To obtain a 2D image from $\mathbf{X}_{t_n}$, we convolve each horizontal slice of $\mathbf{X}_{t_n}$, denoted by $\mathbf{X}^i_{t_n}$, with the corresponding horizontal slice of $\mathbf{H}$, represented by $\mathbf{H}^i$. We do this because the 2D PSF of an emitter varies with its z-coordinate. Then, the blurred 2D images are derived over z as follows:

$$\mathbf{D} = \sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i_{t_n},$$

with $\mathbf{D} \in \mathbb{R}^{N \times N}$. With this step, we aim to simulate the blurred acquisitions of emitters using a microscope. In Fig. 2.2, we can see the image $\mathbf{D}$.



(a) X_{t_n} (b) D

Figure 2.2: *Blurring operation*

2.2.3 Eliminating Fourier-related artifacts

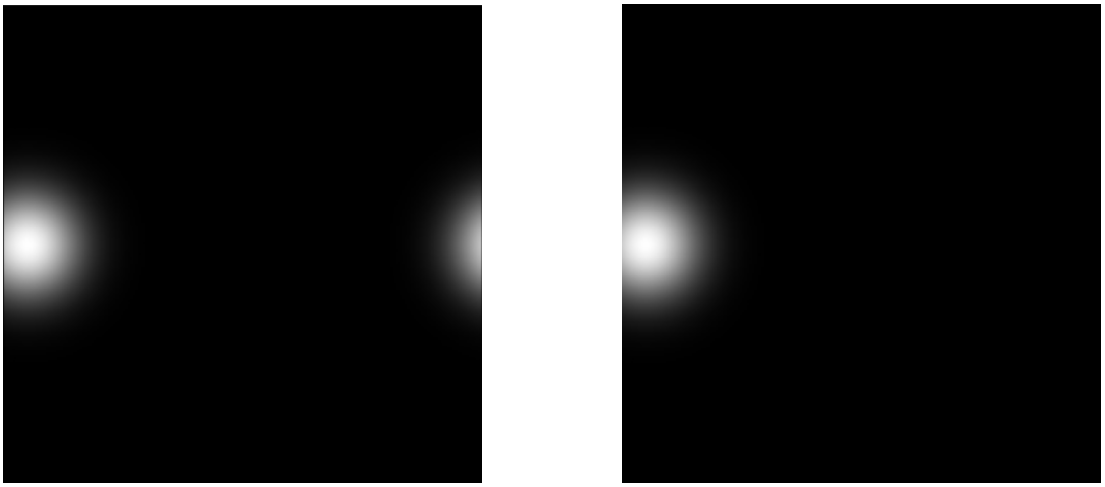
As mentioned above, to accelerate convolution computations, we use the Fast Fourier Transforms (FFT) of the matrices involved. However, using the FFT has a drawback.

To use the FFT, we need to assume periodic boundary conditions for our convolutions. If our activated fluorophores are near the center of the image, there are no issues. If our emitters are near the image border, we can observe light on the other side of the image, as if a fluorophore were present there. We can see an example in Fig 2.3a. This does not happen in real frames. Thus, if we want to simulate real frames, we need a way to remove this artifact. How do we do this? There are two options: zero-padding and apodization [20]:

- zero-padding consists of extending the borders of the image and the PSF with a layer of zeros. After the convolution, we discard the added layer. Thus, if the layer is large enough, there are no artifacts. However, the computational costs increase with the size of the padding layer.

- apodization consists of applying a function to the image that sets the signal to zero near the borders of the image. With apodization, we have lower computational costs than zero-padding. However, we end up with lower data intensity, and apodization doesn't completely eliminate the artifact.

As we can see, each option has pros and cons. For our data, we will use zero-padding. In Fig 2.3b, we can see that the Fourier artifact has been removed.



(a) **(b)**

Figure 2.3: *Elimination of the Fourier artifact: (a) contains the artifact; the artifact is removed in (b)*

2.2.4 Downsampling operator S

After blurring, we apply a downsampling operator S on the image $\mathbf{D}$. Let L be the downsampling factor, we define the downsampling operator as the function $S : \mathbb{R}^{N\times N} \longrightarrow \mathbb{R}^{N/L\times N/L}$. Let $\mathbf{D} \in \mathbb{R}^{N\times N}$, then $S(\mathbf{D})$ is a matrix of size $N/L \times N/L$ whose elements are the sums of the elements in blocks of $\mathbf{D}$ with a dimension of $L \times L$. In particular, we have:

$$S(\mathbf{D}) = \mathbf{M}\mathbf{D}\mathbf{M}^T,$$

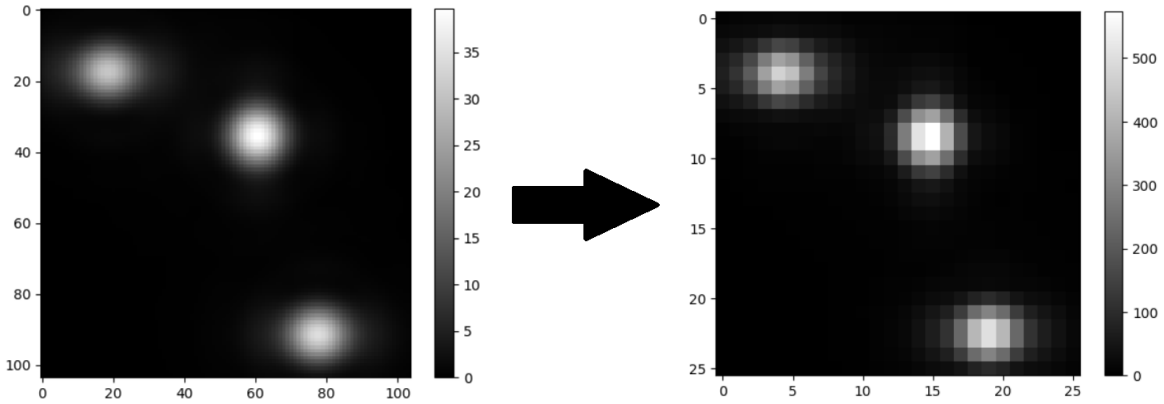
with M a matrix of blocks of ones and zeros. Assuming, for example, we have $L = 4$ and $N = 12$, then a matrix $\mathbf{D} \in \mathbb{R}^{12\times 12}$ is downsampled into $S(\mathbf{D}) \in \mathbb{R}^{3\times 3}$ by the following matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 12}.$$

Likewise, we can define the operator $S^T : \mathbb{R}^{N/L\times N/L} \longrightarrow \mathbb{R}^{N\times N}$ as follows:

$$S^T(\mathbf{Y}) = \mathbf{M}^T\mathbf{Y}\mathbf{M}.$$

In contrast to S, S^T increases the dimensions of the image Y. Using downsampling, we aim to represent the transition from a high-resolution setting to a low-resolution, discrete setting for the emitters. In Fig. 2.4, we can see the result of applying S to $\mathbf{D}$. For this figure, we used a downsampling factor of $L = 4$.

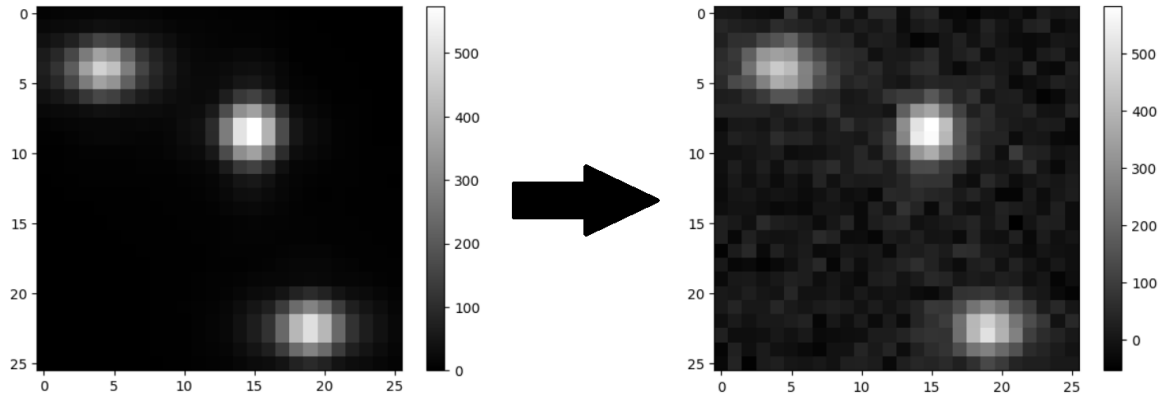


(a) $\boldsymbol{D}$ (b) $S(\boldsymbol{D})$

Figure 2.4: *Downsampling operation*

2.2.5 Noise and background perturbation

To simulate the electronic noise present in real images, we add noise to the downsampled images. Since we assume each active fluorophore emits 10000 photons, we do not work in a low-photon situation. Thus, we approximate the Poisson noise with Gaussian noise. Electronic noise has a zero mean; regarding the variance of the noise to use, we attempt to estimate the variance of the noise present in real images acquired using the imaging system we are trying to simulate and use it for our simulated images. In Fig. 2.5, we can see the addition of Gaussian noise to $S(\mathbf{D})$.



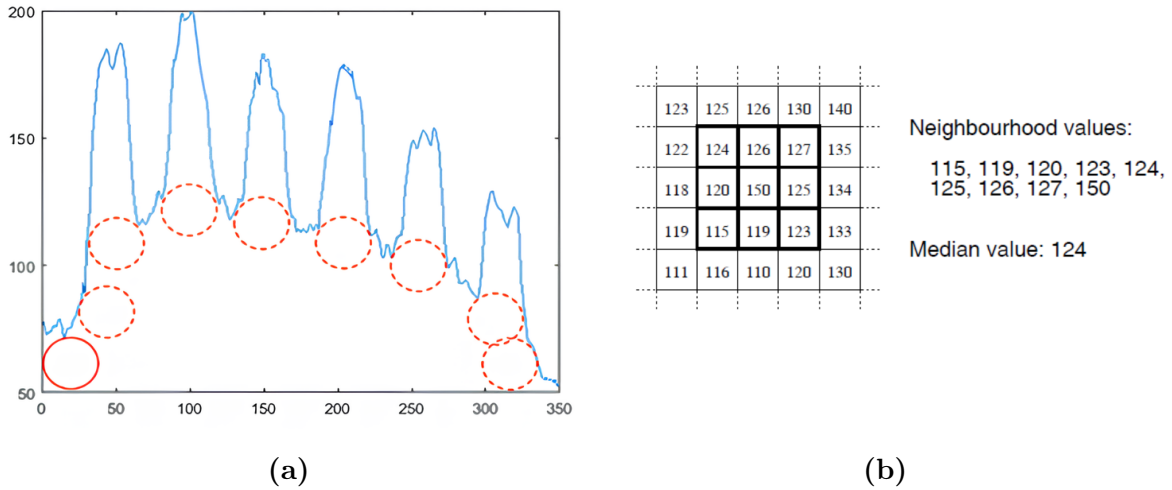
(a) $S(\boldsymbol{D})$ (b) $S(\boldsymbol{D}) + \boldsymbol{N}_{t_n}$

Figure 2.5: *Noise perturbation*

Afterward, we also add a constant uniform background to our noisy image. The background represents all the light captured by the optical system that was not generated by the currently active fluorophores. To estimate the background of an image, we consider two methods:

- The rolling ball method [21], which treats the image as a 3D surface and rolls a ball underneath it. The contact points between the ball and the surface form the estimated background. The ball radius influences the result; it must be large enough to avoid the peaks;
- The median filter [22], which computes the background for an image pixel as the median over a given neighborhood of the pixel. The median is the value falling halfway in a sorted set. Thus, peaks are ignored.

Both methods have pros and cons. The median filter creates more uniform backgrounds. However, the image with the estimated background removed may contain pixels with negative intensity. The rolling ball method does not have this issue, but the backgrounds are less uniform. In Fig 2.6 and Fig 2.7, these methods and their application results are shown. On the left of Fig 2.7 is the original image; in the center is the background estimated using a median filter of size 30×30 pixels; on the right is the background estimated using the rolling ball method with a radius of 30 pixels. In Fig. 2.8, we can see the addition of the background to $S(\mathbf{D}) + \mathbf{N}_{t_n}$. In Fig. 2.8, we add a Gaussian-shaped background rather than a uniform one so that the background is distinguishable. In the simulated images of the datasets presented in Chapter 4, the background is constant and uniform.



(a) **(b)**

Figure 2.6: *Example of rolling ball method (a) and median filter (b)*

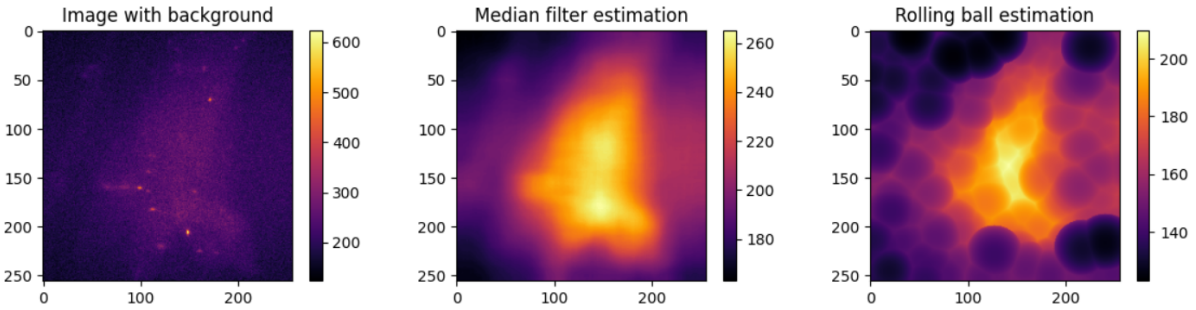
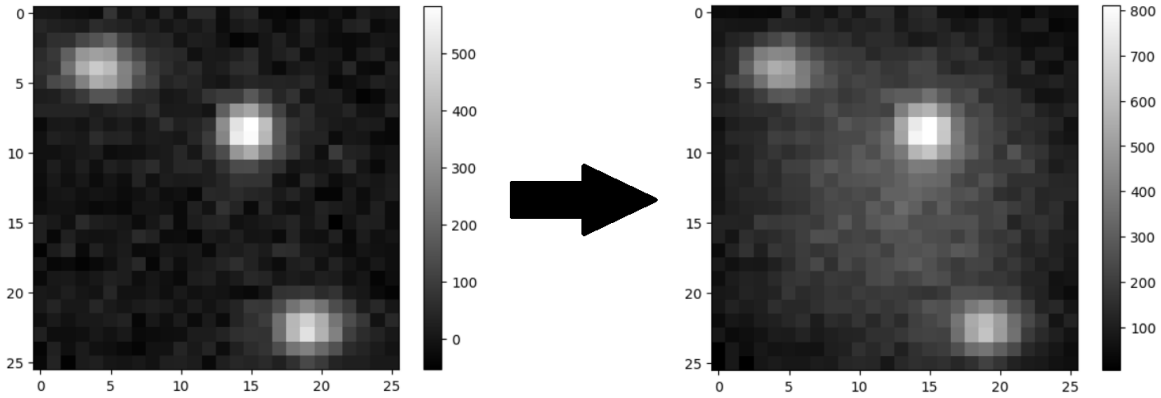


Figure 2.7: *Background estimation*



(a) $S(\boldsymbol{D}) + \boldsymbol{N}_{t_n}$ **(b)** $S(\boldsymbol{D}) + \boldsymbol{N}_{t_n} + \boldsymbol{B}_{t_n}$

Figure 2.8: *Background perturbation*

Considering the entire described acquisition process, the **forward acquisition model** for the image generation process reads as follows:

$$\mathbf{Y}_{t_n} = S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i_{t_n}\right) + \mathbf{N}_{t_n} + \mathbf{B}_{t_n}, \tag{2.1}$$

where $\mathbf{H}, \mathbf{X}_{t_n} \in \mathbb{R}^{N_z \times N \times N}$, S is the downsampling operator with a factor of L, $\mathbf{Y}_{t_n} \in \mathbb{R}^{N/L \times N/L}$ is the image we obtain at time $t_n \in [0, T]$ from $\mathbf{X}_{t_n}$, $\mathbf{N}_{t_n} \in \mathbb{R}^{N/L \times N/L}$, $\mathbf{N}_{t_n}(i,j) \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian noise, and $\mathbf{B}_{t_n} \in \mathbb{R}^{N/L \times N/L}$ is the background. Usually, one STORM sequence contains thousands of images, so we have $T > 1000$. In Fig. 2.9, we show the image generation process. In Fig. 2.10, we show a real STORM image. In Fig. 2.11, we present three patches from it, while in Fig. 2.12, we present three simulated STORM images created using the model (2.1) with a downsampling factor of 8, $N = 104$, and $px = 20$. The real images have a pixel size of 162.5 nm, while the simulated ones have a pixel size of 160 nm. However, the real image captures a region of size 41600 nm $\times$ 41600 nm, while the simulated ones capture a region of size 2080 nm $\times$ 2080 nm.

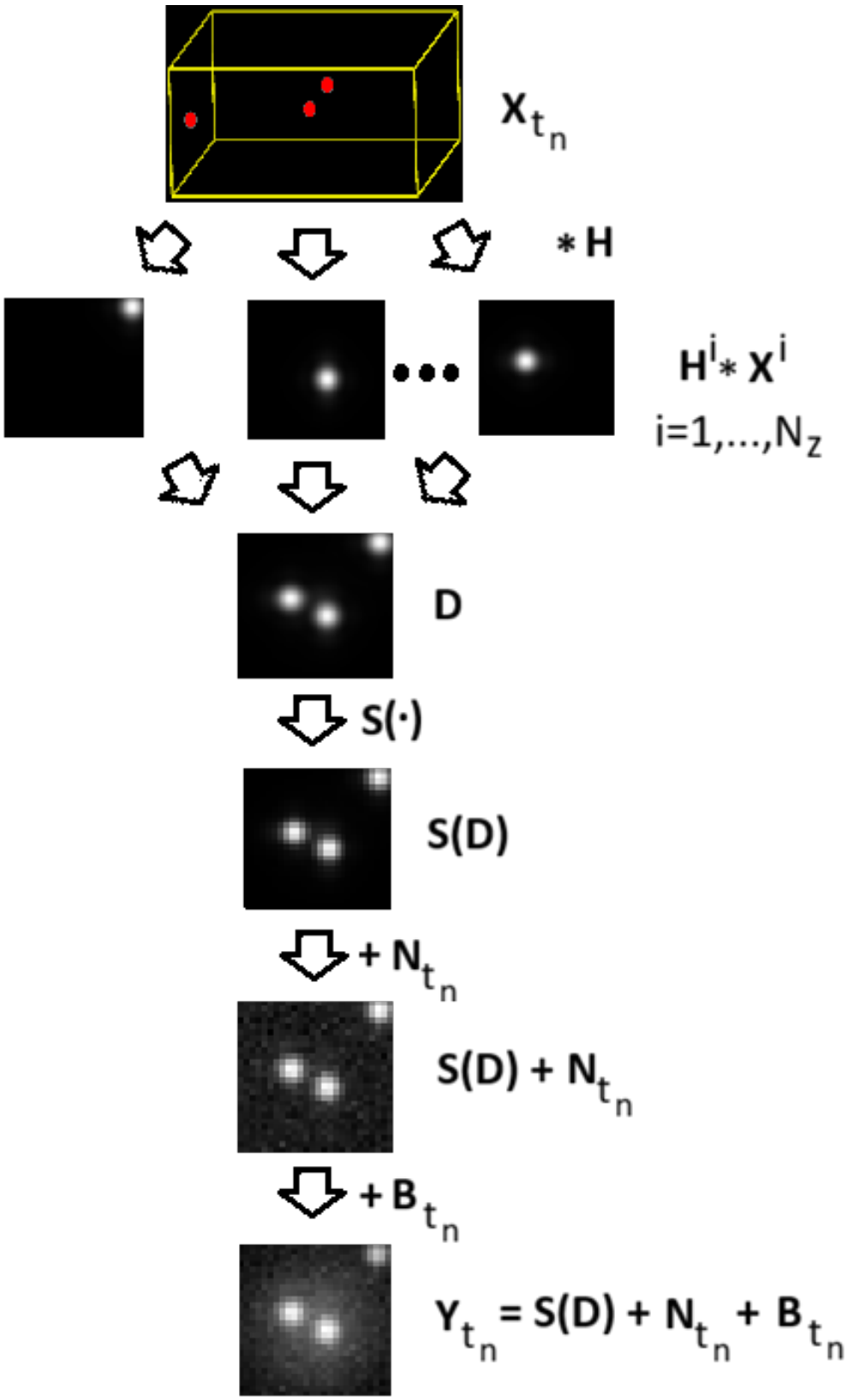


Figure 2.9: *Image degradation model from* $\boldsymbol{X}_{t_n}$ *to* $\boldsymbol{Y}_{t_n}$

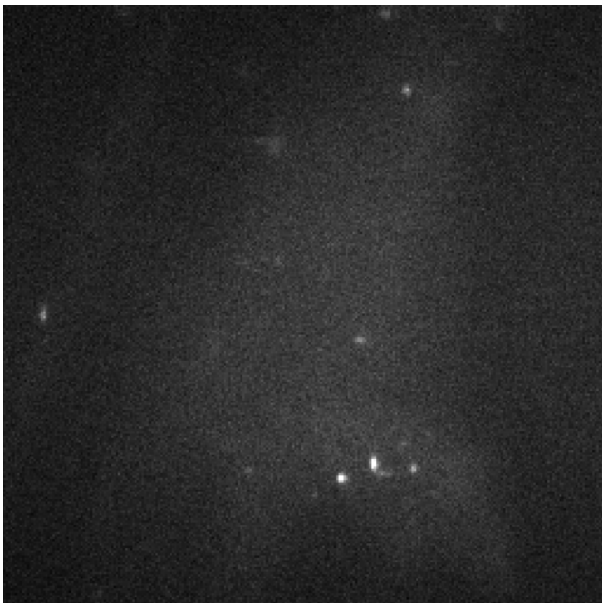


Figure 2.10: *Real STORM image* $\boldsymbol{Y}_{t_n}$

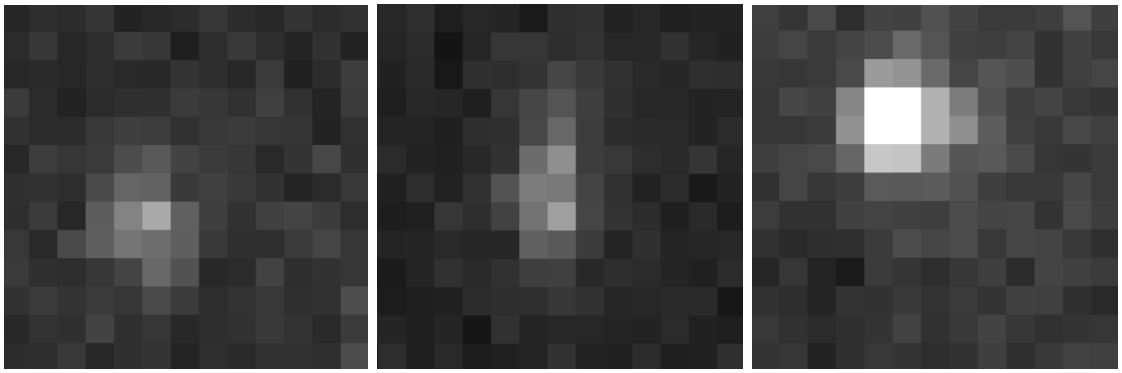


Figure 2.11: *Patches from Fig. 2.10*

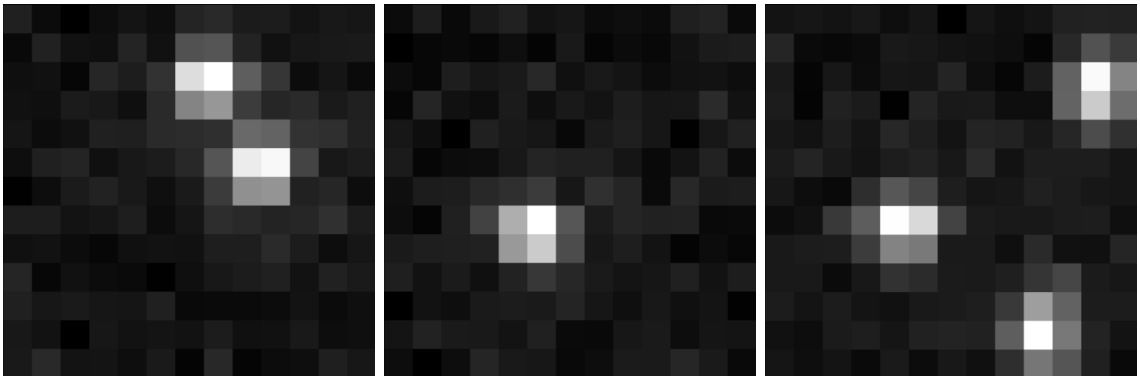


Figure 2.12: *Simulated STORM images* $\boldsymbol{Y}_{t_n}$

Chapter 3

Inverse mathematical model for SMLM

3.1 Preliminary definitions and theorems

Definition 3.1. *Let $n \in \mathbb{N}$, $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$. f is a convex function if and only if:*

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}), \qquad \forall t \in [0,1],\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

f is a proper function if and only if:

$$\exists \boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \neq +\infty.$$

f is a coercive function if and only if:

$$\lim_{\|x\|_2 \to +\infty} f(\boldsymbol{x}) = +\infty, \qquad \boldsymbol{x} \in \mathbb{R}^n.$$

f is a lower semi-continuous function (l.s.c) if and only if:

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x}_k \to \boldsymbol{x} \ \Rightarrow\ f(\boldsymbol{x}) \leq \lim_{k \to +\infty} \inf f(\boldsymbol{x}_k).$$

Definition 3.2. *Let $n \in \mathbb{N}, \boldsymbol{x} \in \mathbb{R}^n$. We define the ℓ_2 norm of $\boldsymbol{x}$ as:*

$$\|\boldsymbol{x}\|_2 := \sqrt{\sum_{i=1}^{n} \boldsymbol{x}_i^2}.$$

Let $\boldsymbol{X} \in \mathbb{R}^{N_z \times N_x \times N_y}$, with $N_z, N_x, N_y \in \mathbb{N}$. We denote:

$$\|\boldsymbol{X}\|_F = \|\boldsymbol{x}\|_2,$$

where $\boldsymbol{x} = vec(\boldsymbol{X})$.

Definition 3.3. *Let $n \in \mathbb{N}, f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ be a proper, differentiable function. We say that f is a gradient-Lipschitz continuous function with constant L_f if and only if:*

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \quad \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

L_f is called the Lipschitz constant.

Theorem 3.4 (Existence of global minimizers)**.** *Let $n \in \mathbb{N}, f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ be a proper, l.s.c. and coercive function. f admits a global minimizer.*

Theorem 3.5 (Cauchy-Schwarz inequality)**.** *Let $n \in \mathbb{N}, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Then:*

$$|\boldsymbol{x} \cdot \boldsymbol{y}| \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2. \tag{3.1}$$

3.2 Inverse mathematical model

Given the acquisition model (2.1), we aim to approximate the reconstruction $(\mathbf{X}_{t_n})^*$ at time $t_n \in [0, T]$ by solving the following optimization problem:

$$(\mathbf{X}_{t_n})^* \in J(\mathbf{X}) = \arg \min_{\mathbf{X} \in \mathbb{R}^{N_z \times N \times N}} f(\mathbf{X}) + \lambda R(\mathbf{X}), \tag{3.2}$$

where $f(\mathbf{X})$ is called the fidelity term, $R(\mathbf{X})$ is called the regularization term, and $\lambda \in \mathbb{R}_{++}$ is called the regularization parameter. The fidelity term measures how well $\mathbf{Y}_{t_n}$ is predicted by $\mathbf{X}$, while the regularization term measures the regularity of the solution. With the regularization term, we aim to suppress noise in the solution, while the fidelity term aims to obtain a solution sufficiently close to $\mathbf{X}_{t_n}$. The parameter λ serves as a balancing term between them. If it is too small, the solution will contain too much noise. If it is too large, the result is a poor approximation of $\mathbf{X}_{t_n}$. Thus, the λ we choose influences the result we obtain by solving the problem.

The choice of the fidelity and regularization terms depends on prior knowledge of the problem. Since we assume our noise distribution is Gaussian, we use the l_2 norm as the fidelity term, and $\mathbf{B}_{t_n}$ due to the background present in our images. Thus, we have:

$$f(\mathbf{X}) := \frac{1}{2} \left\| \mathbf{Y}_{t_n} - S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i \right) - \mathbf{B}_{t_n} \right\|_F^2. \tag{3.3}$$

Furthermore, since the tensor $\mathbf{X}_{t_n}$ is very sparse, we want regularizer functions that promote sparsity. Thus, we could use the ℓ_0 norm as the regularizer function.

Definition 3.6. *Let $\boldsymbol{x} \in \mathbb{R}^n$. We define the ℓ_0 norm of $\boldsymbol{x}$ as:*

$$\|\boldsymbol{x}\|_0 := \#\{i : \boldsymbol{x}_i \neq 0\}.$$

If we have $\boldsymbol{X} \in \mathbb{R}^{N_z \times N_x \times N_y}$*, with* $N_z, N_x, N_y \in \mathbb{N}$*, we define:*

$$\|\boldsymbol{X}\|_0 := \|\boldsymbol{x}\|_0,$$

where $\boldsymbol{x} = vec(\boldsymbol{X})$*.*

The ℓ_0 norm is not really a norm since it doesn't satisfy the property: $\lambda\|\mathbf{x}\|_0 = \|\lambda\mathbf{x}\|_0$ $\forall \mathbf{x} \in \mathbb{R}^N$, $\forall \lambda \in \mathbb{R}_{++}$. However, for simplicity, we will continue to refer to it as a norm.

Now, if we use the ℓ_0 norm, the problem (3.2) will be NP-hard. To avoid this, we can use regularizer functions other than the ℓ_0 norm that promote sparsity, such as:

- ℓ_1 norm, which is defined as $\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |\mathbf{x}_i|$ for $\mathbf{x} \in \mathbb{R}^n$;
- CEL0 penalty [23];
- ℓ_0-Bregman relaxations (B-rex) penalty B_Ψ [24];
- Truncated Huber Penalty (THP) [25].

Some of these functions are parameterized by variables that can be modified to influence their behavior. These are the CEL0 penalty, B-rex penalty, and THP. In Fig 3.1, we show the plots of the 1D regularizers. In particular, the CEL0 penalty and the B-rex penalty are equal in the plot due to the chosen parameters. For all the plots in this chapter, λ is set to 1.

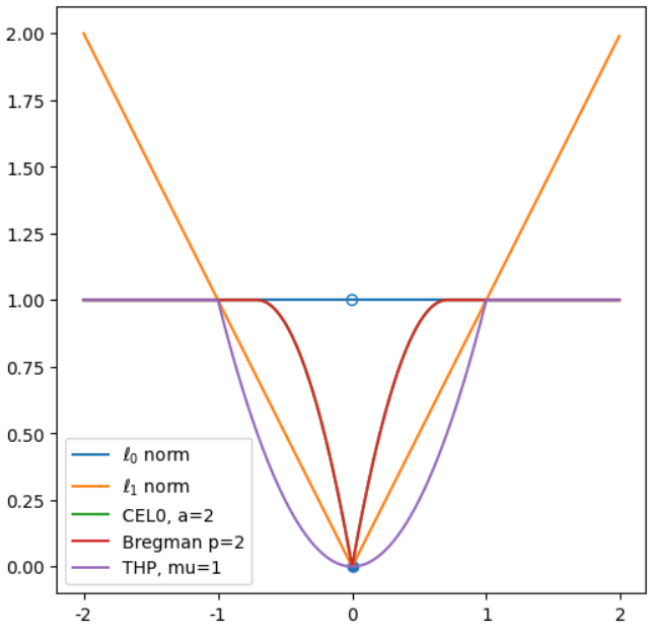


Figure 3.1: *Plot of the scalar functions* $R(\boldsymbol{X})$

Given a generic regularized inverse problem (3.2), we will now discuss the method we will use to solve it. The numerical method we apply is the Proximal Gradient Method [26] (PGM). When using the PGM, we will specify the regularization function used.

3.3 Proximal Gradient Method

The Proximal Gradient Method is an algorithm used to solve optimization problems of the form:

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) + R(\mathbf{x}),$$

where $R(\cdot)$ is a closed proper convex function, and $f(\cdot)$ is a proper, closed, convex, continuously differentiable function. In our case, instead of $\mathbf{x} \in \mathbb{R}^N$, we have $\mathbf{X} \in \mathbb{R}^{N_z\times N\times N}$. The PGM is an iterative algorithm which, starting from $\mathbf{X}_0$, performs two steps at each iteration k:

1. $\mathbf{W}_{k+1} = \mathbf{X}_k - \tau\nabla f(\mathbf{X}_k)$ [forward step];
2. $\mathbf{X}_{k+1} = prox_{\lambda R}(\mathbf{W}_{k+1})$ [backward step].

$\mathbf{X}_{\bar{k}+1} \in \mathbb{R}^{N_z\times N\times N}$, obtained upon convergence of the method or upon reaching the maximum number of iterations allowed, will be the reconstruction of the ground truth. For $j \in \{1,\dots,N_z\}$, the formula for the gradient of the fidelity term in (3.2) is:

$$(\nabla f(\mathbf{X}))^j = (\mathbf{H}^j)^T * (S^T(S(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i) - \mathbf{Y}_{t_n} + \mathbf{B}_{t_n})). \tag{3.4}$$

Meanwhile, $\tau \in \mathbb{R}_{++}$ is the step size and $prox_{\lambda R}$ is the proximal operator of the function λR.

Definition 3.7. *Let $R : \mathbb{R}^n \longrightarrow \mathbb{R}\cup\{+\infty\}$ be a closed proper convex function, $\lambda \in \mathbb{R}_{++}$. The proximal operator of λR is defined as:*

$$prox_{\lambda R}(\boldsymbol{w}) = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ R(\boldsymbol{x}) + \frac{1}{2\lambda}\|\boldsymbol{x}-\boldsymbol{w}\|_2^2 \right\}.$$

We define the proximal operator for $R : \mathbb{R}^{N_z\times N\times N} \longrightarrow \mathbb{R}\cup\{+\infty\}$ using the following result.

Theorem 3.8. *Let $n \in \mathbb{N}, \boldsymbol{x} \in \mathbb{R}^n, \lambda \in \mathbb{R}_{++}, R : \mathbb{R}^n \longrightarrow \mathbb{R}$. If R is separable, i.e.:*

$$R(\boldsymbol{x}) = \sum_{i=1}^{n} R_i(x_i), \tag{3.5}$$

with $R_i : \mathbb{R} \longrightarrow \mathbb{R}$, $i \in \{1,\dots,N\}$, we have:

$$prox_{\lambda R}(\boldsymbol{x}) = (prox_{\lambda R_1}(x_1),\dots,prox_{\lambda R_n}(x_n)).$$

This result can be proven similarly for $R : \mathbb{R}^{N_z \times N \times N} \longrightarrow \mathbb{R} \cup \{+\infty\}$. The functions R we consider are all separable.

Let $R : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$. The PGM can be used even when R is non-convex. In such cases, for the backward step, we have:

$$\mathbf{x}_{k+1} \in prox_{\lambda R}(\mathbf{w}_{k+1}).$$

In fact, when R is convex, $R(\cdot)+\frac{1}{2\lambda}\|\cdot-\mathbf{w}_{k+1}\|_2^2$ is strictly convex, so it has only one global minimum. When R is non-convex, we may have multiple global minima of $R(\cdot) + \frac{1}{2\lambda}\| \cdot -\mathbf{w}_{k+1}\|_2^2$ or none, depending on the function R. The different regularization functions we will consider are lower semi-continuous, proper, and non-negative. Thus, $R(\cdot) + \frac{1}{2\lambda}\| \cdot -\mathbf{w}_{k+1}\|_2^2$ is lower semi-continuous, proper, and coercive. Therefore, it admits a global minimizer, so $prox_{\lambda R}(\mathbf{w}_{k+1})$ is a non-empty set. From this set, we can choose any element.

3.4 Proximal operators

Regarding the backward step, different functions have different proximal operators. Thus, considering the sparsity-promoting regularizer functions stated above, we need to determine the following proximal operators:

- Hard-thresholding operator, which is the proximal operator of the ℓ_0 norm;
- Soft-thresholding operator, which is the proximal operator of the ℓ_1 norm;
- Proximal operator of the Continuous Exact ℓ_0 (CEL0) penalty function;
- Proximal operator of the B-Rex penalty;
- Proximal operator of the THP.

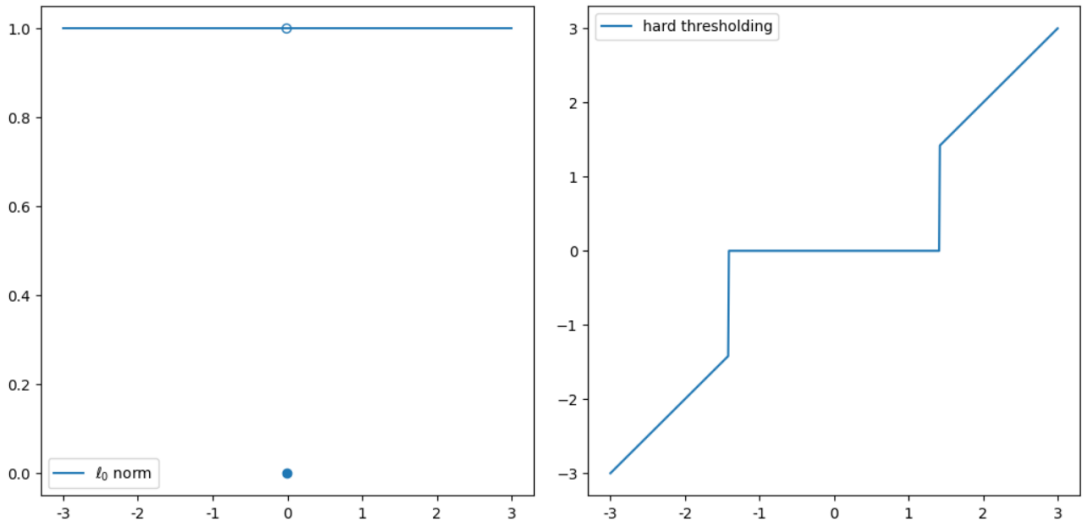
In the following subsections, we will define these proximal operators. The four regularizer functions we are considering all satisfy the property (3.5), which means we can use Theorem 3.8. Thus, we can define the proximal operators for $x \in \mathbb{R}$, and for $\mathbf{X} \in \mathbb{R}^{N_z \times N \times N}$ we apply them element-wise.

3.4.1 Hard-thresholding operator

Given $x, \lambda \in \mathbb{R}, \lambda > 0$, the hard-thresholding operator is defined as follows:

$$prox_{\lambda\|\cdot\|_0}(x) := \text{hard}_{\sqrt{2\lambda}}(x) := \begin{cases} 0 & \text{if} \quad x^2 < 2\lambda \\ \{0, \lambda\} & \text{if} \quad x^2 = 2\lambda \\ x & \text{otherwise} \end{cases} \tag{3.6}$$

In Fig. 3.2, we show the plots of the l_0 norm and its proximal operator.



(a) *Plot of ℓ_0 norm* **(b)** *Plot of hard-thresholding operator*

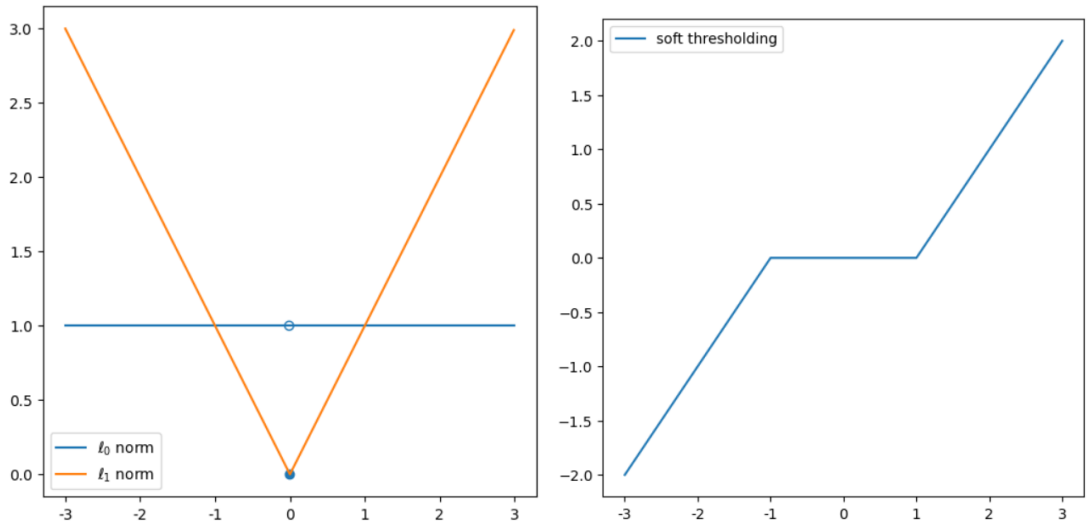
Figure 3.2

3.4.2 Soft-thresholding operator

Given $x, \lambda \in \mathbb{R}, \lambda > 0$, the soft-thresholding operator is defined as follows:

$$\text{prox}_{\lambda\|\cdot\|_1}(x) := \text{soft}_{[-\lambda,\lambda]}(x) := \text{sign}(x)\max(0, |x| - \lambda) \tag{3.7}$$

In Fig. 3.3, we show the plots of the l_1 norm and its proximal operator.



(a) *Plot of ℓ_1 norm* **(b)** *Plot of soft-thresholding operator*

Figure 3.3

3.4.3 Proximal operator of CEL0 penalty

With the CEL0 penalty function, we have a continuous exact relaxation of the problem with the ℓ_0 norm. Let $\mathbf{A}$ be the matrix such that:

$$\mathbf{A}\mathbf{x} = S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i\right), \qquad \mathbf{X} \in \mathbb{R}^{N_z \times N \times N},\ \mathbf{x} = vec(\mathbf{X}) \in \mathbb{R}^{N_z N^2}. \tag{3.8}$$

We can define the CEL0 function as follows:

$$\Phi_{\text{CEL0}}(\mathbf{x}) := \sum_{i=1}^{M} \phi(\|\mathbf{a}_i\|_2, \lambda, x_i) = \sum_{i=1}^{M} \lambda - \frac{\|\mathbf{a}_i\|_2^2}{2}\left(|x_i| - \frac{\sqrt{2\lambda}}{\|\mathbf{a}_i\|_2}\right)^2 \mathbb{1}_{|\mathbf{x}_i| \leq \frac{\sqrt{2\lambda}}{\|\mathbf{a}_i\|_2}} \tag{3.9}$$

Since we are considering tensors, we can write $\Phi_{\text{CEL0}}(\mathbf{X}) = \sum_{i=1}^{M} \phi(\mathbf{A}, \lambda, x_i)$, with x_i being the i-th element of $\mathbf{x} = vec(\mathbf{X})$. Now, the regularized inverse problem (3.2) becomes:

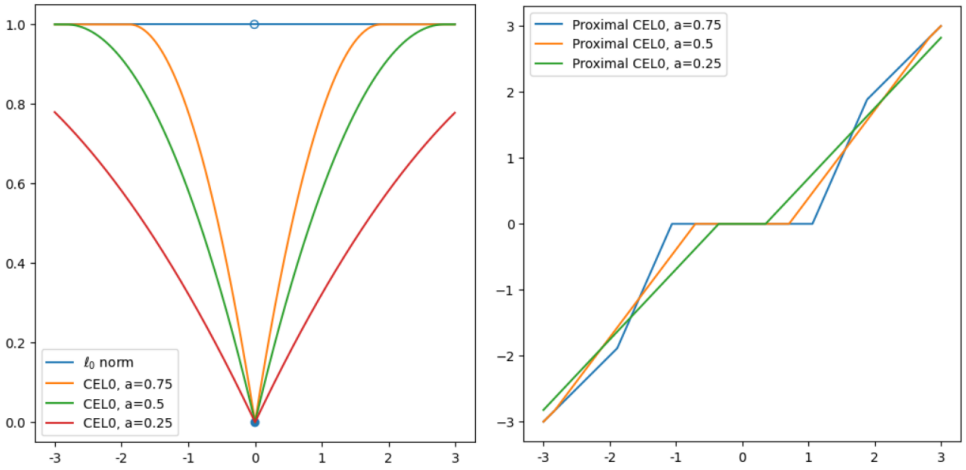
$$\arg\min_{\mathbf{X}} J_C(\mathbf{X}) := \frac{1}{2}\left\|\mathbf{Y}_{t_n} - S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}^i\right) + \mathbf{B}_{t_n}\right\|_2^2 + \Phi_{\text{CEL0}}(\mathbf{X}), \tag{3.10}$$

With the term exact, we mean that $\arg\min_{\mathbf{X}} J_C(\mathbf{X}) = \arg\min_{\mathbf{X}} J_0(\mathbf{X})$, where J_0 is the problem (3.2) with the ℓ_0 norm as regularizer, and $\mathbf{X}$ local minimizer of J_C implies that $\mathbf{X}$ is a local minimizer of J_0

By using the CEL0 penalty function, we aim to obtain a problem with the same global minima and fewer local minima, since we assume the global minima will be a good reconstruction of the ground truth, and reducing the number of local minima helps

the PGM converge to the global minima. The proximal operator of 1-dimensional ϕ_{CEL0} is:

$$\text{prox}_{\gamma\phi_{\text{CEL0}}(a,\lambda,\cdot)}(x) := \begin{cases} \text{sign}(x)\min\left\{|x|, \frac{max\{|x|-\sqrt{2\lambda\gamma}a,0\}}{1-a^2\gamma}\right\} \text{ if } a^2\gamma < 1 \\ x\mathbb{1}_{|x|>\sqrt{2\lambda\gamma}} + \{0,x\}\mathbb{1}_{|x|=\sqrt{2\lambda\gamma}} \text{ if } a^2\gamma \geq 1 \end{cases}$$



(a) *Plot of CEL0 functions* **(b)** *Plot of CEL0 proximal operators*

Figure 3.4

3.4.4 Proximal operator of B-rex

The B-rex penalty B_Ψ depends on the generating functions we consider. In our case, they are $\psi_n(x) := \frac{\gamma_n}{p(p-1)}x$, where $\gamma_n, p \in \mathbb{R}^+$, $p > 1$ are terms we can use to influence the behavior of the B-rex penalty, and $x \in \mathbb{R}$. Given $\mathbf{x} \in \mathbb{R}^N$, we define the ℓ_0-Bregman relaxation as:

$$B_\Psi(\mathbf{x}) := \sum_{n=1}^{N} \beta_{\psi_n}(x_n),$$

with:

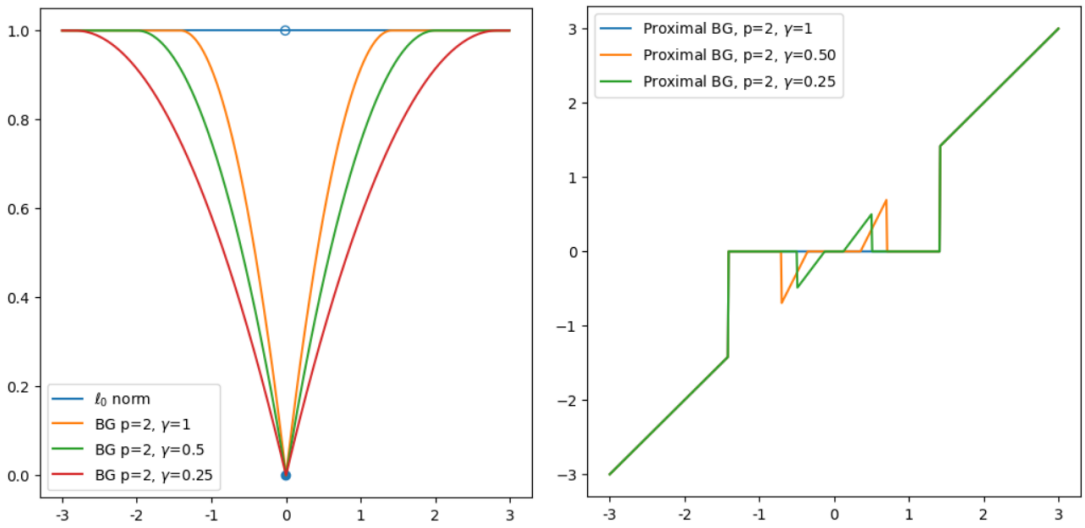
$$\beta_{\psi_n}(x) := \begin{cases} \psi_n(0) - \psi_n(x) + \psi_n'(\alpha_n^-)x & \text{if} \quad x \in [\alpha_n^-, 0] \\ \psi_n(0) - \psi_n(x) + \psi_n'(\alpha_n^+)x & \text{if} \quad x \in [0, \alpha_n^+] \\ \lambda & \text{otherwise} \end{cases}$$

α_n^- and α_n^+ depend on our problem and on the value of γ_n. In certain cases, we find that B-rex for $p = 2$ and the CEL0 penalty coincide. Furthermore, when γ_n assumes certain values, we obtain a continuous exact relaxation of the problem with the l_0 norm.

Assuming $p = 2$, $x > 0$, the proximal operator of the B-rex penalty for the 1D case is:

$$prox_{\lambda\beta_{\psi_n}}(x) := \frac{x - \lambda\psi_n'(\alpha_n^+)}{1 - \lambda\gamma} \cup \{0, x\}$$

For $x < 0$, the computations are similar. When $1 < p < 2$, the proximal operator has different formulas based on the value of p. In Fig. 3.5, we show the plot of the B-rex function and its proximal operator.



(a) *Plot of B-rex functions* **(b)** *Plot of B-rex proximal operators*

Figure 3.5

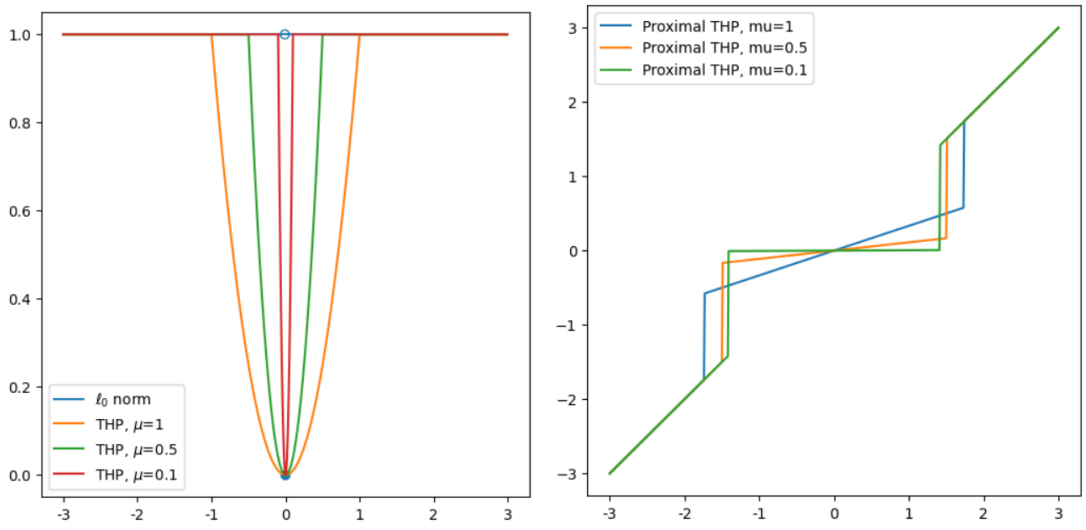
3.4.5 Proximal operator of Truncated Huber Penalty

Finally, we have the THP function. For $x \in \mathbb{R}$, it is defined as:

$$\phi_\mu(x) := \min\left(1, \frac{x^2}{\mu^2}\right),$$

where $\mu \in \mathbb{R}^+$ serves as a threshold. Unlike the classical Huber penalty, it is not biased towards large magnitudes. Now, the proximal operator of $\lambda\phi_\mu(x)$ is:

$$prox_{\lambda\phi_\mu}(x) = \begin{cases} x & \text{if } |x| > \sqrt{\mu^2 + 2\lambda} \\ x \vee \frac{\mu^2}{\mu^2+2\lambda}x & \text{if } |x| = \sqrt{\mu^2 + 2\lambda} \\ \frac{\mu^2}{\mu^2+2\lambda}x & \text{if } |x| < \sqrt{\mu^2 + 2\lambda} \end{cases}$$



(a) *Plot of THP functions* **(b)** *Plot of THP proximal operators*

Figure 3.6

3.4.6 Plots of the operators

In Fig. 3.7, we show plots of the proximal operators all together.

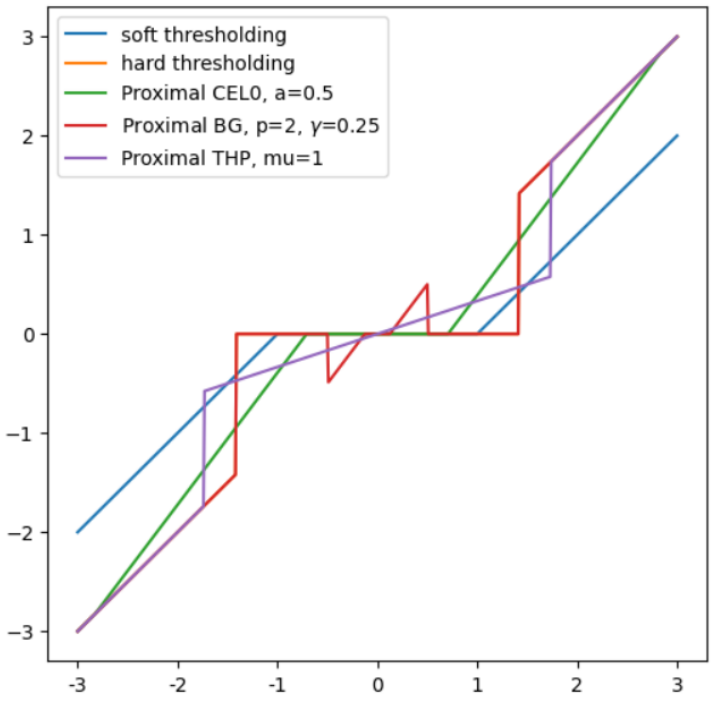


Figure 3.7: *Plot of proximal operators*

3.5 Discrepancy principle

As already mentioned, the λ parameter we choose for the inverse problem influences the results we obtain. Therefore, it is critical to select an optimal regularization parameter. How can we find such a λ? We can use the discrepancy principle. We know that the noisy images $\mathbf{Y}_{t_n}$ we will analyze are obtained as in (2.1). Assuming that the noise is Gaussian with variance σ and zero mean, we have $\|\mathbf{N}_{t_n}\|_2^2 = d\sigma^2$, where d is the number of elements of $\mathbf{N}_{t_n}$. Therefore, we have:

$$\mathbf{Y}_{t_n} - \left(S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}_{t_n}^i \right) + \mathbf{B}_{t_n} \right) = \mathbf{N}_{t_n}, \tag{3.11}$$

which leads to:

$$\left\| \mathbf{Y}_{t_n} - \left(S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}_{t_n}^i \right) + \mathbf{B}_{t_n} \right) \right\|_2^2 = \|\mathbf{N}_{t_n}\|_2^2 = d\sigma^2.$$

Suppose we know an estimate of the variance σ^2, then an optimal estimate of the regularization parameter λ is the one that satisfies:

$$\left\| \mathbf{Y}_{t_n} - \left(S\left(\sum_{i=1}^{N_z} \mathbf{H}^i * \mathbf{X}_{t_n}^i \right) + \mathbf{B}_{t_n} \right) \right\|_2^2 = \tau d\sigma^2, \tag{3.12}$$

where $\mathbf{B}_{t_n}$ is the estimated background, $\tau > 1, \tau \approx 1$ is a scaling factor. The main disadvantage of the discrepancy principle is its sensitivity to the accuracy in estimating σ^2. If we have an inaccurate estimate of σ^2, this method does not produce an optimal λ.

3.6 PGM algorithms for SMLM

The structure of our PGM algorithms varies slightly depending on the regularizer function used. Below, we will present four versions of the PGM algorithm.

Algorithm 1. PGM-ℓ_1 (ISTA)

Input: $\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of iterations, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term

Initializations: $\mathbf{B}_{t_n}$ estimated background, $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\mathbf{Y}_{t_n})$ for $j \in \{1, \dots, N_z\}$, k=0

while $(k < maxit)$:

compute $\nabla f(\mathbf{X}_k)^j$ by (3.4) for $j = 1 : \dots : N_z$

Forward step $\mathbf{W}_{k+1} = \mathbf{X}_k - \tau \nabla f(\mathbf{X}_k)^T$

Backward step $\mathbf{X}_{k+1} = \text{soft}_{[-\lambda,\lambda]}(\mathbf{W}_{k+1})$

$k = k + 1$

if $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F / \|\mathbf{X}_{k-1}\|_F < tol$: break

end while

Output: $(\mathbf{X}_{t_n})^* = \mathbf{X}_k$

Algorithm 2. PGM-CEL0

Input: $\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of iterations, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term

Initializations: $\mathbf{A}$ tensor containing the norms of the columns of the matrix associated with $\mathbf{H}$, $\mathbf{B}_{t_n}$ estimated background, $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\mathbf{Y}_{t_n})$ for $j \in \{1, \dots, N_z\}$, k=0

while $(k < maxit)$:

compute $\nabla f(\mathbf{X}_k)^j$ by (3.4) for $j = 1 : \dots : N_z$

Forward step $\mathbf{W}_{k+1} = \mathbf{X}_k - \tau \nabla f(\mathbf{X}_k)^T$

Backward step $\mathbf{X}_{k+1} = \text{prox}_{\lambda \phi_{\text{CEL0}}(\mathbf{A},\tau,\cdot)}(\mathbf{W}_{k+1})$

$k = k + 1$

if $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F / \|\mathbf{X}_{k-1}\|_F < tol$: break

end while

Output: $(\mathbf{X}_{t_n})^* = \mathbf{X}_k$

Algorithm 3. PGM-B-rex

Input: $\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of iterations, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term

Initializations: $\mathbf{A}$ tensor containing the norms of the columns of the matrix associated with $\mathbf{H}$, $\mathbf{B}_{t_n}$ estimated background, $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\mathbf{Y}_{t_n})$ for $j \in \{1, \ldots, N_z\}$, k=0

while $(k < maxit)$:

compute $\nabla f(\mathbf{X}_k)^j$ by (3.4) for $j = 1 : \ldots : N_z$

Forward step $\mathbf{W}_{k+1} = \mathbf{X}_k - \tau \nabla f(\mathbf{X}_k)^T$

Backward step $\mathbf{X}_{k+1} = prox_{\lambda B_\Psi}(\mathbf{W}_{k+1})$

$k = k + 1$

if $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F / \|\mathbf{X}_{k-1}\|_F < tol$: break

end while

Output: $(\mathbf{X}_{t_n})^* = \mathbf{X}_k$

Algorithm 4. PGM-THP

Input: $\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of iterations, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term

Initializations: $\mathbf{B}_{t_n}$ estimated background, $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\mathbf{Y}_{t_n})$ for $j \in \{1, \ldots, N_z\}$, k=0

while $(k < maxit)$:

compute $\nabla f(\mathbf{X}_k)^j$ by (3.4) for $j = 1 : \ldots : N_z$

Forward step $\mathbf{W}_{k+1} = \mathbf{X}_k - \tau \nabla f(\mathbf{X}_k)^T$

Backward step $\mathbf{X}_{k+1} = \text{prox}_{\lambda \phi_\mu}(\mathbf{W}_{k+1})$

$k = k + 1$

if $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F / \|\mathbf{X}_{k-1}\|_F < tol$: break

end while

Output: $(\mathbf{X}_{t_n})^* = \mathbf{X}_k$

As we can see, the main differences are in the initializations and the backward step. For the PGM-CEL0 and PGM-B-rex penalty functions, we need to initialize the tensor $\mathbf{A}$ to compute the proximal operator, while the PGM-THP and PGM-ℓ_1 do not require this tensor. Instead, the algorithms differ because each regularizer has a different proximal operator.

3.6.1 Backtracking algorithm

For the PGM, the choice of the step size is important. To determine an optimal step size, we use the following result, which can be extended to $\mathbf{X} \in \mathbb{R}^{N_z \times N \times N}$.

Theorem 3.9 (Descent Lemma)**.** *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a gradient-Lipschitz continuous function with Lipschitz constant L_f. For any $L \geq L_f$, we have:*

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + < \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x}) > + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$$

Proof. Using the Taylor expansion of $f(\mathbf{x})$, we have:

$$\begin{aligned}
f(\mathbf{x}) &= f(\mathbf{y}) + \int_0^1 \nabla f(\mathbf{y} + \tau(\mathbf{x} - \mathbf{y}))^T (\mathbf{x} - \mathbf{y}) d\tau \\
&= f(\mathbf{y}) + \nabla f(\mathbf{y})^T (\mathbf{x} - \mathbf{y}) + \int_0^1 (\nabla f(\mathbf{y} + \tau(\mathbf{x} - \mathbf{y})) - \nabla f(\mathbf{y}))^T (\mathbf{x} - \mathbf{y}) d\tau \\
&\leq f(\mathbf{y}) + \nabla f(\mathbf{y})^T (\mathbf{x} - \mathbf{y}) + \int_0^1 \|\nabla f(\mathbf{y} + \tau(\mathbf{x} - \mathbf{y})) - \nabla f(\mathbf{y})\|_2^T \|\mathbf{x} - \mathbf{y}\|_2 d\tau \\
&\leq f(\mathbf{y}) + \nabla f(\mathbf{y})^T (\mathbf{x} - \mathbf{y}) + L \int_0^1 \|\mathbf{x} - \mathbf{y}\|^2 \tau d\tau \\
&= f(\mathbf{y}) + < \mathbf{x} - \mathbf{y}, \nabla f(\mathbf{x}) > + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2
\end{aligned}$$

For the first inequality, we use the Cauchy–Schwarz inequality, and for the second, we use the fact that f is gradient-Lipschitz continuous. □

Thus, we have a convex quadratic upper bound for f that reaches a minimum with step size $\tau = \frac{1}{L_f}$. We can use this result even though the fidelity term f of the inverse problem is $f : \mathbb{R}^{N_x \times N \times N} \longrightarrow \mathbb{R}$, by considering vectorized version of the tensors. Thus, if L_f is known, we can use the constant step size $\tau = \frac{1}{L_f}$. Given the fidelity term (3.3), the associated linear operator (3.8) can be written in matrix form as

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{N_z}],$$

where each $\mathbf{A}_i$ is a matrix such that:

$$\mathbf{A}_i \mathbf{x}^i = S(\mathbf{H}_i * \mathbf{X}^i), \qquad \mathbf{X} \in \mathbb{R}^{N_z \times N \times N}, \mathbf{x}_i = vec(\mathbf{X}^i).$$

Let $\mathbf{x} \in \mathbb{R}^{N_z N^2}, \mathbf{x} = vec(\mathbf{X}) \in \mathbb{R}^{N_z \times N \times N}$, and let $\mathbf{y}_{t_n} = vec(\mathbf{Y}_{t_n}), \mathbf{b}_{t_n} = vec(\mathbf{B}_{t_n}) \in \mathbb{R}^{N^2}$. The gradient of the fidelity term can be formulated as:

$$\nabla f(\mathbf{x}) = \mathbf{A}^T (\mathbf{A}\mathbf{x} - \mathbf{y}_{t_n} + \mathbf{b}_{t_n}).$$

Let $\hat{\mathbf{x}} \in \mathbb{R}^{N_z N^2}$. Then,

$$\|\nabla f(\mathbf{x}) - \nabla f(\hat{\mathbf{x}})\|_2 = \|\mathbf{A}^T \mathbf{A}(\mathbf{x} - \hat{\mathbf{x}})\|_2 \leq \|\mathbf{A}^T \mathbf{A}\|_2 \, \|\mathbf{x} - \hat{\mathbf{x}}\|_2. \tag{3.13}$$

Therefore, the Lipschitz constant of ∇f is:

$$L_f := \|\mathbf{A}^T\mathbf{A}\|_2.$$

Let $\mathbf{x}^i = vec(\mathbf{X}^i) \in \mathbb{R}^{N^2}$. We have:

$$\|\mathbf{A}\mathbf{x}\|_2 = \left\|\sum_{i=1}^{N_z} \mathbf{A}_i\mathbf{x}^i\right\|_2 \leq \sum_{i=1}^{N_z} \|\mathbf{A}_i\|_2 \|\mathbf{x}^i\|_2 \leq \left(\sum_{i=1}^{N_z} \|\mathbf{A}_i\|_2^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{N_z} \|\mathbf{x}^i\|_2^2\right)^{\frac{1}{2}},$$

where the last inequality follows from the Cauchy-Schwarz inequality with the vectors $(\|\mathbf{A}_1\|_2, \ldots, \|\mathbf{A}_{N_z}\|_2)$, $(\|\mathbf{x}^1\|_2, \ldots, \|\mathbf{x}^{N_z}\|_2)$. Since:

$$\|\mathbf{A}^T\mathbf{A}\|_2 = \|\mathbf{A}\|_2^2 = \max_{\mathbf{x}\in\mathbb{R}^{N_z N^2}, \|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2^2,$$

it follows that:

$$\|\mathbf{A}^T\mathbf{A}\|_2 \leq \sum_{i=1}^{N_z} \|\mathbf{A}_i\|_2^2 =: L_{est}. \tag{3.14}$$

In particular, if our discrete astigmatic PSF $\mathbf{H}$ is normalized so that the sum of all elements in the same horizontal slice is 1, we have:

$$\|\mathbf{A}_i\|_2^2 = L^2,$$

where L is the downsampling factor.

Now, if L_{est} is very high, and thus the step size $\tau = \frac{1}{L_{est}}$ is very small, we can use a backtracking strategy. The fidelity term is gradient-Lipschitz continuous with Lipschitz constant L_f, so for any $L \geq L_f$ we have in our case:

$$f(\mathbf{X}) \leq f(\mathbf{Y}) + < \mathbf{X} - \mathbf{Y}, \nabla f(\mathbf{X}) > + \frac{L}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2$$

Let:

$$Q_{1/L}(\mathbf{X}, \mathbf{Y}) := f(\mathbf{Y}) + < \mathbf{X} - \mathbf{Y}, \nabla f(\mathbf{X}) > + \frac{L}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2 + \lambda R(\mathbf{X}),$$

and:

$$p_L^{f,\lambda R}(\mathbf{Y}) := prox_{\lambda R\frac{1}{L}}(\mathbf{Y} - \frac{1}{L}\nabla f(\mathbf{Y})),$$

for any $L \geq L_f$ we have:

$$F(p_L^{f,\lambda R}(\mathbf{X})) \leq Q_{1/L}(p_L^{f,\lambda R}(\mathbf{X}), \mathbf{X}) \tag{3.15}$$

Therefore, we can choose $L < L_{est}$ and adjust the guess to satisfy the condition (3.15). Below, we show an implementation of PGM-CEL0 with a backtracking strategy. In this case, if our step size doesn't satisfy the condition (3.15), we reduce it by a positive factor $\rho < 1$. We continue until the condition is satisfied or the maximum number of iterations per backtracking is reached.

Algorithm 5. PGM-CEL0 with backtracking

Input:$\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of global iterations, N maximum number of iterations per backtracking, $\tau = \frac{1}{L}$ starting step-size, $\rho < 1$, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term

Initializations: $\mathbf{A}$ tensor containing the norms of the columns of the matrix associated with $\mathbf{H}$, $\mathbf{B}_{t_n}$ estimated background, $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\mathbf{Y}_{t_n})$ for $j \in \{1, \ldots, N_z\}$, k=0, n=0

while $(k < maxit)$:

compute $\nabla f(\mathbf{X}_k)^j$ by (3.4) for $j = 1 : \ldots : N_z$

while $n < N$ and $F(p_{1/\tau}^{f,\lambda R}(\mathbf{X}_k)) > Q_\tau(p_{1/\tau}^{f,\lambda R}(\mathbf{X}_k), \mathbf{X}_k)$:

$\tau = \tau\rho$, $n = n + 1$

end while

$\mathbf{X}_{k+1} = p_{1/\tau}^{f,\lambda R}(\mathbf{X}_k)$

k=k+1, $n = 0$

if $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F / \|\mathbf{X}_{k-1}\|_F < tol$: break

end while

Output: $(\mathbf{X}_{t_n})^* = \mathbf{X}_k$

Chapter 4

Numerical experiments

In this chapter, we discuss the numerical implementation of methods for localizing emitters in STORM images, the data used for testing, and the results obtained. We will also introduce other methods for localizing emitters in STORM images.

4.1 PGM-based SMLM algorithm

First, we present the SMLM-PGM-R algorithm we have developed, where R denotes the regularization function we use. This algorithm solves the optimization problem (3.2)-(3.3) via the PGM numerical method, followed by a post-processing step for emitter localization. Given an image $\mathbf{Y}_{t_n} \in \mathbb{R}^{N/L \times N/L}$, with $t_n \in [0, T]$, the SMLM-PGM-R algorithm is divided into 4 steps:

1. Normalization of $\mathbf{Y}_{t_n}$ by dividing each element by the maximum value of the elements of $\mathbf{Y}_{t_n}$. As a result, $\mathbf{Y}_{t_n} \in [0,1]^{N_z \times N \times N}$;

2. Estimation of the background $\mathbf{B}_{t_n}$ in $\mathbf{Y}_{t_n}$ and its removal from the image to obtain $\hat{\mathbf{Y}}_{t_n} = \mathbf{Y}_{t_n} - \mathbf{B}_{t_n}$. We use the median filter to estimate the background;

3. Computation of the 3D reconstruction of $(\mathbf{X}_{t_n})^*$ by solving (3.2)-(3.3) using the PGM. Since we removed the background beforehand, we can assume $\mathbf{B}_{t_n} = 0$ for the computation of the gradient, and our initial point will be $\mathbf{X}_0$ so that $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\hat{\mathbf{Y}}_{t_n})$ for $j \in \{1, \ldots, N_z\}$;

4. Extraction of local maxima in the 3D reconstruction $(\mathbf{X}_{t_n})^*$ with values above a certain threshold. These local maxima will be considered the emitters' localizations.

We summarize the SMLM-PGM-R algorithm in Algorithm 6.

Algorithm 6. SMLM-PGM-R

Input: $\mathbf{Y}_{t_n}$ STORM acquisition, τ step size, *maxit* maximum number of iterations, *tol* tolerance for the relative error between reconstructions, $\mathbf{H}$ discretized PSF, S downsampling operator, λ regularization term, maximum threshold η

Initializations: $\mathbf{X}_0 = \mathbf{H}^T * S^T(\mathbf{Y}_{t_n} - \mathbf{B}_{t_n})$

Step 1: Normalization of $\mathbf{Y}_{t_n}$ by dividing each element by the maximum value;

Step 2: Estimation of the background $\mathbf{B}_{t_n}$ in $\mathbf{Y}_{t_n}$ and its removal from the image to obtain $\hat{\mathbf{Y}}_{t_n} = \mathbf{Y}_{t_n} - \mathbf{B}_{t_n}$;

Step 3: Computation of the 3D reconstruction of $(\mathbf{X}_{t_n})^*$ by using the PGM-R initialized by $\mathbf{X}_0^j = (\mathbf{H}^j)^T * S^T(\hat{\mathbf{Y}}_{t_n})$ for $j \in \{1, \dots, N_z\}$;

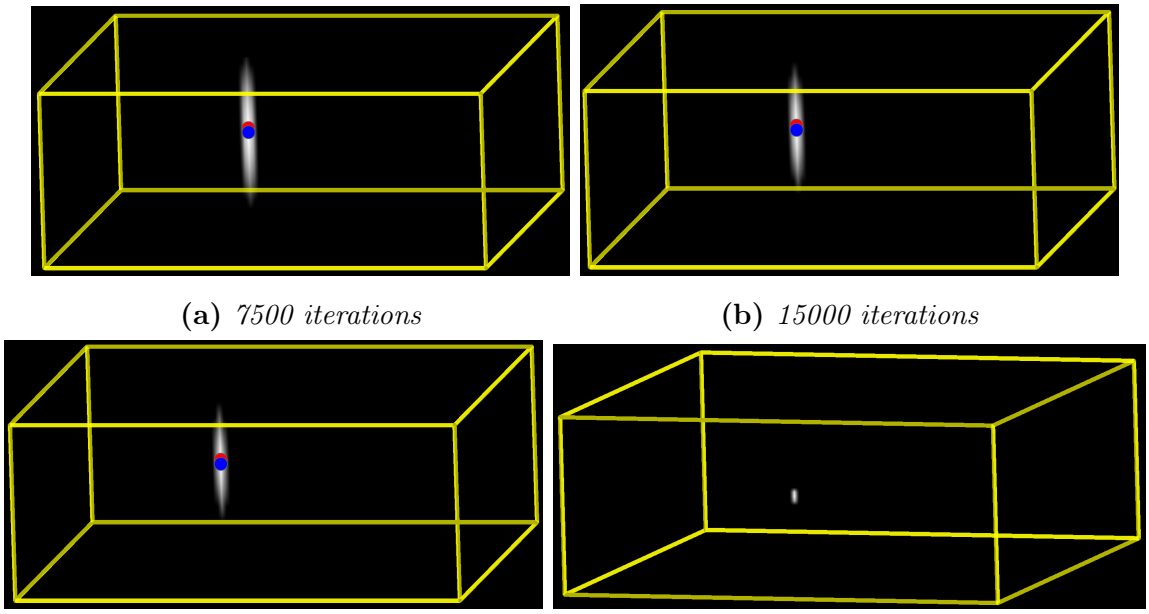
Step 4: Extraction of local maxima in the 3D reconstruction $(\mathbf{X}_{t_n})^*$ with values above the threshold η.

Output: List of local maxima

We use this algorithm for every frame $\mathbf{Y}_{t_n}$ of our image sequences. Since the PSF remains constant between different time frames, we can compute its FFT once and use the result for every frame. The tolerance *tol* for the reconstruction's relative error in Algorithm PGM-R is $tol = 10^{-4}$. The step size is $\tau = \frac{1}{L_{est}}$, where L_{est} is the upper bound of the Lipschitz constant computed in Chapter 3 using $S(\cdot)$ and $\mathbf{H}$. Since we remove the background in $\mathbf{Y}_{t_n}$ during step 2, we do not include the background in the fidelity term. Without this step, we would need to include the matrix $\mathbf{B}_{t_n}$ in the fidelity term. For the third step, we can use any of the four PGM algorithms shown in the previous chapter.

In the fourth step, we extract the maxima because a very high number of iterations is required for the reconstruction to resemble a point, and we have a limited computational budget. The parameter η used for $\mathbf{Y}_{t_n}$ is $\eta = max(\mathbf{Y}_{t_n} - \mathbf{B}_{t_n})/3$, where $max(\mathbf{Y}_{t_n} - \mathbf{B}_{t_n})$ is the maximum element of $\mathbf{Y}_{t_n} - \mathbf{B}_{t_n}$. We show the obtained reconstruction using different number of iterations in Fig. 4.1. The white shape represents the reconstructed $\mathbf{X}_{t_n}$, the red dot represents the ground truth, and the blue dot represents the local maximum in the reconstruction. As the number of iterations increases, the reconstruction becomes more accurate. In Fig. 4.1d, the reconstructions resemble a point, which is the result we aim to achieve.

Instead of applying Algorithm 6 to the entire image, we could use it for a few Regions of Interests (ROIs) of $\mathbf{Y}_{t_n}$. The ROIs will be the neighborhoods of local maxima above a certain threshold of $\mathbf{Y}_{t_n}$ minus the estimated background. To apply the previous algorithm to the ROIs, we will also extract the central part of the PSF along the x and y directions. After computing the maximum from the reconstruction, we save it in a list



(a) *7500 iterations* **(b)** *15000 iterations*

(c) *22500 iterations* **(d)** *29423 iterations*

Figure 4.1: *Evolution of the obtained reconstruction* $(\boldsymbol{X}_{t_n})^*$

and subtract from $\mathbf{Y}_{t_n}$ in the ROI the image resulting from applying the downsampling operator S and the blurring operator $\mathbf{H}$. To avoid issues when extracting the ROI, we add zero-padding around $\mathbf{Y}_{t_n}$.

The ROI size depends on the downsampling factor L. If $L = 2$, we use a square of size 19×19 pixels; if $L = 4$, the square is 9×9; if $L = 8$, the square is 5×5. We choose a size so that a single emitter is entirely inside the ROI, but not so large as to significantly increase computational costs. In all cases, the squares are centered on the local maxima of $\mathbf{Y}_{t_n}$ minus the background. In Fig 4.2, we show an example of a ROI inside the yellow square.

For the SMLM-PGM algorithm and the computation of L_{est}, we do not consider the entire PSF $\mathbf{H}$, but rather a portion of it corresponding in size to the ROI.

4.1.1 Estimation of the regularization parameter λ

The value of λ used in the PGM influences the results we obtain. If we do not have information on the noise present in our images, we cannot use the discrepancy principle, since it is sensitive to the estimated noise magnitude.

When analyzing a STORM movie from our simulated data, we want a λ that maximizes the Jaccard index over the entire movie. However, the best λ varies from image to image. So, how can we find a good λ?

To obtain it, we can define a set of different values for λ in a range. From this

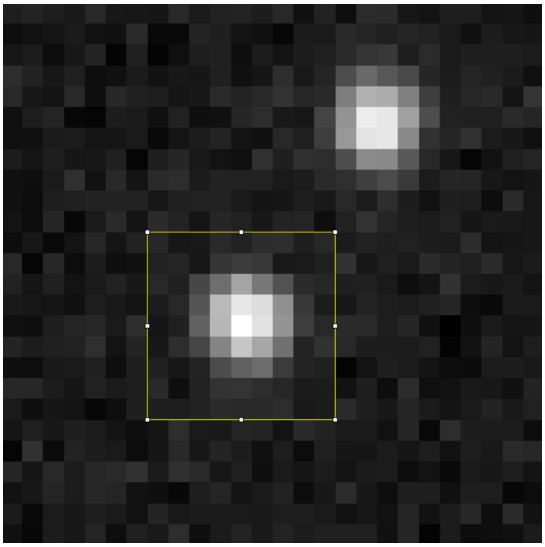


Figure 4.2: *Example of a ROI*

uniform discretization, we will select the element that maximizes the Jaccard index over a specified subset of frames from the movie and use it for all the images. We can do this if we have the ground truth of the images from the movie. In Fig. 4.3, we can see the result of a search for the optimal λ.

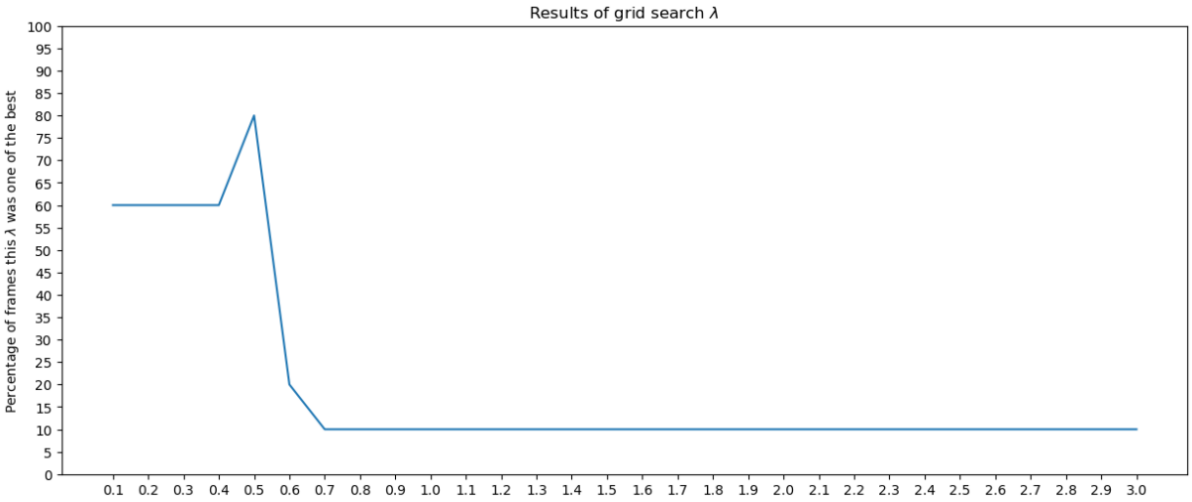


Figure 4.3: λ *search result*

Another approach could be to implement a neural network that, given an image, estimates an optimal λ to recover the (x, y, z) coordinates of the emitters in the image.

4.2 Non-proximal gradient methods

We will compare our method with another software used for 3D SMLM. This software is called ThunderSTORM [27]. We will also introduce other methods, such as:

- 3D-DAOSTORM [28];
- DECODE [29];

4.2.1 ThunderSTORM

ThunderSTORM is a plugin for ImageJ that accomplishes tasks such as simulating, processing, localizing, and visualizing images obtained via SMLM methods. ImageJ is open-source software for processing and analyzing scientific images. It is highly customizable, allowing users to add plugins. In particular, ThunderSTORM localizes emitters in three steps:

1. Firstly, it applies a filter to the image to remove some noise. There are several options available. Some filter options are Gaussian, Difference of Gaussian, and median;
2. Secondly, it computes the approximate location of emitters by taking either the local maxima in the image, the centroids of connected components, or using non-maxima suppression;
3. Finally, it computes the sub-pixel location of the emitters using fitting methods such as least squares, weighted least squares, or maximum likelihood estimation for neighborhoods of the emitters. For fitting, ThunderSTORM uses the calibration curves for a user-provided astigmatic PSF.

For the fitting of a single emitter using least-squares, ThunderSTORM minimizes the sum:

$$\chi^2(\theta \mid \mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} (\mathbf{Y}_{t_n}(x,y) - h(x,y \mid \theta))^2,$$

where:

- $\mathcal{D}$ is the fitting region, which is a neighborhood of the emitter;
- h is the astigmatic PSF, which is user-given and approximated using Gaussian fitting;
- $\mathbf{Y}_{t_n}$ is the STORM image being analyzed;

- θ are the PSF parameters, $\theta = (\theta_x, \theta_y, \theta_z, \theta_N, \theta_b)$. $(\theta_x, \theta_y, \theta_z)$ are the emitter's coordinates, θ_N is the number of photons being emitted, and θ_b is the strength of the background.

4.2.2 3D-DAOSTORM

For the PSF approximation, 3D-DAOSTORM uses Gaussian fitting of the PSF. 3D-DAOSTORM is an iterative algorithm. In every iteration, the following 4 steps are performed:

1. Identifying localizations by localizing pixels with higher intensity than all their neighbors within a specified radius, greater than a user-selected threshold. In this step, we can also choose localizations that were previously discarded in later steps, but only once for each localization. If no new localizations have been found, the algorithm stops. Otherwise, it continues to step 2;

2. Refining these localizations by minimizing the fitting error for each emitter. For the minimization of each localization, our parameters are the x and y coordinates of the localization's center, the σ_x and σ_y of the astigmatic PSF, and the localization's intensity;

3. Discarding bad localizations that have at least one parameter whose value has converged to a negative value, and running step 2 one more time to give localizations that have not yet converged a chance to converge. Then, we advance to step 4;

4. Updating the residual image with the converged localizations. After this, we return to step 1 if we have not reached the maximum number of iterations.

4.2.3 DECODE

DECODE (deep context dependent) is a neural network that achieves the localization of single emitters in high-density situations with very high accuracy. Furthermore, it is consistent in handling the 3D case when using different imaging techniques. How does DECODE manage to do this?

1. DECODE uses temporal context. This means that to analyze a time frame, DECODE also uses the frames before and after, since some emitters can persist in multiple consecutive frames;

2. DECODE can be divided into two stages: in the first, we analyze the three frames independently to create feature representations of single frames; in the second, we combine these representations to generate the final results;

3. Concerning the results, DECODE doesn't compute the voxel containing the emitter. It computes the probability that an emitter is in a voxel, the brightness of the emitter, which is measured by the number of photons N, the vector $(\Delta x, \Delta y, \Delta z)$ to determine the position of the molecule inside the voxel, the background intensity for each pixel, the uncertainty of the localization $(\sigma_x, \sigma_y, \sigma_z)$ in each direction, and the uncertainty of the brightness σ_N;

4. DECODE uses a cubic spline approximation to model the 3D PSF, which is more accurate than a Gaussian approximation;

5. The loss function consists of three terms: a count loss that compares the number of detected and true emitters, a localization loss that estimates localizations and their uncertainties, and a background loss that estimates the background.

In Fig. 4.4, we can see the architecture of DECODE.

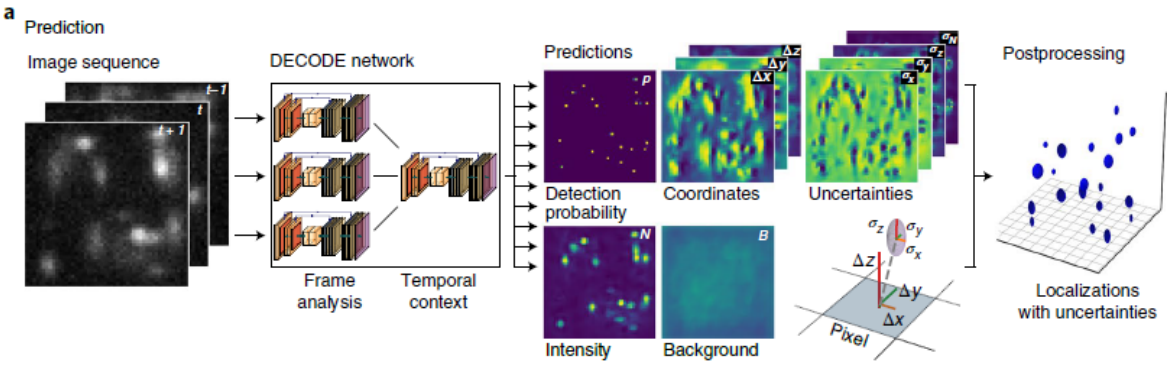


Figure 4.4: *Architecture of DECODE [29]*

4.3 Validation metrics

In the following, we present the metrics to evaluate the data we have and the results obtained from the localization methods.

4.3.1 Image evaluation

To evaluate an image from our dataset, we can use the following metrics:

- $PSNR$, which represents the peak strength of the true signal compared to the noise. Given an image $\mathbf{Y} \in \mathbb{R}^{M \times N}$ without noise, and its noisy version $\hat{\mathbf{Y}} \in \mathbb{R}^{M \times N}$, we have:

$$PSNR(\mathbf{Y}, \hat{\mathbf{Y}}) = 10 \log_{10} \left(\frac{\max(\mathbf{Y})^2}{\|\mathbf{Y} - \hat{\mathbf{Y}}\|_F^2 / (MN)} \right),$$

  where $\|\mathbf{Y}\|_F := \left(\sum_{i=1}^{M} \sum_{j=1}^{N} (\mathbf{Y}_{ij})^2 \right)^{1/2}$ is the Frobenius norm of $\mathbf{Y}$.

- *Spot SBR*, which evaluates the signal strength of a spot against the background in the surrounding area with no signal. Let MS be the mean of the true signal, and MB the mean of the surrounding area. We have:

$$spot\ SBR(MS, MB) = \frac{MS - MB}{MB}$$

- *Spot SNR*, which is used to evaluate the strength of the true signal in a spot against the noise in the surrounding area where there is no signal. Let *std* be the standard deviation in the surrounding area, then the *spot SNR* is defined as follows:

$$spot\ SNR(MS, MB, std) = \frac{MS - MB}{std}$$

4.3.2 Results assessment

To evaluate the results of Algorithm 6 SMLM-PGM-R and ThunderSTORM, we can use the following groups of metrics:

- Reconstruction-based metrics compare the original image with the image obtained using the reconstructed image (in the 2D case) or the reconstructed volume (in the 3D case).

- Localization-based metrics compare the extracted maxima from the reconstruction (using certain criteria) to the ground truth by matching them. For these metrics, true positives (TP) are the number of localizations that matched with an emitter in the ground truth, false positives (FP) are the number of localizations that did not match an emitter in the ground truth, and false negatives (FN) are the number of emitters in the ground truth that did not match a localization. In Fig. 4.5, we show examples of TP, FP, and FN, with red circles representing elements of the ground truth, and blue circles representing localizations.

For reconstruction-based metrics, we can compute the residual mean square error ($RMSE$) between the original image and the reconstructed one. For localization-based metrics, there are three options:

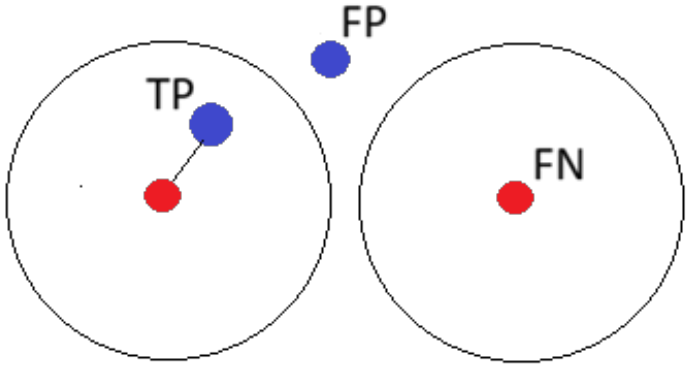


Figure 4.5: *Example of TP, FP, FN*

- The Jaccard index, which is a percentage that measures the fraction of correct localizations in a dataset. It is defined as:

$$Jaccard(TP, FP, FN) = 100\frac{TP}{TP + FN + FP};$$

- The $RMSE$ lateral and axial, which are used to compute the accuracy of correct localizations. $RMSE$ lateral is defined as:

$$RMSE\ \ lat(\{(x_i^l, y_i^l, z_i^l)\}_{i\in S}, \{(x_i^g, y_i^g, z_i^g)\}_{i\in S}) = \sqrt{\frac{1}{TP}\sum_{i\in S}(x_i^l - x_i^g)^2 + (y_i^l - y_i^g)^2},$$

  where S is the set of indices of correct localizations, $x_i^l.y_i^l$ and z_i^l represent the coordinates of the i-th correct localizations, and x_i^g, y_i^g and z_i^g are the coordinates of the i-th ground truth's emitter. Instead, the $RMSE$ axial is defined as:

$$RMSE\ \ ax(\{(x_i^l, y_i^l, z_i^l)\}_{i\in S}, \{(x_i^g, y_i^g, z_i^g)\}_{i\in S}) = \sqrt{\frac{1}{TP}\sum_{i\in S}(z_i^l - z_i^g)^2};$$

- The efficiency, which combines the previous metrics into one. It is defined as:

$$Efficiency(Jaccard, RMSE) = 100 - \sqrt{(100 - JAC)^2 + \alpha^2 RMSE^2}.$$

  The parameter α balances the Jaccard index and the $RMSE$. If we consider the RMSE axial, we can set $\alpha = 0.5$ nm^{-1}. If we only consider the $RMSE$ lateral, we can set $\alpha = 1$ nm^{-1}. If we want to use both to compute the efficiency, we can take the average of the two $RMSEs$.

These metrics can only be used to evaluate our algorithms on simulated data, where the ground truth is available.

4.4 Simulated PSF and experimental datasets

To test these methods and our algorithm, we will use simulated data and PSF.

4.4.1 Simulated PSF

The PSF used in our numerical experiments was generated with the BrightEyes-ISM toolbox [30]. In particular, 30 PSFs have been generated using the vectorial case of the Richard's-Wolf integral in Cartesian parameterization [31]. These PSFs are polarized, so they are averaged over linear polarization orientations to obtain a non-polarized PSF. Afterwards, the PSF is normalized so that the sum of the elements in the same horizontal slice is 1. In Fig 4.6, we show slices of polarized PSFs, while in 4.7, we show the averaged PSF.

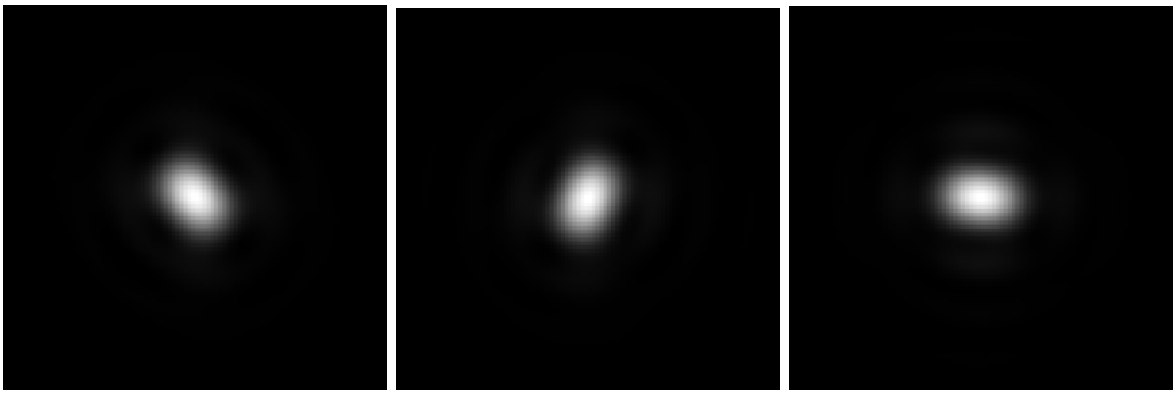


Figure 4.6: *Different polarized PSFs*

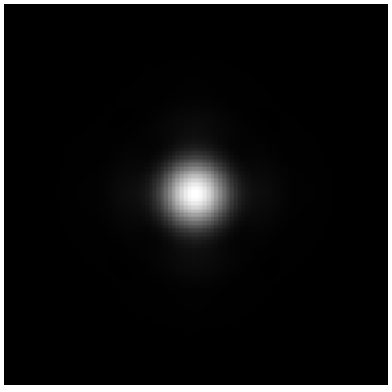


Figure 4.7: *Averaged out PSF*

4.4.2 Simulated datasets

To create the simulated datasets, we start by loading a low-resolution ball-and-stick representation of a small piece of chromosome 21 [32]. This representation spans the portion 10.4–46.7 Megabases (Mb) of chromosome 21, with a genomic resolution of 50 kilobases (kb). The chromosome is from the hg38 assembly. To create our datasets, we will use only 1 Mb.

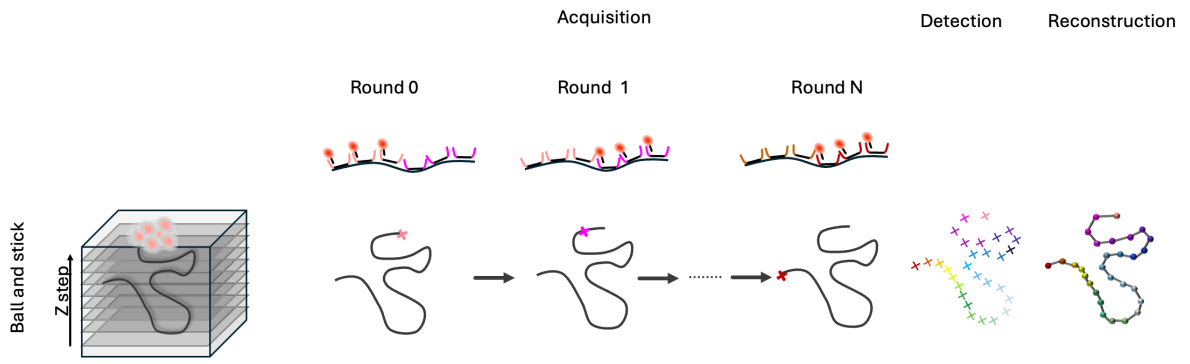


Figure 4.8: *Example of ball-and-stick representation*

What is a ball-and-stick representation? Suppose we want to analyze different genomic regions of a sample. For each genomic region, a few Oligopaint probes with fluorophores are used. These fluorophores are activated and localized, and the centroid of these localizations is computed. We do this for every region. In Fig 4.8, we show an example of a ball-and-stick representation. Spheres represent the centroids of regions, and sticks represent connections between neighboring regions.

Now, the loaded ball-and-stick representation is a list of coordinates of the spheres that exists inside a bounding box $\mathcal{B}$ which is 2000 nm tall, and 2080 nm wide and long. We discretize the bounding box and the coordinates of the centers of the spheres to simulate a point cloud $\mathbf{X} \in \mathbb{R}^{201\times104\times104}$ consisting of three different types of points:

1. Bound probes, whose positions are determined using the centroids and segments of the ball-and-stick representation;

2. Unbound probes;

3. Probes not attached to the regions of the chromosomes to which they are supposed to bind.

The last two types of points are considered to include noise in the simulation. However, in our case, we want to localize all of them. For volumetric chromatin tracing, we want

to separate these localizations. Since the point cloud is 2000 nm tall, we divide it into 7 z-sections, each one 400 nm tall. We call the first section from the bottom z-1, and so forth. As seen in Fig 4.9, a central cluster that likely contains points of the first type is present, surrounded by considerable noise. The other points will be more of the second and third types.

Using an image simulator created with the mathematical forward model defined in Chapter 2 and $\mathbf{X}$, we generate different z-stacks of images $\mathbf{Y}_{t_n} \in \mathbb{R}^{104/L \times 104/L}$ and the corresponding ground truths $\mathbf{X}_{t_n} \in \mathbb{R}^{41 \times 104 \times 104}$, with L being the downsampling factor used. For each z-stack, we have 7 sequences of images $\mathbf{Y}_{t_n}$ and ground truths $\mathbf{X}_{t_n}$, one for each z-section. In particular, we have two datasets:

- Dataset 1 contains z-stacks with no constraints on the emitters' positions in the same image;
- Dataset 2 contains z-stacks with one constraint: emitters in the same image must be more than 400 nm apart.

For each dataset, the same ground truths were used to create z-stacks with downsampling factors of 2, 4, and 8. Furthermore, each frame contains between 1 and 3 emitters. Dataset 1 is more difficult than dataset 2 due to the much higher quantity of unresolved emitters. In Fig 4.10, Fig 4.11, Fig 4.12, and Fig 4.13, we show $\mathbf{Y}_0, \mathbf{X}_0, \mathbf{Y}_1$, and $\mathbf{X}_1$ from the movie capturing the section z-4 in dataset 1; in Fig 4.14 and Fig 4.15, we show $\mathbf{X}_0$ and $\mathbf{Y}_0$ from the movie capturing the section z-4 in dataset 2. In Table 4.1, we show parameter settings shared by both datasets.

L	Pixel size	Image size	σ	μ
2	40 nm	52×52	6.25	77.5
4	80 nm	26×26	23	300
8	160 nm	13×13	75	1000

Table 4.1: Image characteristics

In Table 4.1, we have:

- L is the downsampling factor used for the images $\mathbf{Y}_{t_n}$;
- Pixel size (px, py) is the size of a pixel in nanometers after downsampling;
- Image size is the size of the images $\mathbf{Y}_{t_n}$ in pixels;

- σ is the standard deviation of the Gaussian noise $\mathbf{N}_{t_n}$ present in the simulated images;
- μ is the intensity of the constant uniform background $\mathbf{B}_{t_n}$ in the images $\mathbf{Y}_{t_n}$.

As for the ground truth $\mathbf{X}_{t_n}$ of $\mathbf{Y}_{t_n}$, for any downsampling factor we have $\mathbf{X}_{t_n} \in \mathbb{R}^{41\times104\times104}$. Meanwhile, the values σ and μ have been estimated so that images with a single emitter have, on average, a spot SBR of 0.95 and a spot SNR of 11, which are the mean values of the spot SBR and SNR across 100 emitters in real images from the microscope we are trying to simulate with the image generator. For the images with multiple emitters, the same μ and σ as images with a single emitter have been used.

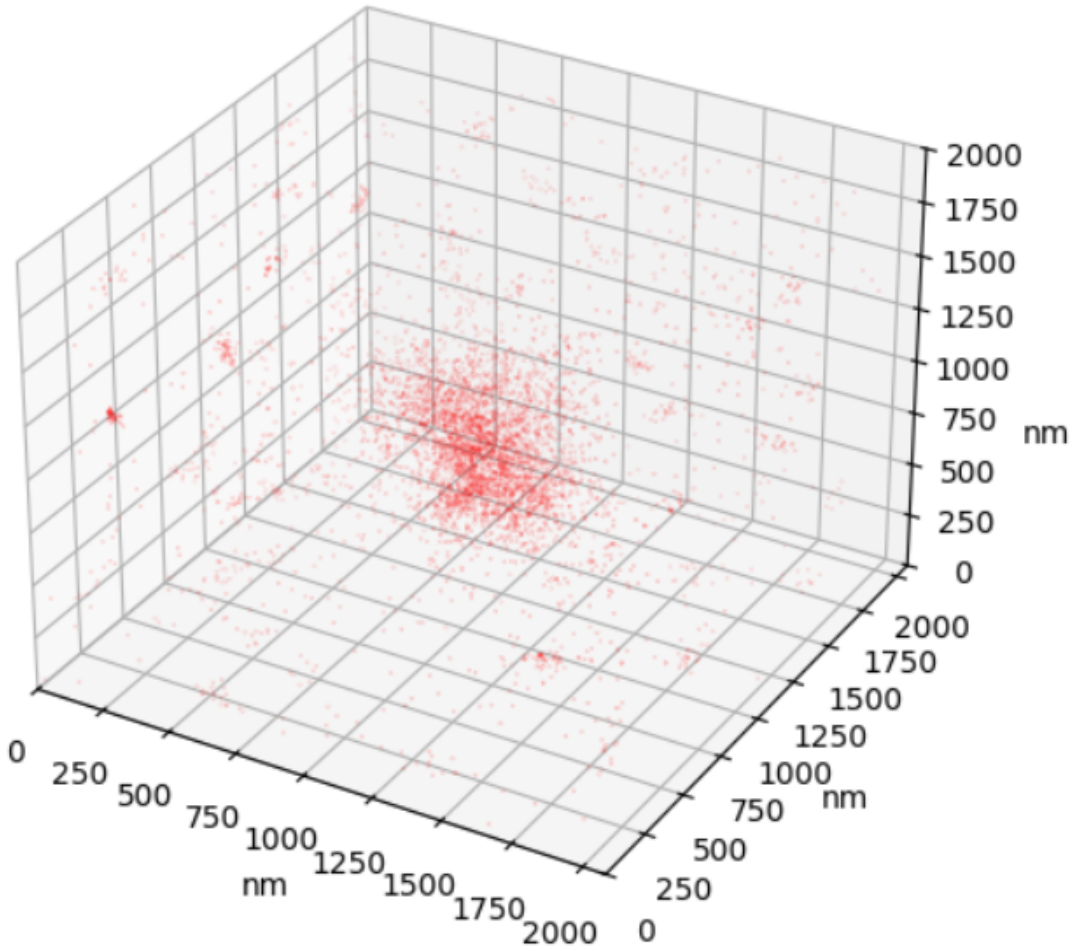
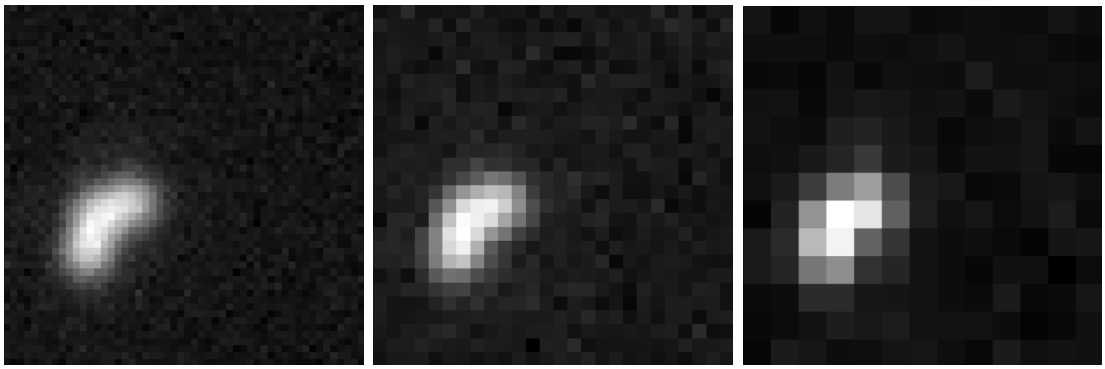


Figure 4.9: *Point cloud* $\boldsymbol{X}$



(a) *L=2* (b) *L=4* (c) *L=8*

Figure 4.10: $\boldsymbol{Y}_0$ *extracted from the movie of section z-4 of dataset 1 with different downsampling factors*

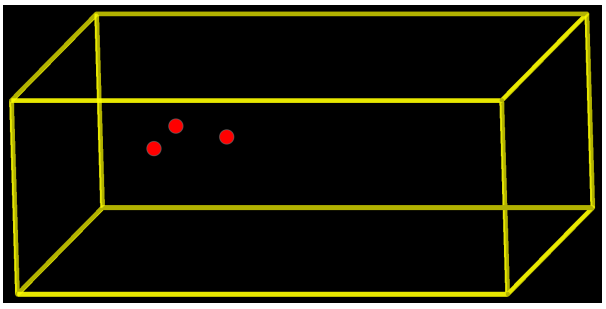
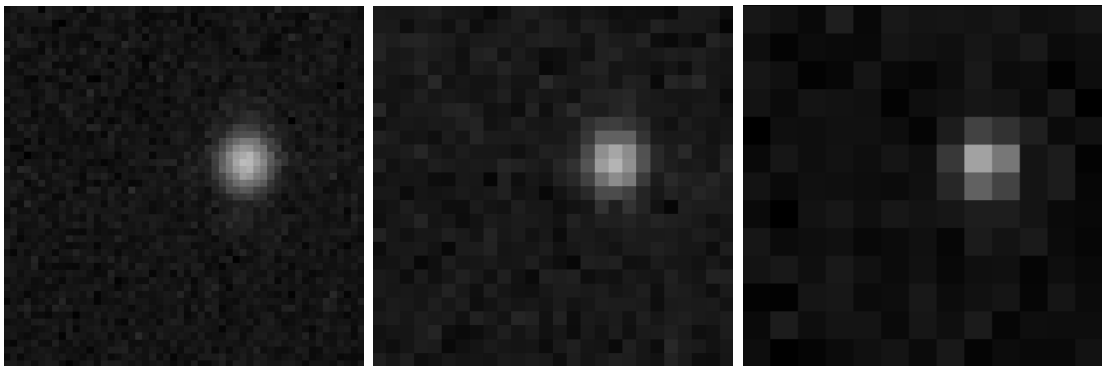


Figure 4.11: $\boldsymbol{X}_0$ *extracted from the movie of section z-4 of dataset 1*



(a) *L=2* (b) *L=4* (c) *L=8*

Figure 4.12: $\boldsymbol{Y}_1$ *extracted from the movie of section z-4 of dataset 1 with different downsampling factors*

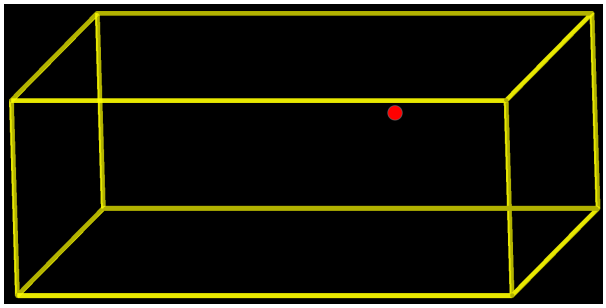
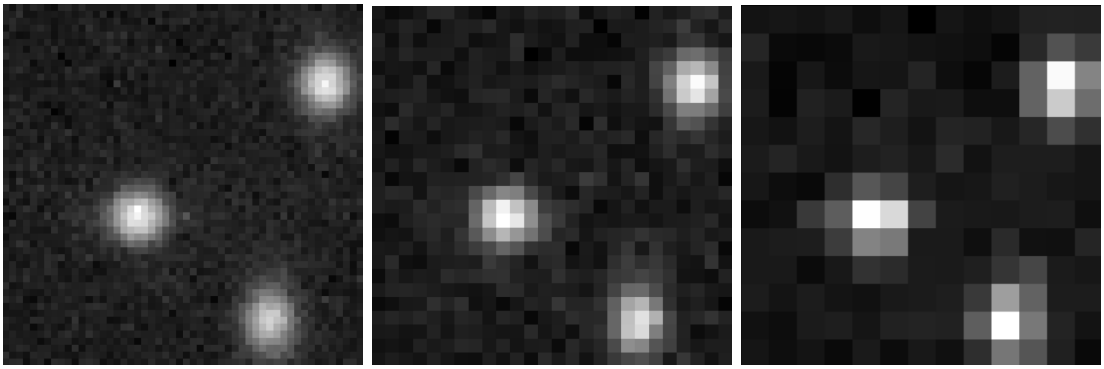


Figure 4.13: $\boldsymbol{X}_1$ *extracted from the movie of section z-4 of dataset 1*



(a) *L=2* **(b)** *L=4* **(c)** *L=8*

Figure 4.14: $\boldsymbol{Y}_0$ *extracted from the movie of section z-4 of dataset 2 with different downsampling factors*

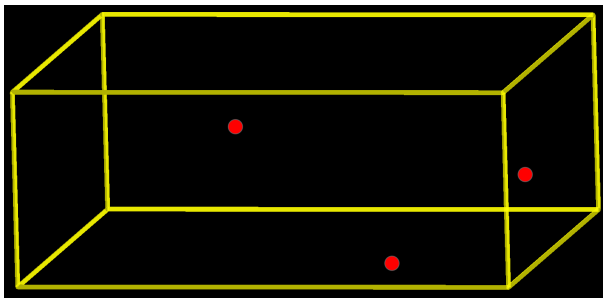


Figure 4.15: $\boldsymbol{X}_0$ *extracted from the movie of section z-4 of dataset 2*

4.5 Numerical results

We have implemented the Algorithm 6 SMLM-PGM-R with four different regularizers. To evaluate the performance of the algorithms used, matches between localizations and emitters from the ground truth are used. We will use the following matching criteria: the maximum matching distance between one localization and one ground truth emitter is $\delta_{xy} = 30$ nm along the xy-plane and $\delta_z = 60$ nm along the z-direction. We have a much larger tolerance in the z-direction because the precision of 3D SMLM techniques for computing the z-position is lower than that for the x and y positions. This is because the z-position is encoded in the image by engineered PSFs, so we need to determine it indirectly from the shape of the emitters' PSF.

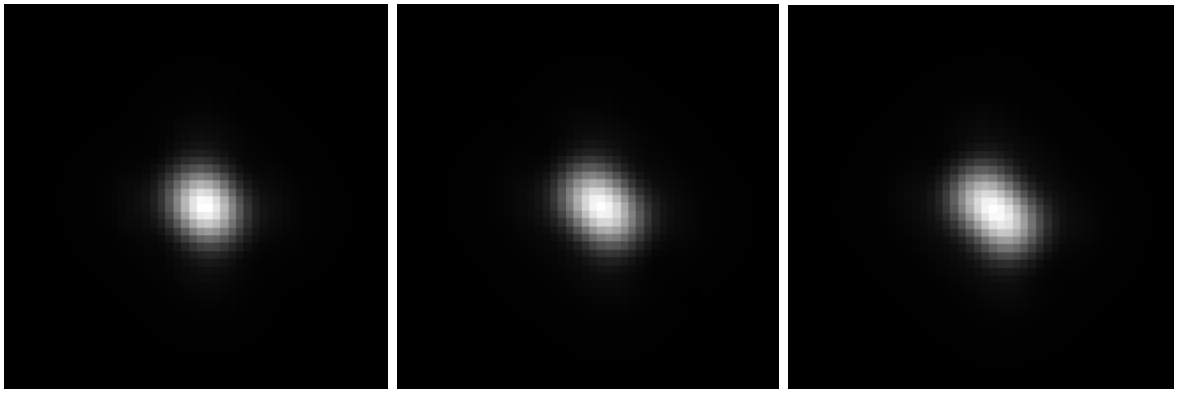
4.5.1 Overlap estimation

The point cloud is dense in the center, while the points are more spread out on the outside. Thus, when we activate some fluorophores, they are more likely to be close to each other, leading to localization issues. Since we have the ground truth for each frame in this case, we can compute the percentage of fluorophores that, when active, are close to each other. This metric depends entirely on the distance used to establish when two emitters are too close. For its computation, we have used distances of 120, 160, and 200 nm along the xy plane.

z-section	Overlap 120 nm	Overlap 160 nm	Overlap 200 nm
1	1.57%	1.57%	2.35%
2	4.92	6.15%	8.62%
3	4.91	8.41%	14.27%
4	9.50%	16.20%	23.09%
5	10.87%	18.38%	25.69
6	3.61%	4.81%	6.63%
7	2.29%	2.29%	4.57%

Table 4.2: Estimation of the overlap

As seen in Table 4.2, in the central z-planes, emitters are closer to each other than in the other planes. Therefore, we expect to obtain worse results there. In Fig. 4.16, we show two emitters at different distances from each other. As we can see, it is very hard to distinguish them.



(a) *Distance of 120 nm* **(b)** *Distance of 160 nm* **(c)** *Distance of 200 nm*

Figure 4.16: *Two emitters at different distances from each other*

4.5.2 Computational costs

To evaluate the computational costs of the various algorithms we tested, we computed the time required to analyze the first 100 images $\mathbf{Y}_{t_n}$ from the movie of the section z-4 from dataset 2 with a downsampling factor $L = 2$, using the SMLM-PGM-CEL0 algorithm with a maximum of 500 iterations. For these experiments, we used a node of the Franklin HPC cluster at IIT. Each node is equipped with an AMD EPYC 7713 containing 512 GB of RAM and 128 cores. With 128 cores, 128 images can be analyzed in parallel. Thus, we can examine the computational costs of both a parallel and a non-parallel version of the SMLM-PGM-CEL0 algorithm. For parallelization, we use the functions *apply_ async()*, *set_ start_ method()*, and *Pool()* from the Python library *multiprocessing*.

For computing the time required, *perf_ counter()* from the Python library *time* was used. For now, we have only managed to parallelize the algorithm that does not use the ROI approach, since the ROI approach has a more complex structure requiring a different parallelization strategy. We present the result obtained in Table 4.3.

As we can see:

- The ROI approach is more than 3 times faster. This is because we apply the SMLM-PGM-CEL0 algorithm to only a few patches of the image, rather than the whole image. In addition, our images have few emitters;

- The backtracking strategy for the step size has higher computational costs than the constant step size strategy. This may happen because backtracking leads to faster convergence for a low number of frames. Furthermore, to check the sufficient

decrease condition for backtracking, we need to perform more computations in each iteration than with a constant step size strategy, which increases the time cost;

- By parallelizing the code, we halve the time required to obtain the results.

ROI	Backtracking	Parallelization	CPU time (seconds)
No	No	No	3917.75
No	Yes	No	4406.98
No	No	Yes	1711.44
No	Yes	Yes	2194.84
Yes	No	No	1031.35
Yes	Yes	No	1236.93

Table 4.3: Computational costs of SMLM-PGM-CEL0 for 100 images. Top panel without ROI, bottom panel with.

Meanwhile, ThunderSTORM takes less than a second to analyze 100 images on a computer with an Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz (1.50 GHz) and 8 Gb of RAM. Thus, SMLM-PGM-CEL0 is much more computationally expensive than ThunderSTORM.

4.5.3 Result analysis with dataset 1

We now evaluate Algorithm 6 SMLM-PGM-CEL0 using the images from dataset 1. In particular, we will analyze the first 100 out of 844 images $\mathbf{Y}_{t_n}$ from the movie of section z-4, which captures the central part of the point cloud, so we have $t_n \in [0, \ldots, 99]$. We chose this number of images because analyzing 100 images $\mathbf{Y}_{t_n}$ is already computationally expensive, as seen in Table 4.3. We will only analyze the case $L = 2$. For all results obtained using backtracking, the maximum number of iterations per backtracking is 15. In all the tables below, L is the downsampling factor used for the images, λ is the regularization parameter used for the PGM, maxit is the maximum number of iterations for the PGM, and Jaccard is the Jaccard index computed using the localized emitters.

In Table 4.4, we show the results obtained using the SMLM-PGM-CEL0 algorithm with a constant step size.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
3	2	500	72	77	122	26.57%	7.82 nm	37.14 nm	25.20%
3	2	1000	75	80	119	27.37%	6.93 nm	37.81 nm	25.99%

Table 4.4: Results obtained using SMLM-PGM-CEL0 and constant step size $\tau = \frac{1}{L_{est}}$

In Table 4.5, we show the results obtained using the SMLM-PGM-CEL0 algorithm with a backtracking strategy. For backtracking, we use $\rho = 0.8$ and an initial step size $\tau = \frac{8}{L_{est}}$, where L_{est} is determined using (3.14).

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
3	2	250	86	48	108	35.54%	8.07 nm	34.78 nm	34.13%
3	2	500	84	50	110	34.43%	8.16 nm	33.15 nm	33.14%

Table 4.5: Results obtained with SMLM-PGM-CEL0 and backtracking

In Table 4.6, we present the results obtained using the SMLM-PGM-CEL0 algorithm with the ROI approach and a constant step size.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
0.11	2	500	76	63	118	29.57%	6.88 nm	36.31 nm	28.25%
0.11	2	1000	90	86	104	32.14%	8.69 nm	39.03 nm	30.48%

Table 4.6: Results obtained using SMLM-PGM-CEL0 with ROI approach and constant step size $\tau = \frac{1}{L_{est}}$

In Table 4.7, we present the results obtained using the SMLM-PGM-CEL0 algorithm with the ROI approach and a backtracking strategy. We use $\tau = \frac{3.5}{L_{est}}$, $\rho = 0.9$ for backtracking.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
0.11	2	250	91	48	103	37.60%	7.56 nm	39.25 nm	35.86%
0.11	2	500	96	43	98	40.51%	7.36 nm	37.97 nm	38.79%

Table 4.7: Results obtained using SMLM-PGM-CEL0 with ROI approach and backtracking

Comments to the results:

- We do not achieve high Jaccard indices and efficiency percentages. This is because this dataset contains many overlapping emitters, which are hard to distinguish and localize;
- Increasing the maximum number of allowed iterations almost always results in better results. With a higher maximum number of iterations, the reconstructions have more time to approach the ground truths;
- With the backtracking strategy, results always improve. The backtracking strategy improves the convergence speed, meaning we need fewer iterations for the reconstructions to get closer to the ground truths;
- Results obtained by analyzing ROI instead of the whole image are always better than the results without them.

4.5.4 Result analysis with dataset 2

We now analyze the first 100 images $\mathbf{Y}_{t_n}$ out of 978 from the movie of section z-4 of dataset 2, capturing the central part of the point cloud, so we have $t_n \in [0, \dots, 99]$. For all the results using backtracking, the maximum number of iterations per backtracking is 15.

In table 4.8, we present the results obtained using SMLM-PGM-CEL0 with a constant step size.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1.98	2	500	137	64	64	51.70%	3.42 nm	32.78 nm	50.27%
1.98	2	1000	150	51	51	59.52%	4.90 nm	33.64 nm	57.67%
0.72	4	500	133	70	68	49.08%	3.47 nm	34.91 nm	47.55%
0.72	4	1000	141	90	60	48.45 %	5.59 nm	31.48 nm	47.12%
0.10	8	500	124	77	77	44.60%	9.84 nm	36.62 nm	42.69%
0.10	8	1000	124	71	77	45.59%	8.23 nm	34.79 nm	43.91%

Table 4.8: Results obtained using SMLM-PGM-CEL0 with costant step size $\tau = \frac{1}{L_{est}}$

In table 4.9, we present the results obtained using SMLM-PGM-CEL0 with backtracking. For $L = 2$, we use $\rho = 0.9$, $\tau = \frac{5}{L_{est}}$; for $L = 4$, we use $\rho = 0.8$, $\tau = \frac{7}{L_{est}}$; for $L = 8$, we use $\rho = 0.9$, $\tau = \frac{3}{L_{est}}$.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1.98	2	250	163	38	38	68.20%	6.46 nm	29.22 nm	66.25%
1.98	2	500	165	37	36	69.33%	6.42 nm	28.07 nm	67.44%
0.72	4	250	164	38	37	68.62%	7.65 nm	28.20 nm	66.63%
0.72	4	500	163	39	38	67.92%	7.67 nm	26.91 nm	66.10%
0.10	8	250	156	45	45	63.41%	10.0 nm	31.81 nm	61.08%
0.10	8	500	159	44	42	64.90%	10.16 nm	30.84 nm	62.55%

Table 4.9: Results obtained using SMLM-PGM-CEL0 with backtracking

In table 4.10, we present the results obtained using SMLM-PGM-CEL0 with the ROI approach and a constant step size.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
0.357	2	500	133	64	68	50.19%	5.75 nm	33.45 nm	48.64%
0.357	2	1000	134	63	67	50.76%	3.86 nm	32.48 nm	49.36%
0.11	4	500	121	68	80	44.98%	4.07 nm	34.77 nm	43.55%
0.11	4	1000	125	64	76	47.17%	4.38 nm	33.91 nm	45.74%
0.021	8	500	122	73	79	44.53%	8.49 nm	35.40 nm	42.81%
0.021	8	1000	124	71	77	45.59%	8.23 nm	34.79 nm	43.91%

Table 4.10: Results obtained using SMLM-PGM-CEL0 with ROI approach and constant step size $\tau = \frac{1}{L_{est}}$

In Table 4.11, we present the results obtained using SMLM-PGM-CEL0 with ROI approach and backtracking. When we use backtracking for $L = 2$, we use $\tau = \frac{4}{L_{est}}$, $\rho = 0.9$; for $L = 4$, we use $\tau = \frac{4}{L_{est}}$, $\rho = 0.9$; for $L = 8$, we use $\tau = \frac{2.5}{L_{est}}$, $\rho = 0.9$.

λ	L	maxit	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
0.357	2	250	144	59	57	55.38%	5.27 nm	32.07 nm	53.82%
0.357	2	500	148	55	53	57.81%	5.20 nm	31.35 nm	56.23%
0.11	4	250	137	59	64	52.69%	3.82 nm	32.43 nm	51.25%
0.11	4	500	140	56	61	54.47%	3.78 nm	31.03 nm	53.10 %
0.021	8	250	136	59	65	52.31%	8.74 nm	33.30 nm	50.49%
0.021	8	500	140	55	61	54.69%	8.62 nm	32.73 nm	52.84%

Table 4.11: Results obtained using SMLM-PGM-CEL0 with ROI and backtracking

As we can see:

- As the downsampling factor L increases, the results obtained worsen. As L increases, the problem becomes more ill-posed and thus harder to solve;
- With the backtracking strategy, results always improve;
- In almost all cases, increasing maxit leads to better results;
- The results with the ROI approach are almost always worse than those obtained without it. This may be because, in some cases, emitters are so close to each other that a ROI contains one emitter and part of another, which can cause issues.

In Table 4.12, we present the results obtained from the first 100 images $\mathbf{Y}_{t_n}$ of each sequence of images from dataset 2, capturing all z-sections of the point cloud, with a downsampling factor $L = 8$ and $\lambda = 0.10$. The method used is SMLM-PGM-CEL0 with maxit=500, and backtracking with step size $\tau = \frac{3}{L_{est}}$ and $\rho = 0.9$.

z-section	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1	149	56	56	57.09%	8.82 nm	32.96 nm	55.10%
2	133	60	60	52.57%	11.76 nm	31.95 nm	50.53%
3	112	71	70	44.27%	9.63 nm	33.86 nm	42.59%
4	159	44	42	64.90%	10.16 nm	30.81 nm	62.55%
5	139	54	52	56.73 %	9.74 nm	31.26 nm	54.82%
6	151	50	48	60.64%	8.91 nm	32.86 nm	58.48%
7	132	62	62	51.5625%	8.16 nm	35.11 nm	49.67%
All	975	397	390	55.33%	9.65 nm	32.64 nm	53.36%

Table 4.12: Results obtained using SMLM-PGM-CEL0 with backtracking

As shown in Table 4.12, the central z-section has the best Jaccard index and efficiency. The cause of this behavior may be:

- The λ was optimized for the fourth plane. For other planes, other λ might be better;
- In the central section, we have a higher percentage of emitters near the focus plane than in other z-sections. Emitters closer to the focal plane are easier to localize than distant emitters for SMLM-PGM.

All these tests were conducted during the internship at IIT. Due to time constraints and the high computational costs of the algorithms involved, we tested only one regularizer extensively: the CEL0 penalty function.

In the 3D plots in Fig 4.17, Fig 4.18, and Fig 4.19, the x and y axes represent pixels, and the z axis represents height. To convert it into nanometers, the following formulas can be used:

$$x_{nm} = x * 20 - 10, \qquad y_{nm} = y * 20 - 10, \qquad z_{nm} = z * 10 - 200$$

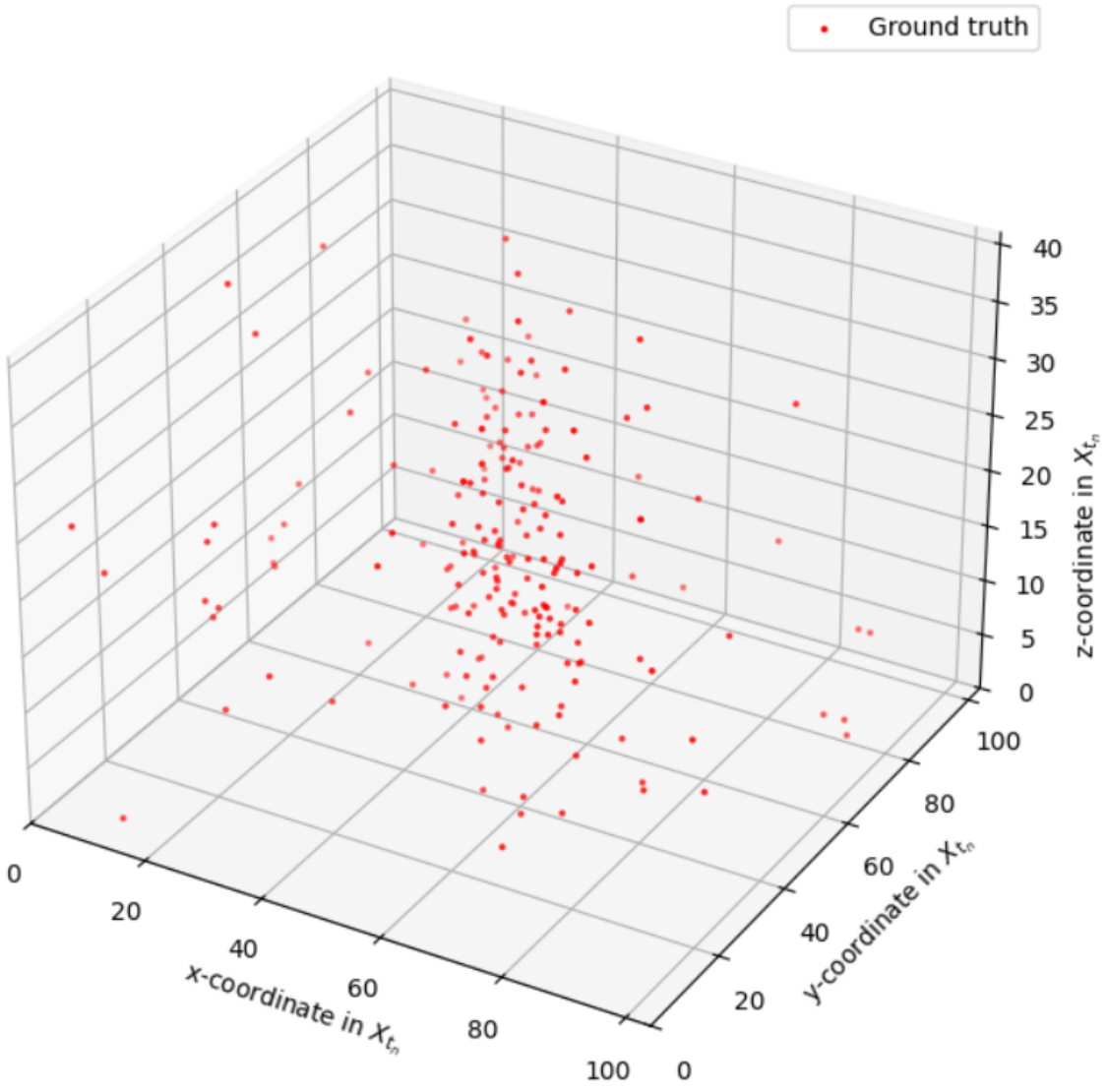


Figure 4.17: *Plot of the ground truth* $\boldsymbol{X}_{t_n}$ *for* $t_n \in [0, 99]$

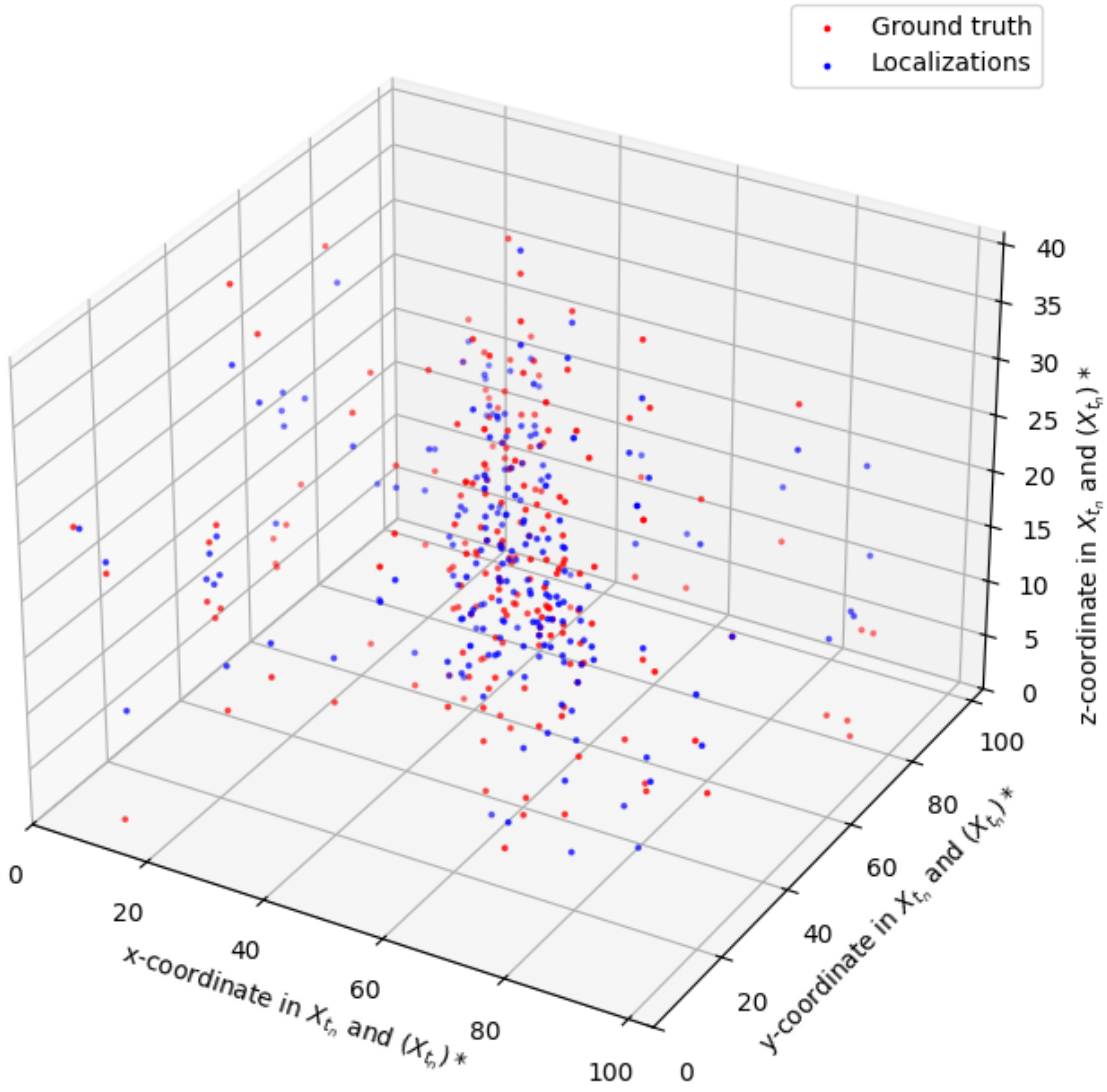


Figure 4.18: *Plot of ground truth* $\boldsymbol{X}_{t_n}$ *in red and reconstructed localizations* $(\boldsymbol{X}_{t_n})^*$ *in blue for* $t_n \in [0, 99]$ *from the movie of the section z-4 from dataset 2 with* $L = 2$ *using SMLM-PGM-CEL0 with maxit=500 and backtracking strategy*

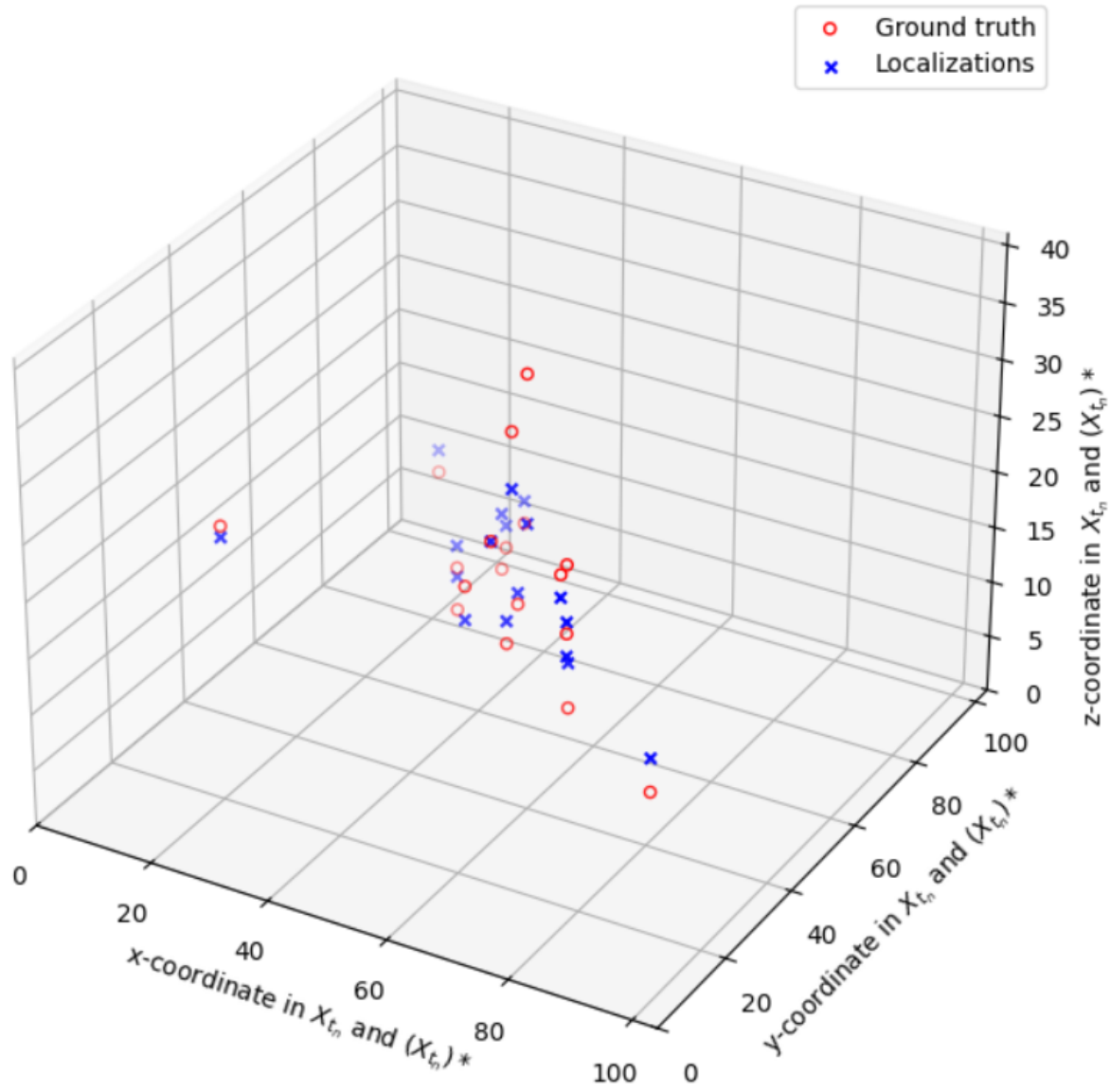


Figure 4.19: *Plot of ground truth $\boldsymbol{X}_{t_n}$ in red and reconstructed localizations $(\boldsymbol{X}_{t_n})^*$ in blue for $t_n \in [30, 39]$ from the movie of the section z-4 from dataset 2 with $L = 2$ using SMLM-PGM-CEL0 with maxit=500 and backtracking strategy*

4.5.5 Comparison with ThunderSTORM

We also used ThunderSTORM to localize the emitters in our movies. For ThunderSTORM, we used a matching distance δ between localizations and ground truth positions of 30 nm along the xy-plane and 60 nm along the z-direction. For matching, the ground truth points are on a grid, with each grid point at the center of a pixel. We do this because the localizations computed by ThunderSTORM are not on a grid. In Table 4.13,

the 'Dataset' column represents which dataset was utilized. Furthermore, least squares fitting was used because the noise in the analyzed images is Gaussian. If it were Poisson, maximum likelihood estimation would have been more accurate.

Dataset	L	Radius	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1	2	10	56	85	138	20.07%	15.00 nm	24.88 nm	18.89%
2	2	8	116	47	85	46.77%	15.04 nm	26.01 nm	44.95%
2	4	4	100	62	101	38.02%	15.38 nm	24.28 nm	36.49%
2	8	2	83	48	118	33.33%	17.30 nm	31.65 nm	31.29%

Table 4.13: Results obtained with ThunderSTORM

Dataset	L	λ	maxiter	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1	2	3	250	86	48	108	35.54%	8.07 nm	34.78 nm	34.13%
2	2	1.98	500	165	37	36	69.33%	6.41 nm	28.07 nm	67.44%
2	4	0.72	250	164	38	37	68.62%	7.65 nm	28.20 nm	66.63%
2	8	0.10	500	159	44	42	64.90%	10.16 nm	30.84 nm	62.55%

Table 4.14: Best results obtained with SMLM-PGM-CEL0 without using ROI

Dataset	L	λ	maxiter	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1	2	0.11	500	96	43	98	40.51%	7.36 nm	39.25 nm	38.79%
2	2	0.357	500	148	55	53	57.81%	5.20 nm	31.35 nm	56.23%
2	4	0.11	500	140	56	61	54.47%	3.78 nm	31.03 nm	53.10%
2	8	0.021	500	140	55	61	54.69%	8.62 nm	32.73 nm	52.84%

Table 4.15: Best results obtained with SMLM-PGM-CEL0 using ROI

As shown in Table 4.13, Table 4.14, and Table 4.15, for each downsampling factor L, our localization algorithm outperforms ThunderSTORM in terms of the Jaccard index and efficiency metrics. This difference is particularly significant for the case $L = 8$, which has a pixel size very close to that of real images. However, this performance increase comes with a high computational cost compared to ThunderSTORM, as shown in Table 4.3.

When we analyze the first 100 images from each movie in dataset 2 with a downsampling factor of 8, we obtain the results in Table 4.16. For these results, the fitting radius parameter was set to 2 pixels.

z-section	TP	FP	FN	Jaccard	RMSE lat	RMSE ax	Efficiency
1	17	47	188	6.75%	19.77 nm	30.87 nm	5.07%
2	50	58	143	19.92%	17.40 nm	27.17 nm	18.41%
3	69	61	113	28.40%	16.66 nm	31.10 nm	26.60%
4	83	48	118	33.33%	17.54 nm	31.43 nm	31.29%
5	58	43	133	23.67%	15.54 nm	30.45 nm	22.14%
6	41	56	158	16.08%	18.03 nm	28.39 nm	14.52%
7	36	56	158	14.40%	15.66 nm	32.73 nm	12.91%
All	354	380	1011	20.29%	17.03 nm	30.40 nm	18.67%

Table 4.16: Results obtained using ThunderSTORM

Compared to the results obtained with SMLM-PGM-CEL0 in Table 4.12, the Jaccard index and efficiency are much lower for all z-sections. However, the better results from SMLM-PGM-CEL0 come with a high computational cost compared to ThunderSTORM.

Conclusions

In this thesis, we presented a mathematical model describing the image formation process in STORM imaging. Based on this model, we formulated an inverse problem to recover the emitters' coordinates and investigated algorithms for solving it.

As shown in Chapter 4, the tested algorithm exhibits both strengths and limitations. In particular, SMLM-PGM-CEL0 is significantly slower than ThunderSTORM, but localizes substantially more emitters. To further improve the results, several directions could be explored. For instance, the other regularizers presented in Chapter 3 could be tested, or a Deep Unrolling approach could be implemented to estimate the optimal regularization parameter λ for each image. In addition, the matching procedure between ground truth emitters and localizations could be improved using algorithms such as the Gale–Shapley algorithm. Nevertheless, even with these improvements, the computational cost would likely remain high.

To address this limitation, machine learning techniques could be combined with the PGM framework. One possible approach is the use of Plug-and-Play (PnP) methods, in which the proximal operator in the backward step of the algorithm is replaced with a neural network. In our case, the network would be trained specifically to localize emitters.

Furthermore, in Chapter 1, we introduced two imaging modalities beyond astigmatic imaging: biplane and double-helix imaging. So far, our experiments have focused on an astigmatic PSF. It would therefore be valuable to evaluate the performance of our algorithms when applied to other PSF models, such as double-helix or biplane PSFs. For biplane imaging, the algorithm would need to be adapted, since two images are acquired at each time t_n.

In conclusion, this thesis highlights the potential of optimization-based approaches, such as SMLM-PGM-CEL0, for accurate emitter localization in STORM imaging. While the method can localize more emitters, its computational cost remains a significant limitation. Future research that combines numerical optimization techniques with machine learning methods, such as PnP methods, may offer a promising direction for reducing

computational cost while maintaining or even improving localization accuracy.

Bibliography

[1] Beliveau BJ, Boettiger AN, Nir G, Bintu B, Yin P, Zhuang X, Wu CT. In Situ Super-Resolution Imaging of Genomic DNA with OligoSTORM and OligoDNA-PAINT. Methods Mol Biol. 2017;1663:231-252. doi: 10.1007/978-1-4939-7265-4_19. PMID: 28924672; PMCID: PMC5919218.

[2] https://ibidi.com/content/215-widefield-fluorescence

[3] https://andor.oxinst.com/learning/view/article/microscopy-dictionary-a-to-z-glossary#r

[4] Hell SW, Wichmann J. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. Opt Lett. 1994 Jun 1;19(11):780-2. doi: 10.1364/ol.19.000780. PMID: 19844443.

[5] Rust MJ, Bates M, Zhuang X. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM). Nat Methods. 2006 Oct;3(10):793-5. doi: 10.1038/nmeth929. Epub 2006 Aug 9. PMID: 16896339; PMCID: PMC2700296.

[6] Betzig E, Patterson GH, Sougrat R, Lindwasser OW, Olenych S, Bonifacino JS, Davidson MW, Lippincott-Schwartz J, Hess HF. Imaging intracellular fluorescent proteins at nanometer resolution. Science. 2006 Sep 15;313(5793):1642-5. doi: 10.1126/science.1127344. Epub 2006 Aug 10. PMID: 16902090.

[7] https://www.microscope.healthcare.nikon.com/it_EU/products/super-resolution-microscopes/n-storm-super-resolution/the-principle-of-stochastic-optical-reconstruction-microscopy

[8] Bates M, Huang B, Dempsey GT, Zhuang X. Multicolor super-resolution imaging with photo-switchable fluorescent probes. Science. 2007 Sep 21;317(5845):1749-53. doi: 10.1126/science.1146598. Epub 2007 Aug 16. PMID: 17702910; PMCID: PMC2633025.

[9] Nir G, Farabella I, Pérez Estrada C, Ebeling CG, Beliveau BJ, Sasaki HM, et al. (2018) Walking along chromosomes with super-resolution imaging, contact maps, and integrative modeling. PLoS Genet 14(12): e1007872. https://doi.org/10.1371/journal.pgen.1007872

[10] Rosa, S., & Shaw, P. (2013). Insights into Chromatin Structure and Dynamics in Plants. Biology, 2(4), 1378-1410. https://doi.org/10.3390/biology2041378

[11] OVESNÝ, Martin. Computational methods in single molecule localization microscopy. Dizertační práce, vedoucí Hagen, Guy Michael. Praha: Univerzita Karlova, 1. lékařská fakulta, Ústav buněčné biologie a patologie 1. LF UK, 2016.

[12] von Diezmann L, Shechtman Y, Moerner WE. Three-Dimensional Localization of Single Molecules for Super-Resolution Imaging and Single-Particle Tracking. Chem Rev. 2017 Jun 14;117(11):7244-7275. doi: 10.1021/acs.chemrev.6b00629. Epub 2017 Feb 2. PMID: 28151646; PMCID: PMC5471132.

[13] Bo Huang et al. ,Three-Dimensional Super-Resolution Imaging by Stochastic Optical Reconstruction Microscopy.Science319,810-813(2008).DOI:10.1126/science.1153529

[14] Juette, M., Gould, T., Lessard, M. et al. Three-dimensional sub–100 nm resolution fluorescence microscopy of thick samples. Nat Methods 5, 527–529 (2008). https://doi.org/10.1038/nmeth.1211

[15] S.R.P. Pavani,M.A. Thompson,J.S. Biteen,S.J. Lord,N. Liu,R.J. Twieg,R. Piestun, & W.E. Moerner, Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function, Proc. Natl. Acad. Sci. U.S.A. 106 (9) 2995-2999, https://doi.org/10.1073/pnas.0900245106 (2009).

[16] OVESNÝ, Martin. Computational methods in single molecule localization microscopy. Dizertační práce, vedoucí Hagen, Guy Michael. Praha: Univerzita Karlova, 1. lékařská fakulta, Ústav buněčné biologie a patologie 1. LF UK, 2016.

[17] von Diezmann L, Shechtman Y, Moerner WE. Three-Dimensional Localization of Single Molecules for Super-Resolution Imaging and Single-Particle Tracking. Chem Rev. 2017 Jun 14;117(11):7244-7275. doi: 10.1021/acs.chemrev.6b00629. Epub 2017 Feb 2. PMID: 28151646; PMCID: PMC5471132.

[18] Thompson RE, Larson DR, Webb WW. Precise nanometer localization analysis for individual fluorescent probes. Biophys J. 2002 May;82(5):2775-83. doi: 10.1016/S0006-3495(02)75618-X. PMID: 11964263; PMCID: PMC1302065.

[19] Babcock, H.P., Zhuang, X. Analyzing Single Molecule Localization Microscopy Data Using Cubic Splines. Sci Rep 7, 552 (2017). https://doi.org/10.1038/s41598-017-00622-w

[20] Sage D, Donati L, Soulez F, Fortun D, Schmit G, Seitz A, Guiet R, Vonesch C, Unser M. DeconvolutionLab2: An open-source software for deconvolution microscopy. Methods. 2017 Feb 15;115:28-41. doi: 10.1016/j.ymeth.2016.12.015. Epub 2017 Jan 3. PMID: 28057586.

[21] Sternberg, "Biomedical Image Processing," in Computer, vol. 16, no. 1, pp. 22-34, Jan. 1983, doi: 10.1109/MC.1983.1654163.

[22] T. Huang, G. Yang and G. Tang, "A fast two-dimensional median filtering algorithm," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 27, no. 1, pp. 13-18, February 1979, doi: 10.1109/TASSP.1979.1163188.

[23] Soubies, E., Blanc-Féraud, L., Aubert, G.: A continuous exact ℓ_0 penalty (CEL0) for least squares regularized problem. SIAM Journal on Imaging Sciences 8(3), 1607-1639 (2015)

[24] Mhamed Essafri & Luca Calatroni & Emmanuel Soubies, 2025. "Exact continuous relaxations of ℓ_0 -regularized criteria with non-quadratic data terms," Journal of Global Optimization, Springer, vol. 93(3), pages 651-699, November.

[25] Yang, L., Morigi, S., Ng, M. K., & Wen, Y. (2025). Truncated Huber Penalty for Sparse Signal Recovery with Convergence Analysis. ArXiv. https://arxiv.org/abs/2504.04509

[26] Combettes, P.L. and Wajs, V.R. (2005) Signal Recovery by Proximal Forward-Backward Splitting. Multiscale Modeling & Simulation, 4, 1168-1200.

[27] Ovesný M, Křížek P, Borkovec J, Svindrych Z, Hagen GM. ThunderSTORM: a comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging. Bioinformatics. 2014 Aug 15;30(16):2389-90. doi: 10.1093/bioinformatics/btu202. Epub 2014 Apr 25. PMID: 24771516; PMCID: PMC4207427.

[28] Babcock, H., Sigal, Y.M. & Zhuang, X. A high-density 3D localization algorithm for stochastic optical reconstruction microscopy. Opt Nano 1, 6 (2012). https://doi.org/10.1186/2192-2853-1-6

[29] Speiser A, Müller LR, Hoess P, Matti U, Obara CJ, Legant WR, Kreshuk A, Macke JH, Ries J, Turaga SC. Deep learning enables fast and dense single-molecule localization with high accuracy. Nat Methods. 2021 Sep;18(9):1082-1090. doi: 10.1038/s41592-021-01236-x. Epub 2021 Sep 3. Erratum in: Nat Methods. 2021 Nov;18(11):1410. doi: 10.1038/s41592-021-01305-1. PMID: 34480155; PMCID: PMC7611669.

[30] Zunino, A., Slenders, E., Fersini, F. et al. Open-source tools enable accessible and advanced image scanning microscopy data analysis. Nat. Photon. (2023). https://doi.org/10.1038/s41566-023-01216-x

[31] Liu, Y., Stergiopoulou, V., Chuah, J., Bezzam, E., Both, G.-J., Unser, M., Sage, D., & Dong, J. (2025). Revisiting PSF models: Unifying framework and high-performance implementation. Journal of Microscopy, 1-13. https://doi.org/10.1111/jmi.70045

[32] Su JH, Zheng P, Kinrot SS, Bintu B, Zhuang X. Genome-Scale Imaging of the 3D Organization and Transcriptional Activity of Chromatin. Cell. 2020 Sep 17;182(6):1641-1659.e26. doi: 10.1016/j.cell.2020.07.032. Epub 2020 Aug 20. PMID: 32822575; PMCID: PMC7851072.

[33] Sage, D., Pham, TA., Babcock, H. et al. Super-resolution fight club: assessment of 2D and 3D single-molecule localization microscopy software. Nat Methods 16, 387-395 (2019). https://doi.org/10.1038/s41592-019-0364-4