Universiteit Utrecht

# Exploring the multi-emitter localization problem in high and low density settings

*Tomas Ehrencron*

Mathematical Sciences

Master Thesis

*Supervisors*:

Dr. P. Salanevich
Utrecht University

Dr. I Kryven
Utrecht University

Dr. H. van der Voort
Scientific Volume Imaging

M. Ram MS
Scientific Volume Imaging

Dr. F. van der Have
Scientific Volume Imaging

November 15, 2022

## Abstract

Single Molecule Localization Microscopy is a set of techniques that allows to overcome the diffraction limit and create higher resolution images by processing entire image series and combining the results. Molecules that lie too close to each other to separate, are separated in time instead. The experiments to acquire these images can be tedious and time-consuming. To speed up this process, we can image more molecules per frame, but this will result in overlapping molecules. We explore two methods to solve the issue of overlapping molecules in the image. We present a new method in solving this problem using sparse regularization.

# Contents

## 1 Notation and conventions

Before we start we briefly discuss the notations and conventions used in this thesis. We use the following definitions:

$$\begin{aligned} \mathbb{N} &= \{0, 1, 2, \ldots\} \\ \mathbb{R}_+ &= \{r \in \mathbb{R} \mid r \geq 0\} \\ [x] &= \{1, \ldots, x\} \quad \text{for } x \in \mathbb{N} \setminus \{0\} \end{aligned}$$

For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we define the *gradient* $\nabla f(x)$ as the $n \times 1$ vector

$$\nabla f(x) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{pmatrix}.$$

With some slight abuse of notation, we can consider a subset $X \subseteq (x_1, \ldots, x_n)$ of the set of variables. Then $\nabla_X f(x)$ is the vector containing the partial derivatives of the variables in $X$. If $f$ is twice continuously differentiable, then the *Hessian* matrix $\nabla^2 f(x)$ is the $n \times n$ matrix containing the second partial derivatives:

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2}{\partial x_1^2} f(x) & \ldots & \frac{\partial^2}{\partial x_1 \partial x_n} f(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} f(x) & \ldots & \frac{\partial^2}{\partial x_n^2} f(x) \end{pmatrix}$$

For a function $g : \mathbb{R}^n \to \mathbb{R}^m : g = (g_1, \ldots, g_m)$ where $g_i$ is continuously differentiable for all $i = 1 \ldots, m$, we can define the $m \times n$ *Jacobian* matrix $\mathbb{J}_g$ as the matrix containing the first partial derivatives

$$\mathbb{J}_g = \begin{pmatrix} \frac{\partial}{\partial x_1} g_1(x) & \ldots & \frac{\partial}{\partial x_n} g_1(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} g_m(x) & \ldots & \frac{\partial}{\partial x_n} g_m(x) \end{pmatrix}$$

When it is clear from the context of which function we consider the Jacobian, we simply write $\mathbb{J}$.

### 1.1 Image notation

An image consists of a series of $T$ frames where each frame has width $W$ and height $H$. We can formally define an *image (series)* $I$ as a function $I : [W] \times [H] \times [T] \to \mathbb{R}_+$. A single frame $t$ can then be defined as $I(t) : [W] \times [H] \to \mathbb{R}_+, (w, h) \mapsto I(w, h, t)$. When $T = 1$, we use some abuse of notation and denote an image $I$ as a function $I : [W] \times [H] \to \mathbb{R}_+$ or even $I : [WH] \to \mathbb{R}_+$ where we use the translation $t : [W] \times [H] \to [WH]$ given by $t(i, j) = (j - 1)W + i$. In this last case, we can also view $I$ as vector in $\mathbb{R}_+^{WH}$ or as matrix in $\mathbb{R}_+^{W \times H}$.

Acquisition images measure the light intensity by counting a certain number of electrons (see Section 3.3), so the pixel values are in $\mathbb{N}$. We defined the images in $\mathbb{R}_+$, because we are not only using the raw image data. We also apply operations to these images which results in non-integer intensities. For the specific case where $I : [W] \times [H] \times [T] \to \mathbb{N}$ we define the histogram $\mathcal{H}(I) : \mathbb{N} \to \mathbb{N}$ as

$$\mathcal{H}(I)_m = |\{(w, h, t) \in [W] \times [H] \times [T] \mid I(w, h, t) = m\}|$$

where $m \in \mathbb{N}$. Finally, we define the *neighborhood* of size $\rho$ of a pixel $p = (w, h, t)$, denoted by $\mathcal{N}_\rho(p)$, as

$$\mathcal{N}_\rho(p) = \{(w', h') \in [W] \times [H] \mid |w - w'| \leq \rho, |h - h'| \leq \rho\}.$$

## 2 Introduction and problem description

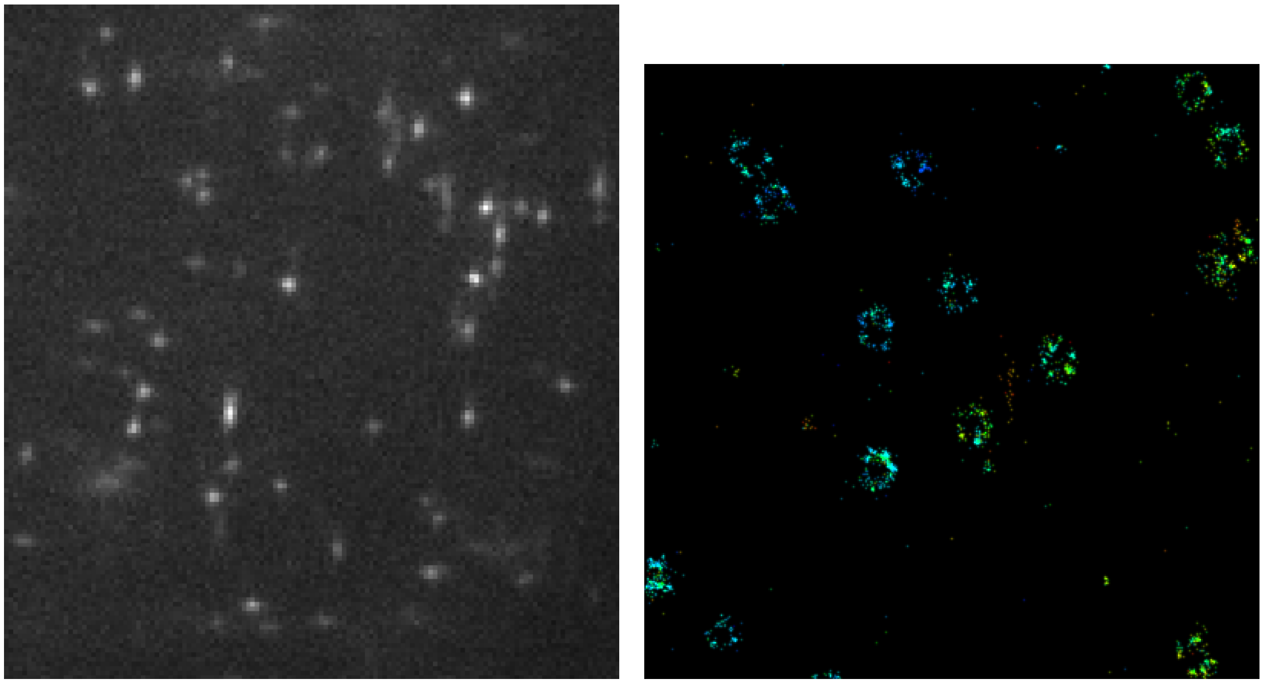
Optical microscopy is limited by a physical boundary called Abbe's diffraction limit [21]. This limit is defined as

$$d = \frac{\lambda}{2n \sin\theta},$$

where $\lambda$ is the wavelength of the light measured, $n$ is the refractive index of the medium, and $\theta$ is the half-angle of the cone of light that can enter the lens. Point sources that lie closer to each other than this distance will overlap, because their interference patterns are overlapping. Since $\sin\theta \leq 1$, and the refractive index of the medium is also bounded (which is often around 1.5), there is an upper bound on $d$ depending on $\lambda$. For microscopes that use light from the visible spectrum, the measured wavelength is also bounded from below, resulting in a maximum resolution of around a 200-300 nm [17].

## 2.1 SMLM

Single-molecule localization microscopy (SMLM) is a set of techniques with the aim of increasing the image resolution above the diffraction limit. The idea behind SMLM is that molecules that lie close to each other, are better localized when they emit light at different moments. In short, these techniques consist of the following steps. First, fluorescent molecules are attached to a sample. Then, via a chemical or light-dependent process, molecules are put in a 'blinking' stage. In this stage, the molecules repeatedly start emitting for a short period of time before turning off again. We call such molecules *emitters* and the bright spots they produce *fluorophores*. Generally, we will use the term 'emitter' to also refer to fluorophores when the distinction is not relevant. A microscope detects these fluorophores in a series of images. In the final step, the images are analyzed to determine the molecule locations with higher accuracy, showing the structure of the sample. Figure 1 shows two example frames of a data set and the superresolution image constructed using the 3D-DAOSTORM [2].



(a) (b)

Figure 1: Figure (a) shows one frame of a data set taken from an evaluation study by Sage et al.[36]. Figure (b) shows a zoomed-in region of the resolved image by 3D-DAOSTORM [2].

**Chemical processes** There are many different methods to turn molecules into a fluorescent stage. Now, we highlight a few prominent ones. Rust et al. developed *stochastic optical reconstruction microscopy* (STORM) [34]. In this method we activate synthetic dyes using a certain wavelength and deactivate them using another. The method is stochastic, since a random fraction of the total number of molecules is activated. If we repeat these steps a large enough number of times, each molecule is likely to be activated at least once, so we can determine its location with a higher accuracy. This accuracy can be improved, if we have multiple frames for each molecule. The number of times a molecule can be activated and deactivated is bounded. However, Rust et al. found that molecules can be cycled hundreds of times before going to a final photobleached state in which they cannot be activated again. A similar method, called PALM, is presented by Betzig et al. [4]. PALM, photoactivated localization microscopy, uses fluorescent proteins to turn molecules into fluorescent stage.

Sharonov and Hochstrasser developed the *points accumulation for imaging in nanoscale topography* (PAINT) method[38]. This method works differently from STORM and PALM. Here, emitting dyes move freely through the sample and occasionally bind to certain molecules for a certain period of time. Although the dyes are constantly emitting, we still achieve bright spots, since bound dyes remain fixed on a position, so this region will emit a constant amount of light. Other regions will emit a much smaller amount, because the dyes are constantly moving and are not in this spot for the entire duration of the frame.

Although the event of a molecule lighting up is random, we have some control over the rate at which molecules become emitters. We do not want all molecules to emit at the same time, since this will lead to overlapping fluorophores which are harder to locate precisely. Ideally, the *molecule density* is low. That is, each frame in our series contains only a few fluorophores which are nicely distributed without overlapping. Then, we can precisely locate the molecules in each frame and combine the results in a single high resolution image. To achieve this, we have to make sure that the rate at which molecules emit light is small enough, so that the probability of fluorophores overlapping is small. As only a small number of molecules light up at the same time, creating such an image series takes a lot of time. This is a disadvantage for several reasons. In many situations, molecules are not fixated at an exact position, but they can move slightly. A longer acquisition time means that molecules can move larger distances. The drift, the movement of the sample as a whole, also becomes more prevalent. And of course the experiments will take longer to conduct. This has lead to the development of algorithms that can analyze high-density data sets where there are many overlapping fluorophores. The problem that these algorithms are trying to solve is often called the *multi-emitter fitting problem*. In this thesis we investigate two approaches to this problem.

In Section 4, we will discuss many different algorithms for localizing molecules in both single-emitter and multi-emitter settings.

**Parameter of an emitter** During the localization step of an emitter there are several parameters that we can choose to incorporate in the fitting procedure. Most importantly are the positional parameters which model the emitter's $x$- and $y$-coordinate, since we are eventually interested in the exact location of the emitter. Other common parameters are the brightness and the width of the fluorophore as they determine the shape of the fluorophore we observe.

# 3 Preliminaries

## 3.1 Probability and statistics

**Maximum likelihood estimation** An important technique in parameter estimation is the *maximum likelihood estimation* (MLE). In this setting, we consider a random variable $X$ and a tuple $x = (x_1, \dots, x_m)$ containing $m$ realizations of $X$ that are all *independent and identically distributed* (IID). We assume that $X$ follows a distribution determined by a parameter $\theta \in \Theta$. For discrete probability distributions, we say that the *likelihood* of a parameter $\theta$ given a single realization $x_i$ is equal to

$$\mathcal{L}(\theta \mid x_i) := P_\theta(X = x_i).$$

The likelihood given all observations, then becomes

$$\mathcal{L}(\theta \mid x) := \prod_{i=1}^{m} P_\theta(X = x_i).$$

For continuous probability distributions, we cannot use this definition, as $P_\theta(X = x_i) = 0$. Instead, we use the density function to get

$$\mathcal{L}(\theta \mid x_i) := f_\theta(x_i).$$

Again, for a set of observations, this becomes a product. We do want to mention that there exists a more general definition that uses measure theory, but since we will not need this, we omit it here.

We define the maximum likelihood estimation as $\hat{\theta} \in \Theta$ that maximizes the likelihood. That is,

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta \mid x).$$

In practice, whenever we want to compute the MLE, it is easier to compute the parameter $\hat{\theta}$ that maximizes the *log-likelihood*

$$\ell(\theta \mid x) = \log \mathcal{L}(\theta \mid x)$$

instead. For example, in distributions containing exponents, taking the logarithm of the likelihood simplifies the optimization function. The underlying reason for this is that many distributions are log-concave, meaning that taking the logarithm results in a concave function [5] and concave functions are easier to maximize. Because the logarithm is a strictly increasing function, maximizing the log-likelihood is equivalent to maximizing the likelihood. We do have to be careful, since the likelihood can be 0. If we simply interpret $\log 0 = -\infty$ and $-\infty < a$ for all $a \in \mathbb{R}$, then there is no confusion.

**Fisher information** When estimating a parameter based on some data, we consider an *estimator* $\hat{\theta}$, a function that takes the data as argument and outputs an estimation of the parameter we are interested in. Since the data is randomly distributed, any estimator itself is also a random variable. To quantify the performance of an estimator $\hat{\theta}$, we consider both the bias and its variance. In many situations, we can design an unbiased estimator. This estimator will have bias 0 independently of the size of the data. On the other hand, the variance depends on the size of the data. Intuitively, there is an upper limit on how much information we can extract from a set of data. This vague bound is formalized using Fisher information and the Cramér-Rao bound. More precisely, the Cramér-Rao bound gives a lower bound on the variance of any possible unbiased estimator [30].

Before we define this bound, we formalize the concept of information. The general idea is that, whenever we are estimating a certain parameter, any observation gives a certain amount of information about the parameter. However, this amount can vary greatly between different random variables and the observed value. The goal of the Fisher information is to quantify how much information we expect to get, as a function of the parameter $\theta$.

As an example, we consider two normally distributed variables $X_1 \sim \mathcal{N}(\mu, 1)$ and $X_2 \sim \mathcal{N}(\mu, 10)$ and we want to estimate $\mu$. Because the variance of $X_2$ is larger, we need more samples to get the same amount of precision compared to $X_1$. In that sense we get less information per observation. Vaguely worded, we are looking for a measure to quantify the 'sharpness' of the likelihood function.

**Definition 3.1.1** (Score)**.** Let $X$ be a random variable with density function $f_\theta$ that is determined by the parameter $\theta \in \Theta$. The score $s(\theta \mid x)$ is then defined as the gradient of the log-likelihood with respect to $\theta$ given data $x$ [37]. That is, we define

$$s(\theta \mid x) = \nabla_\theta \ell(\theta \mid x) = \nabla_\theta \log \mathcal{L}(\theta \mid x).$$

If we only have one parameter, this simply reduces to $s(\theta \mid x) = \frac{\partial}{\partial\theta} \log \mathcal{L}(\theta \mid x)$. The score describes the rate of change of log-likelihood given the data as a function of $\theta$. If $\theta$ is the true parameter, we expect that this rate is 0, as the log-likelihood should be maximized here. Indeed, it holds that $\mathbb{E}\left[s(\theta \mid x) \mid \theta\right] = 0$ where the expectation is taken over all values of $x$. Before we can properly define the Fisher information, we need to ensure that the following regularity conditions hold [37]:

1. The partial derivatives of $f_\theta(x)$ over must be defined almost everywhere.
2. We can integrate $f_\theta(x)$ over $\theta$.
3. The support of $f_\theta(x)$ does not depend on $\theta$.

These conditions exclude for example the uniform distributed $X \sim \mathcal{U}(0, \theta)$, since its support is equal to $[0, \theta]$.

**Definition 3.1.2** (Fisher information)**.** If the above conditions hold, then the *Fisher information* is defined as the variance of the score, given that $\theta$ is the true parameter, i.e.:

$$\mathcal{I}(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log\mathcal{L}(\theta \mid X)\right)^2 \,\middle|\, \theta\right]$$

where we take the expectation with respect to our random variable $X$. Recall that this was the random variable that we observe with realizations $x$.

This is indeed equal to the variance, as one can show that the expectation of the score is equal to 0, that is $\mathbb{E}\left[\frac{\partial}{\partial\theta}\log\mathcal{L}(\theta \mid X) \,\middle|\, \theta\right] = 0$. In higher dimensions, this becomes the *Fisher information matrix* which is defined as

$$\mathcal{I}(\theta)_{i,j} = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta_i}\log\mathcal{L}(\theta \mid X)\right)\left(\frac{\partial}{\partial\theta_j}\log\mathcal{L}(\theta \mid X)\right) \,\middle|\, \theta\right].$$

**Proposition 3.1.3.** *If, on top of the regularity conditions, the second partial derivatives are also defined almost everywhere, we get an equivalent definition:*

$$\mathcal{I}(\theta)_{i,j} = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log\mathcal{L}(\theta \mid X) \,\middle|\, \theta\right]$$

*Proof.* We have that

$$\begin{aligned}\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log\mathcal{L}(\theta \mid X) &= \frac{\partial}{\partial\theta_i}\left(\frac{\frac{\partial}{\partial\theta_j}\mathcal{L}(\theta \mid X)}{\mathcal{L}(\theta \mid X)}\right)\\ &= \frac{\mathcal{L}(\theta \mid X)\left(\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\theta \mid X)\right) - \left(\frac{\partial}{\partial\theta_i}\mathcal{L}(\theta \mid X)\right)\left(\frac{\partial}{\partial\theta_i}\mathcal{L}(\theta \mid X)\right)}{\mathcal{L}^2(\theta \mid X)}\\ &= \frac{\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\theta \mid X)}{\mathcal{L}(\theta \mid X)} - \left(\frac{\partial}{\partial\theta_i}\log\mathcal{L}(\theta \mid X)\right)\left(\frac{\partial}{\partial\theta_j}\log\mathcal{L}(\theta \mid X)\right).\end{aligned}$$

Then, we can show that

$$\begin{aligned}\mathbb{E}\left[\frac{\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\theta \mid X)}{\mathcal{L}(\theta \mid X)} \,\middle|\, \theta\right] &= \int_{\mathbb{R}}\frac{\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}(\theta \mid x)}{\mathcal{L}(\theta \mid x)}f(x \mid \theta)\,\mathrm{d}x\\ &= \int_{\mathbb{R}}\frac{\frac{\partial^2}{\partial\theta_i\partial\theta_j}f(\theta \mid x)}{f(\theta \mid x)}f(x \mid \theta)\,\mathrm{d}x\\ &= \frac{\partial^2}{\partial\theta_i\partial\theta_j}\int_{\mathbb{R}}f(\theta \mid x)\,\mathrm{d}x\\ &= \frac{\partial^2}{\partial\theta_i\partial\theta_j}1 = 0.\end{aligned}$$

□

This concludes the proof.

**Example 3.1.4.** As an example, we consider a Gaussian random variable with mean $\theta$ and variance $\sigma^2$ and we assume that we have $n$ independently selected samples $x = (x_1, \dots, x_n)$. In this case, we have for every individual observation

$$\mathcal{L}(\theta \mid x_i) = f_\theta(x_i) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x_i-\theta)^2}{2\sigma^2}}.$$

The second derivative of the log-likelihood becomes

$$\frac{\partial^2}{\partial\theta^2}\ell(\theta \mid x_i) = \frac{\partial^2}{\partial\theta^2}\left(-\log\left(\sqrt{2\pi}\sigma\right) - \frac{(x_i-\theta)^2}{2\sigma^2}\right) = -\frac{1}{\sigma^2}$$

Then, we can compute the Fisher information for the total sample $x$ as follows:

$$\begin{aligned}\mathcal{I}(\theta) &= -\mathbb{E}_\theta\left[\frac{\partial^2}{\partial\theta^2}\log\mathcal{L}(\theta \mid x)\right] \\ &= -\mathbb{E}_\theta\left[\frac{\partial^2}{\partial\theta^2}\log\left(\prod_{i=1}^n \mathcal{L}(\theta \mid x_i)\right)\right] \\ &= -\mathbb{E}_\theta\left[\sum_{i=1}^n \frac{\partial^2}{\partial\theta^2}\ell(\theta \mid x_i)\right] = \frac{n}{\sigma^2}\end{aligned}$$

It follows that a high variance results in low Fisher information. Also note that in this particular case the Fisher information does not depend on $\theta$, since the shape of the density function does not change when shifting the mean. △

**Cramér-Rao bound** The Cramér-Rao bound is a statistical result on the variance of unbiased estimators. Suppose we have a one-dimensional parameter $\theta$ that we want to estimate, using an estimator $\hat{\theta}$ and suppose this estimator is unbiased, i.e. $\mathbb{E}[\hat{\theta}] = \theta$. Then, the Cramér-Rao bound [30] states that

$$\operatorname{var}(\hat{\theta}) \geq \frac{1}{\mathcal{I}(\theta)},$$

where $I(\theta)$ is the Fisher information of $\theta$.

To continue Example 3.1.4, we find that for any estimator $\hat{\theta}$ of the mean, it holds that $\operatorname{var}(\hat{\theta}) \geq \sigma^2/n$. Note that this lower bound is achieved by using the sample mean for $\hat{\theta}$.

## 3.2 Continuous optimization

In almost all settings where we are estimating a certain parameter, we are in fact optimizing some function. We define a minimization problem as

$$\min_{x\in\mathbb{R}^n} h(x)$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a function. Although maximum likelihood estimation is a maximization problem, we can turn it into a minimization problem if we negate the objective function. Note that we use in this section the variable $x$ instead of $\theta$ as we did for the MLE. We chose to use $x$ here, since this section describes general optimization techniques, while we use $\theta$ solely to denote parameters.

Another optimization approach is *least squares (LS) optimization* which is not based on probability theory. Here, we have a vector of measurements $d = (d_1, \ldots, d_m) \in \mathbb{R}^m$ and a set of functions $q_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, m$. We define the residuals $r_i = d_i - q_i(x)$ and we consider the minimization problem

$$\min_{x\in\mathbb{R}^n} R(x) := \min_{x\in\mathbb{R}^n}\sum_{i=1}^m r_i(x)^2 = \min_{x\in\mathbb{R}^n}\sum_{i=1}^m \left(d_i - q_i(x)\right)^2.$$

We estimate the input or parameter by minimizing the sum of the squared errors predicted by the functions. As an example, we can consider the single emitter localization problem in an image $I$. The measurement $d$ consists of the pixel values, $d_i = I(i)$ where $i \in [WH]$. For simplicity we assume that only the positional parameters of the emitter are unknown and that the brightness and width of the fluorophore are known. The functions $q_i$ take then two parameters $(x_1, x_2)$ as arguments and return the predicted intensity of pixel $i \in [WH]$. The goal is then to find positions $(x_1, x_2) \in \mathbb{R}^2$ that minimize $R$.

Generally, it is not possible to find the minimizer of a continuous optimization problem analytically, except for very specific situations such as a least squares problem with linear $q_i$ [5, Chapter 1]. Otherwise, we have to resort to iterative methods. Iterative methods gradually improve the solution, until we reach a local minimum. Note that these method have no guarantee of finding the global minimum, unless the problem is pseudoconvex [24] or we are very careful with how we choose our initial solution. Now, we present some iterative methods that employ the gradient of the objective function.

**Gradient descent method** The simplest gradient method is the *gradient descent algorithm.* We start with an arbitrary $x_0$ and we use the iteration

$$x_{k+1} := x_k - t_k \nabla h(x_k),$$

where we can choose the step size $t_k$ in each iteration. This method has the advantage that for a small enough $t_k$, the iteration is guaranteed to improve the solution. One choice for $t_k$ is using *exact line search* [5, Chapter 9]. Here, we determine the optimal choice for $t$ in each iteration. Formally, we consider the second optimization problem

$$\min_{t \in \mathbb{R}} h(x_k - t\nabla h(x_k)).$$

One can show that convergence is guaranteed, whenever $h$ is strongly convex and we can also prove a rate of convergence. However, convergence is slow when the condition number of the Hessian matrix is large. This slow convergence is major disadvantage of gradient descent and it is often better to use different methods. [5]

**Newton-Raphson method** Another common approach is the *Newton-Raphson method* or simply *Newton's method* [5, Chapter 9]. Newton's method is originally a root-finding algorithm, but a variation can be used in minimization problems. The iteration for Newton's method is

$$x_{k+1} = x_k - \nabla^2 h(x_k)^{-1} \nabla h(x).$$

where $\nabla^2$ is the Hessian matrix. In the gradient descent method we compute the linear approximation of $h$ at point $x_k$. Then, we move in the optimal direction which is the negative gradient. Newton's method computes the quadratic approximation of $h$ and then jumps to its stationary point. We can show this as follows. We write the quadratic approximation as

$$h(x_k + \delta) \approx h(x) + \nabla h(x)^T \delta + \delta^T \nabla^2 h(x) \delta$$

Then, the stationary point is such that $\nabla_\delta h(x_k + \delta) = 0$, that is

$$0 = \nabla_\delta h(x_k + \delta) \approx \nabla h(x) + \nabla^2 h(x)\delta.$$

It follows that $\delta = -\nabla^2 h(x)^{-1} \nabla h(x)$. Newton's method can achieve a much faster rate of convergence than the gradient descent method, but it only works in certain situations. Specifically, when the Hessian is not invertible, this method fails and when the Hessian is ill-conditioned, computing the inverse is sensitive to numerical errors. Moreover, the fast convergence is only achieved when the starting point is 'close' to the optimum. In [5] sufficient (but not necessary) conditions are described for quadratic convergence.

Another disadvantage is the need for the inverse of the Hessian matrix in every iteration. Previously, we stated that we only consider methods that use the gradient, and we will now remove this dependency.

**Gauss-Newton method** Although the gradient descent method and the Newton-Raphson method have some conditions for convergence, they work for a general set of minimization problems. If we focus specifically on least squares optimization, we can utilize the specific form of this problem. The *Gauss-Newton method* is a modification for LS problems. If we apply the Newton-Raphson method, we compute the gradient and the Hessian as follows:

$$\begin{aligned} \nabla_j R(x) &= \frac{\partial}{\partial x_j} \sum_{i=1}^{m} r_i(x)^2 \\ &= 2 \sum_{i=1}^{m} r_i(x) \frac{\partial}{\partial x_j} r_i(x) \end{aligned}$$

Let $r(x) = (r_1(x), \ldots, r_m(x))$ and let $\mathbb{J}$ be the Jacobian matrix of $r$. Then, we have $\nabla R(x) = 2\mathbb{J}^T r(x)$. For the Hessian, we have

$$\begin{aligned}\nabla^2_{j,k} R(x) &= \frac{\partial^2}{\partial x_j \partial x_k} \sum_{i=1}^m r_i(x)^2 \\ &= 2\sum_{i=1}^m \left[ \frac{\partial}{\partial x_j} r_i(x) \frac{\partial}{\partial x_k} r_i(x) + r_i(x) \frac{\partial^2}{\partial x_j \partial x_k} r_i(x) \right].\end{aligned}$$

Now, we approximate the Hessian by omitting the second order derivatives. We get a matrix

$$H_{j,k} = 2\sum_{i=1}^m \frac{\partial}{\partial x_j} r_i(x) \frac{\partial}{\partial x_k} r_i(x),$$

which we can write as $H = 2\mathbb{J}^T\mathbb{J}$. If we use $H$ instead of the Hessian, we get the recursion

$$x_{k+1} = x_k - (\mathbb{J}^T\mathbb{J})^{-1}\mathbb{J}^T r(x_k).$$

It might not be clear if leaving out the higher order derivative in the Hessian provides a meaningful estimate for it, but we can also derive the same relation in different way. Suppose we write our iteration as $x_{k+1} = x_k + \delta$. We want to choose our $\delta$ in such a way that $R$ attains a local minimum at $x_k + \delta$. This happens whenever $\nabla_\delta R(x_k + \delta) = 0$. Of course, $R$ is too complicated to determine $\delta$ directly, but we can use a linear approximation of $r$ in point $x_k$. We have that

$$r(x_k + \delta) \approx r(x_k) + \mathbb{J}\delta.$$

If we note that we can write $R(x) = r(x)^T r(x)$, we are actually solving the optimization problem

$$\min_\delta R'(\delta) := \min_\delta \|r(x_k) + \mathbb{J}\delta\|_2^2$$

which we can solve as follows. We have that

$$\begin{aligned}\nabla_\delta R'(\delta) &= \nabla_\delta \|r(x_k) + \mathbb{J}\delta\|_2^2 \\ &= \nabla_\delta (r(x_k) + \mathbb{J}\delta)^T (r(x_k) + \mathbb{J}\delta) \\ &= \nabla_\delta \left(2r(x_k)^T\mathbb{J}\delta + \delta^T\mathbb{J}^T\mathbb{J}\delta\right) \\ &= 2\mathbb{J}^T r(x_k) + 2\mathbb{J}^T\mathbb{J}\delta.\end{aligned}$$

Setting this equal to 0, we have that $\delta = -(\mathbb{J}^T\mathbb{J})^{-1}\mathbb{J}^T r(x_k)$, which gives the same result as before.

**Levenberg-Marquardt algorithm** The Gauss-Newton method suffers from the same problems as Newton-Raphson. The convergence is only suitable whenever our solution is already 'close' to the optimum. When the current iteration lies further away from the optimum, the next iteration might even increase the optimization function. In these cases we might produce an even worse solution. Levenberg [19] and Marquardt [25] both independently suggested a combination of Gauss-Newton and gradient descent. More specifically, we add a matrix to $\mathbb{J}^T\mathbb{J}$ before inverting it. One choice is simply $\lambda I$ where $I$ is the identity matrix of appropriate size and $\lambda > 0$ is the *dampening factor*. Then, the iteration becomes

$$x_{k+1} = x_k - (\mathbb{J}^T\mathbb{J} + \lambda I)^{-1}\mathbb{J}^T r(x_k).$$

When $\lambda$ is close to 0, this iteration resembles the Gauss-Newton method. If $\lambda$ becomes large, the iteration behaves like gradient descent with $t_k = 1/\lambda$.

From the viewpoint of regularization, we can arrive at the same iteration if we consider the problem

$$\min_\delta R'(\delta) + \lambda\|\delta\|_2^2.$$

This minimization problem aims to minimize $R'$ while keeping $\delta$ relatively small. Then, we have

$$\begin{aligned}\nabla_\delta \left(R'(\delta) + \lambda\|\delta\|_2^2\right) &= 2\mathbb{J}^T r(x_k) + 2\mathbb{J}^T\mathbb{J}\delta + 2\lambda\delta \\ &= 2\mathbb{J}^T r(x_k) + 2\left(\mathbb{J}^T\mathbb{J} + \lambda I\right)\delta\end{aligned}$$

which leads to the same iteration. Instead of using the identity matrix, both Levenberg and Marquardt also suggest using $\operatorname{diag}(\mathbb{J}^T\mathbb{J})$. Then, the dampening factor is scaled differently for each parameter. Intuitively, we take larger steps in the direction of parameters with a small gradient. This speeds up the slow convergence of gradient descent in flat areas.

Finally, we need to choose the dampening parameter $\lambda$ for the algorithm. Marquardt [25] suggests the following. Start with a choice $\lambda_0$ and choose some $\nu > 1$. Then, at iteration $k$, we compute the optimal $\delta$ for both $\lambda_k$ and $\lambda_k/\nu$. If both values result in a worse solution, we choose $x_{k+1} = x_k$ and instead let $\lambda_{k+1} = \nu\lambda_k$. If $\lambda_k/\nu$ results in a worse solution and $\lambda_k$ results in an improvement, we use the latter, but leave $\lambda_k$ as it is. If using $\lambda_k/\nu$ improves the solution, we choose $\lambda_{k+1} = \lambda_k/\nu$ and continue with the improved solution.

**Majorization minimization** An optimization technique that is not based on gradients is the MM algorithm. It either stands for *majorization minimization* or *minorization maximization*, depending on which form of optimization is used. In this section we focus on majorization minimization, but maximization works analogously.

Suppose we have a function $h : \mathbb{R}^n \to \mathbb{R}$ that we want to minimize without constraints. The MM algorithm is an iterative method where we start with an initial solution $x_0$ which we improve in each iteration. We denote the point in iteration $i$ by $x_i$. The general idea is in each iteration to construct a surrogate function $g$ that majorizes $h$.

**Definition 3.2.1.** Let $h : \mathbb{R}^n \to \mathbb{R}$ be a function. We call $g(\cdot \mid \widehat{x}) : \mathbb{R}^n \to \mathbb{R}$ a *majorizer* of $h$ in point $\widehat{x}$ if:

- $g(\cdot \mid \widehat{x})$ touches $h$ in $\widehat{x}$, i.e. $g(\widehat{x} \mid \widehat{x}) = h(\widehat{x})$;
- $g(\cdot \mid \widehat{x})$ bounds $h$ from above, i.e. $g(x \mid \widehat{x}) \geq h(x)$ for all $x \in \mathbb{R}^n$.

We also say that $g(\cdot \mid \widehat{x})$ *majorizes* $h$ at $\widehat{x}$ or simply that $g(\cdot \mid \widehat{x})$ majorizes $h$, when it is clear from the context at which point $h$ is majorized.

We will use some properties of majorizers.

**Proposition 3.2.2.** *Let $h, h_1, \ldots, h_m : \mathbb{R}^n \to \mathbb{R}$ be functions such that $h = h_1 + \ldots + h_m$ and let $g_1(\cdot \mid \widehat{x}), \ldots, g_m(\cdot \mid \widehat{x})$ be majorizers of $h_1, \ldots, h_m$ respectively. Then, $g(\cdot \mid \widehat{x}) = g_1(\cdot \mid \widehat{x}) + \ldots + g_m(\cdot \mid \widehat{x})$ is a majorizer of $h$.*

*Proof.* The proof is straightforward. It holds trivially that

$$g(\widehat{x} \mid \widehat{x}) = g_1(\widehat{x} \mid \widehat{x}) + \ldots + g_m(\widehat{x} \mid \widehat{x}) = h_1(\widehat{x}) + \ldots + h_m(\widehat{x}) = h(\widehat{x})$$

and

$$g(x \mid \widehat{x}) = g_1(x \mid \widehat{x}) + \ldots + g_m(x \mid \widehat{x}) \geq h_1(x) + \ldots + h_m(x) = h(x)$$

for all $x \in \mathbb{R}^n$. □

The general idea of an MM algorithm at iteration $i$ is to construct a majorizer $g(\cdot \mid x_i)$ and choose $x_{i+1} = \arg\min_x g(x \mid x_i)$. This strategy is of course only viable when the majorizer can be minimized easily and constructing a useful majorizer can be difficult. Hunter and Lange present some common techniques to construct majorizers, including Jensen's inequality, Cauchy-Schwartz inequality or constructing a quadratic polynomial [12]. We will use one technique discussed in their paper which we will utilize.

**Proposition 3.2.3.** *Suppose we have a convex function $\kappa : \mathbb{R} \to \mathbb{R}$ and vectors $c, \widehat{x} \in \mathbb{R}^n$. Then the function $h(x) = \kappa(c^T x)$ can be majorized at $\widehat{x}$ with the function*

$$g(x \mid \widehat{x}) = \sum_{i=1}^{n} \alpha_i \kappa(t_i)$$

*where $t_i = c^T\widehat{x} \cdot x_i/\widehat{x}_i$ and $\alpha_i = c_i\widehat{x}_i/(c^T\widehat{x})$.*

*Proof.* It holds that

$$\begin{aligned} g(\widehat{x} \mid \widehat{x}) &= \sum_{i=1}^{n} \frac{c_i \widehat{x}_i}{c^T \widehat{x}} \kappa \left( \frac{c^T \widehat{x} \cdot \widehat{x}_i}{\widehat{x}_i} \right) = \sum_{i=1}^{n} \frac{c_i \widehat{x}_i}{c^T \widehat{x}} \kappa \left( c^T \widehat{x} \right) \\ &= \kappa(c^T \widehat{x}) \sum_{i=1}^{n} \frac{c_i \widehat{x}_i}{c^T \widehat{x}} = \kappa(c^T \widehat{x}) \cdot \frac{c^T \widehat{x}}{c^T \widehat{x}} = \kappa(c^T \widehat{x}) = h(\widehat{x}) \end{aligned}$$

showing the first property. For the second property we employ the following inequality that holds for convex functions:

$$\kappa \left( \sum_i \beta_i t_i \right) \leq \sum_i \beta_i \kappa(t_i)$$

where $\beta_i \geq 0$ for $i = 1, \ldots, n$ and $\sum_i \beta_i = 1$. This is just a generalization of the defining property of convex functions and the proof can be easily showed by induction. Since our $\alpha_i$ sum to 1, we can show that

$$\begin{aligned} g(x \mid \widehat{x}) &= \sum_{i=1}^{n} \alpha_i \kappa(t_i) \geq \kappa \left( \sum_{i=1}^{n} \alpha_i t_i \right) \\ &= \kappa \left( \sum_i \frac{c_i \widehat{x}_i}{c^T \widehat{x}} \cdot \frac{c^T \widehat{x} \cdot x_i}{\widehat{x}_i} \right) = \kappa \left( \sum_i c_i x_i \right) = h(x). \end{aligned}$$

This concludes the proof. □

## 3.3 Light detection model

Here, we present a model for the photon detection process. This encompasses both modeling the photon detector and the light distribution coming from the sample. The latter part we largely base on [6] that extensively discusses a number of models that increase in how well they model the real world.

**Model of CCDs** As a photon detector we consider the *charge-coupled device (CCD)*. A CCD camera consists of a two-dimensional array of detectors that collect incoming photons such that each detector collects photons from a specific area of the sample. These photons are activating electrons, building up a charge in the detector, which is initially 0. After a certain time this charge is measured and converted to an integer number. A CCD will not activate an electron for every photon that enters and the fraction of photons that are converted is called the *quantum efficiency*. Moreover, the CCD will measure an electric charge and not the number of photons. To recover the number of electrons, a CCD camera specifies how many electrons constitute one unit that the CCD outputs. Finally, the output can have an offset constant over time. This is called the *baseline* of a detector and the offset can vary pixel by pixel, meaning that the same electric charge can produce different output values. There exist methods that can determine the baseline as discussed in [15]. As this is not the primary focus of this thesis, we will simply assume the baseline is known.

Not all electrons that are detected will originate from a photon originating from the sample [6]. Spurious photons can be detected and the detector itself can measure *dark current*, small current that is present in a detector, even when no photons enter the device [8].

Finally, we briefly want to mention *EMCCD (electron-multiplying CCD)* [7]. These devices increase the number of electrons measured. In short, the number of electrons is amplified via a random process that is repeated several times. The main advantage is that noise from reading out the number of electrons becomes less prevalent. This is especially useful, when the number of electrons is very small. Although we will not use this concept in our model, we mentioned it for completeness.

**Photon distribution model** Now, we describe our model for the photons arriving at the detector. This model allows us to design algorithms for our problems, as well as provide a way to simulate images. Note that these models can be applied in a more general setup than our localization setting.

Our first assumption is that the emission of photons by our sample, that are directed at the detector, is a random process. In particular, we assume that it follows a Poisson process with rate $\Lambda_\theta(\tau)$ which we call the *photon detection rate*. Here, $\theta \in \Theta$ is the parameter that determines where and how many photons are emitted by our sample. This is a very general approach, but in our case we can have for example $\theta = (x, y, \lambda)$ where $\theta$ contains the location $(x, y)$ and brightness $\lambda$ of one molecule. In this case mainly $\lambda$ will influence the photon detection rate (unless the molecule's position is outside the detector range). We assume the detection rate to be Poisson, since we model the photons as independent events. Previous photons do not influence the probability of the next photon. The model also allows for the distinction of homogeneous and inhomogeneous Poisson processes. If the light intensity of an object changes over time, we could model it as an inhomogeneous model, while an object with a relative constant brightness follows a homogeneous Poisson process.

Secondly, we describe the *photon distribution profile*. The photon distribution profile is a two-dimensional probability density that describes the probability distribution of the photons on the detector over time. We denote this density by $f_{\theta,\tau}(x, y)$, with $(x, y) \in \mathbb{R}^2$ being the position on the detector and $\theta \in \Theta$ and $\tau \geq 0$ as before. In the case of just one object emitting light, the photon distribution profile is in the literature often called the *point spread function* (PSF). When we have multiple molecules emitting light, we can view the distribution profile as the sum of different translated PSFs. To model the PSF, the two-dimensional Gaussian is a sensible distribution to use, but other distributions might model certain data more accurately. For example, some molecules do not produce a rotationally symmetrical pattern, since they contain a dipole [10]. A different commonly used distribution is the Airy pattern in Figure 2, consisting of a high density spot in the center with less dense concentric circles surrounding it. Note that the figure has brightened lower densities, so the concentric rings become visible. The Airy pattern is a more realistic distribution as light traveling through a perfect lens produces such a pattern. We will limit ourselves to a two-dimensional distribution as that simplifies later equations significantly.

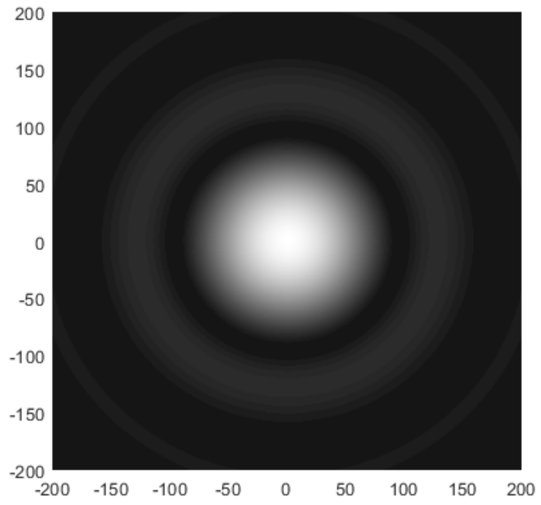


Figure 2: Airy pattern with the altered intensities to enhance the concentric rings

## 3.4 Image data models

There are different ways how we can model the detected data. These models describe how we represent the image data to which we apply our analysis algorithms.

**Fundamental data model** The fundamental data model is a theoretical model that assumes and infinite detection area and no noise or discretization of the photon detection step. This means that the location of each photon that is detected has infinite precision and that no noise comes from the photon detection process. Suppose we consider a time interval $[a, b]$. The number of photons detected then follows the Poisson distribution and the exact times follow the Poisson process $\Lambda_\theta$. The location of each photon detection now follows the photon distribution profile $f_{\theta,\tau}$ exactly.

**Poisson data model** The assumption of no discretization assures that such a model can only be theoretical. In reality, a photon detector has only finite precision which leads to a discretization of the photon detection. This results in a pixel-like structure where we can interpret the brightness of a pixel as the number of photons detected by the sensor in this position. This results in the *Poisson data model* which we will use in the remainder of this thesis. The Poisson data model only considers a finite area and divides this area with a grid. On top of that we also model the noise caused by additional photons mentioned above. We denote the number of pixels by $K$ and let $H_{\theta,k}$ be the random variable representing the detected number of photons in pixel $k$ with $k = 1, \ldots, K$. Then, we have

$$H_{\theta,k} = S_{\theta,k} + B_{\theta,k},$$

where $S_{\theta,k}$ represents the number of photons from our sample detected by pixel $k$ and $B_{\theta,k}$ represents the noise. The random variable $S_{\theta,k}$ follows the same Poisson distribution as the number of photons in the entire image $\Lambda_\theta$ only scaled by the probability density over the area of the pixel. It follows that $S_{\theta,k} \sim \text{Poisson}(\lambda^S_{\theta,k})$, where

$$\lambda^S_{\theta,k} = \int_a^b \Lambda_\theta(t) \int_{C_k} f_\theta(x,y)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t.$$

Here, $C_k$ is the area of pixel $k$. Similarly, we see that $B_{\theta,k} \sim \text{Poisson}(\beta^B_{\theta,k})$, where

$$\beta^B_{\theta,k} = \int_a^b \Lambda^B_\theta(t) \int_{C_k} f^B_\theta(x,y)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t.$$

Here, $\Lambda^B_\theta$ and $f^B_\theta$ are the photon detection rate and the photon distribution profile of the noise, respectively. We assume that $S_{\theta,k}$ and $B_{\theta,k}$ are independent of each other, and since $H_{\theta,k}$ is the sum of two independent Poisson variables, we have that $H_{\theta,k} \sim \text{Poisson}(\lambda^S_{\theta,k} + \beta^B_{\theta,k})$. The fact that the number of detected photons is a random process, means that neighboring pixels with roughly the same parameter $\lambda^S_{\theta,k}$ can have widely different actual intensities. This results in a noisy image which we call *shot noise*.

Finally, we will assume that the background noise parameter $\beta^B_{\theta,k}$ is constant and the same for all pixels. From now on we denote it simply by $\beta$.

# 4 Literature study

A lot of research has already been done in the area of localization microscopy. In this section, we discuss the general structure that localization algorithms follow. Then, for each step, we discuss some of the most common choices, as well as some specific choices that were made in some papers. We want to note that some (steps of) algorithms are applied to each of the frames separately. In that case we simply refer to each frame as an image.

Most localization algorithms, both single-emitter and multi-emitter, follow a similar structure. They consist of four individual steps: a preprocessing step, a detection step, a localization step and a post-processing step. The preprocessing step is used to remove the background from the image. For example, the background can consist of a large bright area in the image which is not caused by emitting photons, but rather by the experimental setup. In the detection step, the goal is to find regions of interest that likely contain a fluorophore. This splits the total image in smaller subimages. In the localization step, we analyze these subimages to get a precise estimation of the location of the molecules. Finally, in the post-processing step, we analyze the locations by filtering localizations or merging localizations that lie too close together.

## 4.1 Preprocessing step

The goal of the preprocessing step is to remove some of the background in the image. A simple approach is to average all frames into a single image and subtract this from each frame. Some pixels will receive a negative value which can cause issues that need to be accounted for. Especially when doing a maximum likelihood estimation, all pixel values need to be non-negative.

Some more advanced methods of background removal are described in [13]. Hüpfel et al. compare two existing methods, the rolling ball algorithm and difference of Gaussians, to a new wavelet filtering approach which all aim to remove brighter background areas, while keeping smaller features. The rolling ball algorithm method visualizes an image as a three-dimensional graph, where the intensity is the $z$-coordinate. For each pixel, it then places a fixed size sphere at the pixel's position. The $z$-coordinate is determined as the largest value such that the sphere lies entirely below the graph. The top of this sphere is then subtracted from the pixel intensity. Background pixels that lie in a large bright background area will roughly be set to a low intensity. The radius of the sphere can be chosen freely, but if the radius is too small, the features of the image are also removed. In contrast, in the difference of Gaussians method we convolve an image (frame) with two different Gaussian filters, where the first filter uses a smaller $\sigma$. Then, we subtract the result of the first convolution from the result of the second one. Again, large bright background areas will be removed, since the convolution in these areas of both filters is the same. Small features are preserved, since the first Gaussian filter has a higher peak.

Finally, Hüpfel et al. [13] introduce a wavelet filtering approach. In this approach, they transform the image to a frequency domain, similar to a Fourier transform, but with a different transformation. Specifically, they use the Haar wavelet [18]. In this frequency domain, they determine the large frequencies, representing the background areas, and construct a background image. After subtracting it from the original image, the result is an image without background.

## 4.2 Molecule detection step

After we have removed the background, we find locations that are likely to be close to a molecule. Then, we take part of the image around each of these locations and on these *regions-of-interest (ROIs)* we perform a localization algorithm. Using a filter to smooth the noise in the image, we can reduce the number of false positive and false negative detections [16].

Determining the ROIs can be done via different methods. We can simply find all pixels above some threshold, and then remove closely situated pixels, as they are likely to correspond to the same fluorophore [39]. A different approach is to compute local maxima. In [11], Huang et al. apply a square maximum filter over the image and then find the pixels whose intensity remained equal. This effectively comes down to finding each pixel that achieves the maximum intensity in the area surrounding it.

WindSTORM by Ma, Xu and Liu [23] applies a one-step deconvolution to separate overlapping fluorophores. This one step deconvolution is done by applying the Fourier transform to the image, dividing by the modulation transfer function (roughly the Fourier transform of the PSF), and transforming the image back using the inverse Fourier transform. Then, they also simply find the local maxima to find individual emitters. An advantage of this approach is that it immediately gives a good estimation for ROIs containing multiple emitters.

A computationally expensive algorithm for the multi-emitter problem is FALCON by Min et al. [26]. In the detection step, they place a finer grid on top of the pixel grid and then perform two entire deconvolutions where they compute a value for each pixel. In both deconvolutions, they assume that the emitters are situated at the locations of these subpixels and they approximate an arbitrary PSF at a certain point using the linear Taylor expansion. They use the LS error between the measured image and the convolution, when emitters with certain intensities are located in the subpixels. To greatly reduce the number of subpixels that will light up, they added a regularization term which is a weighted $\ell_1$ norm over all subpixels. In order to remove some bias, they then perform the same deconvolution without the regularization term, but instead, they only consider the subpixels that had a positive support after the first deconvolution.

## 4.3 Molecule localization step

In the localization step, we must determine the precise location of the molecules in each ROI. The simplest localization method is using the centroid of the ROI [14]. Here, we compute the $x$- and $y$-coordinate of the molecule by taking the weighted average over all pixels. This method is only suitable for single-emitter algorithms, as we can only achieve a single coordinate per ROI. Although it is fast and simple to implement, the results are inaccurate when the ROI contains parts of other fluorophores. The centroid method also has a theoretical downside [27]. We assume an idealized setting where we have no noise and an infinite image which follows the PSF exactly. PSFs like the Airy disk function decrease at the rate of $1/r^3$, where $r$ is the distance to the center. This means that this distribution has infinite variance and, as a consequence, the centroid estimation has infinite variance as well.

A more advanced and commonly used localization algorithm is the maximum likelihood estimation (MLE). In the single-emitter case, we can calculate the likelihood that we observe the image data given some parameters. We can choose which parameters to use depending on our prior knowledge and which PSF we use. For example, if we use a 2D Gaussian, we can only include the molecule position $(x, y)$ and assume that the intensity and the covariance matrix are known and constant. Then, we maximize the likelihood over all positions in the image. This approach produces the theoretical optimal result. By optimal we mean that the variance of this estimator approaches the Cramér-Rao bound as the SNR increases [27, 36]. In [20], Li et al. use a weighted MLE instead. They search for local maxima in the detection step. Since the resulting subimage then has brighter pixels at the center, these pixels contain more reliable information. The subimage is then pointwise multiplied with a 2D Gaussian weight function centered in the subimage. This also decreases the bias of the MLE, whenever a second fluorophore is present at side of the subimage.

For multi-emitter algorithms, we have the additional problem of deciding how many emitters are in a certain subimage. When we have selected this number, we add parameters for the other molecules. In [11], Huang et al. use a MLE with a 2D Gaussian PSF. They iteratively increase the number of emitters until the resulting predicted image approximates the actual image close enough. To ensure robustness, they only model the locations of the molecules and a constant background as the parameters.

A different approach that is also common in the literature is the least squares estimation. We compute the expected intensity for each pixel. The difference with the measured intensities form the residuals that allow us to compute the LS error. This approach is taken by Thompson et al. [39].

The original papers on STORM [34] and PALM [4] both use the LS estimation as well for the molecule locations. In [34], they fit an ellipsoidal Gaussian with arbitrary angle. However, when the ellipticity is too large, they reject the sample, since it could indicate the presence of multiple fluorophores. As mentioned in [17], this can result in bias, since areas with a high molecule density will reject more emitters than low density areas.

Several studies have been done that compare MLE with LS [1, 27]. A good criterion is to compare the bias and variance of both methods. The idea of measuring the accuracy by comparing the variance to the Cramér-Rao bound was first introduced by Ober et al. [31]. As previously stated, LS estimations generally converge much faster than MLE iterations, so the algorithms become faster. In [27], Mortensen et al. compare MLE and LS estimation and find that the average distance to the true location is larger for LS estimation compared to MLE and the Cramér-Rao bound. They suggest to use LS to get a fast and good approximate location and then use MLE for an improvement. Abraham et al. [1] find similar results. Although both MLE and LS estimaton find the true location on average, the variance of LS is higher compared to MLE, which approaches the Cramér-Rao bound.

## 4.4 Post-processing step

After the localization step, we end up with a list of molecule locations. If our data is a time series, we have a list of molecule locations for each frame. The post-processing step is for any tweaking to get more desirable results. For example, we can combine all frames in a time series into a single frame and try to detect molecules that occur more than once. Then, we can either remove the least certain locations or average the two locations in some way. If we use a multi-emitter approach, it is also possible that the subimages are overlapping and then we could have duplicates in a single frame.

The post-processing can also be used to remove undesirable locations, when for example the intensity is too low or the likelihood is too small. It might also be the case that the shape of fluorophore differs too much from the PSF.

## 4.5 Other methods

Not all algorithms follow the four step structure that we previously described. Some of these methods also compute a higher resolution image directly, instead of a list of localizations. These approaches are especially suited for high-density data, where locating individual molecules becomes more difficult, due to the large number of overlapping fluorophores. Most notable is the use of convolutional neural networks. One such example is Deep-STORM [29] by Nehme et al., which focuses on high density images. Contrary to the previous algorithms, Deep-STORM is parameter free, since a single neural network is trained. The network takes a (time) series of images as input and it produces a super-resolved image per frame as output, which are then summed up to get a resulting image. Advantages of this approach is that the eventual algorithm is very fast. Reconstructing a higher resolution image only requires a single feed-forward evaluation of the network. The method is also parameter-free, so it does not have the disadvantage of getting wrong results because of incorrectly chosen parameters. A large difference compared to the previously described algorithms is that the network does not produce a list of molecule locations. Most other algorithms compute a list of locations from which a higher resolution image is constructed. Since Deep-STORM directly outputs such an image, it is difficult to compare its performance in terms of measures like the Cramér-Rao bound. Of course, training the network takes a relatively long time. Nehme et al. stated that the training step took roughly two hours.

In [40] Zelger et al. focus on three-dimensional localization of single emitters using a convolutional neural network. Instead of outputting a super-resolved image, it simply outputs the locations.

A different class of methods focus on deconvolution of a higher resolution pixel grid, similar to FALCON. Gazagnes et al. [9] used this approach with a different regularization term. Let the image be a grid of dimension $N \times N$. They place a finer grid of size $NL \times NL$ over the image. The assumption is then that the original image is the result of a convolution of the ground truth image, which is then scaled down by averaging blocks of $L \times L$ pixels. For a given finer image, we can then construct the theoretical measured image. Gazagnes et al. define the optimization problem of minimizing the sum of square differences between the actual image and the theoretical image. They add a regularization term which is the $\ell_0$-"norm"[1] over all pixel values. The problem then becomes NP-hard [28], so to solve it, they use the *continuous exact* $\ell_0$ (CEL0) penalty which is a continuous relaxation with some useful properties. This is in contrast with the $\ell_1$ norm, which is commonly used as a convex surrogate of the $\ell_0$-pseudonorm.

A very similar approach was taken by Bechensteen et al. [3]. The general setup is the same, except the relaxation of the $\ell_0$-pseudonorm is different. They formulate the $\ell_0$-pseudonorm for any $x \in \mathbb{R}^n$ as a minimization problem:

$$\|x\|_0 = \min_{\substack{-1 \leq u \leq 1 \\ u \in \mathbb{R}^n}} \|u\|_1 \text{ s.t. } \|x\|_1 = x^T u$$

Note that $u_i = 1$ if $x_i > 0$ and $u_i = -1$ if $x_i < 0$ to make the inner product equal to $\|x\|_1$. It follows that all other entries of $u$ become 0 to minimize its $\ell_1$ norm. Bechensteen et al. now substitute the $\lambda\|x\|_0$ term in the optimization function with $\lambda\|u\|_1$ and add the constraint that $x^T u = 1$. The problem then becomes continuous and biconvex. This means that for a fixed $x$, the optimization problem in $u$ is convex and vice versa. They replace this constraint with an additional regularization term $\rho(\|x\|_1 - x^T u)$ and show that for a sufficiently large $\rho$, the solution is equal to the biconvex problem. They solve the optimization problem by increasing the value of $\rho$ in steps.

## 4.6 Performance measures

A great number of studies have been done on the performance of localization algorithms and most papers presenting a new algorithm, compare it to previous algorithms. There are different methods of measuring the performance and in this section we will discuss the most common ones in the literature.

Since most algorithms output a list of localizations, we can base an accuracy measure on these locations compared to the ground truth. Matching localizations with ground truths is non-trivial, since it can happen that two localizations lie close to a single ground truth or that a localization is not close to any ground truth.

[1]The $\ell_0$-norm is not a norm, since it does not fulfill homogeneity property. Sometimes, it is also called a pseudonorm.

In [35], Sage et al. use a bipartite matching between the localizations and the ground truths that minimizes the sum of the distances. Then, they reject matchings that lie too far apart.

For the low-density algorithms, a useful measure is the Cramér-Rao bound. Most emitters can be detected due to their sparsity, so we are then interested in the average distance between the localizations and the ground truths. Methods that approach the Cramér-Rao bound are considered close to optimal, since it is generally impossible to receive better accuracy. In a higher density setting, it is more likely that an algorithm misses a fluorophore entirely when it is overlapping with other fluorophores. We can then use the Jaccard index defined as

$$\frac{TP}{TP+FP+FN},$$

where $TP$, $FP$ and $FN$ stand for the number of true positives, false positives and false negatives, respectively. This measure punishes both finding incorrect localizations and missing ground truths. In [9], Gazagnes et al. use the same matching as in [35] with a maximum distance $\Delta$. Any matched localization is then a true positive. Unmatched localizations count as false positives and unmatched ground truths as false negatives.

In some situations, it is less logical to compare the end result to ground truth locations. The Deep-STORM [29] algorithm outputs an image directly, so instead, Nehme et al. opt to compare normalized mean squared error (NMSE) of the ground truth image with the computed image.

## 5 Preprocessing steps

Before we start the localization process, we want to perform three preprocessing steps. The first step is transforming the digital units back to photon counts. Since we assumed these are known parameters determined by the microscope, this is a simple direct computation. The other two steps are estimating the background and estimating the width of the PSF.

### 5.1 Background estimation

To estimate the background, we consider the assumption we made on it. In Section 3.4 we assumed that the background consists mainly of spurious photons which we modeled as a Poisson random variable with parameter $\beta$ which is constant over all pixels. The goal of the background estimation is to estimate this value $\beta$. Simply computing the mean pixel intensity would produce an accurate estimator, if all pixels follow the Poisson distribution with mean $\beta$. This would in fact be the MLE for $\beta$ [33, Chapter 8]. However, this estimations certainly contains bias, since emitters themselves will bring up the average pixel intensity, especially when the molecule density is large. A better approach is to look at the histogram of the image and try to find the peak of this distribution. Figure 3 shows the histogram of a low density image with background parameter $\beta = 60$ in both a regular and logarithmic plot. Even though the plot on the left resembles the probability mass function of a Poisson variable, the plot on the right reveals the extra pixels from the emitters. This is even more prevalent in Figure 4 where we show the plots for a high density image. Even with many emitters, the fraction of pixels that solely contain background will still outweigh the number of pixels containing signal of emitters. Because pixel values are integer, we could simply determine the most frequent pixel intensity, or the *mode intensity.* This however would also lead to bias, since we would always get an integer estimation of $\beta$. To remedy this, we combine both ideas and average the pixels in small interval of intensities centered around the mode intensity. Formally, we define the following estimator $\widehat{\beta}$. Let $I$ be an image in $\mathbb{N}^{WHT}$. We define $M_B = \arg\max_{n\in\mathbb{N}} \mathcal{H}(I)_n$ where we recall that $\mathcal{H}(I)$ is the histogram of the image. Although $M_B$ will generally be a singleton set, we define $M_B' = \frac{1}{|M_B|}\sum_{m\in M_B} m$ as the average of the elements in $M_B$. Then, we define $K_B = \{k \in [WHT] \mid |I_k - M_B'| < p_B\}$ where $p_B$ stands for 'background parameter' which determines how wide the interval of intensities is. Then, we estimate $\beta$ with

$$\widehat{\beta} = \frac{1}{|K_B|}\sum_{k\in K_B} I_k.$$

We do not use this estimation to do a background subtraction. Instead, we incorporate $\widehat{\beta}$ in the localization process later on.

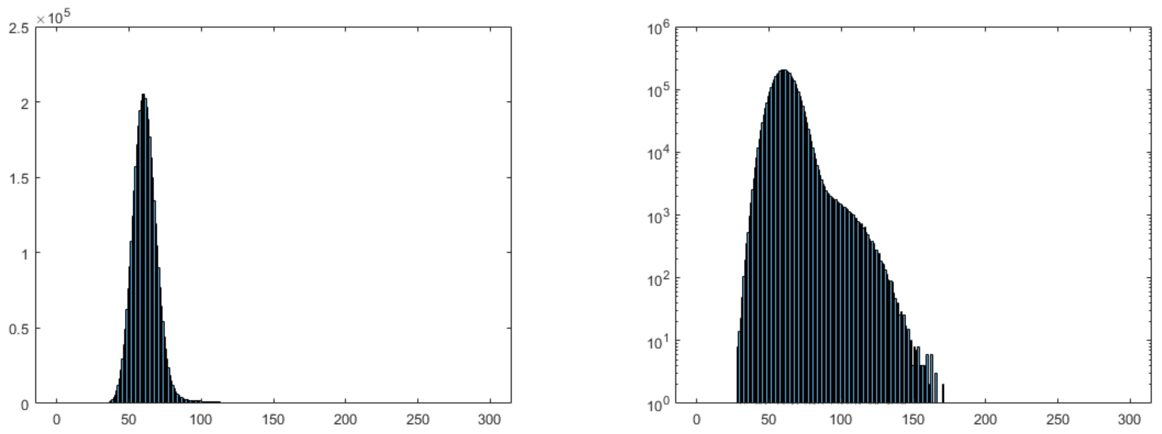


Figure 3: The histograms of a low density data set with $\beta = 60$. On the left the normal plot is shown and on the left the log-linear plot.

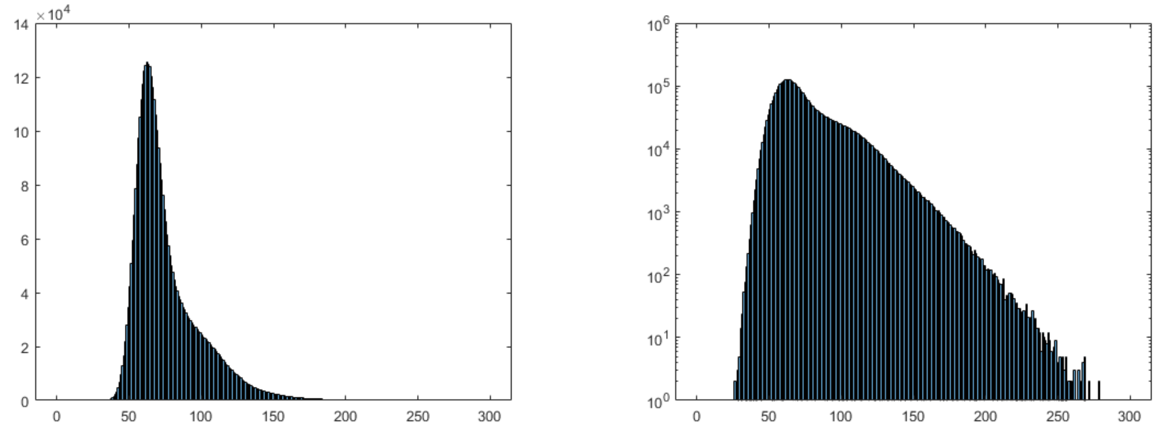


Figure 4: The histograms of a high density data set with $\beta = 60$. On the left the normal plot is shown and on the left the log-linear plot.

This method breaks down when the molecule density becomes large enough that most pixels will contain photon contributions from an emitter. However, those situations we want to avoid regardless, since at such high densities localization becomes very difficult.

## 5.2 FWHM estimation

In this section we assume we already have a set $D$ of detections, small regions of an image containing at least one emitter. In the entire algorithm we perform this step twice, once for the estimation of the width of the PSF and once to detect emitters for localization. The reason for this double work is that we need the width of the PSF to search for local maxima in a correctly sized region. However, to estimate this width, we need detections. To break this dependency we perform the detection step in this section with a fixed width of the subregion. This width may be non-optimal for maximizing the number of emitters detected, but that is for estimating the PSF width not relevant yet. We address how we find these detections in Section 6.1.

We assumed that the shape of our PSF is a two-dimensional Gaussian distribution with a fixed width. However, we still need this width for our localization procedure. We could include a parameter $\sigma$ for each emitter in our parameter space, but under the assumption that $\sigma$ is constant for all emitters, we can also try to estimate $\sigma$ beforehand.

One such way is to compute the *full width at half maximum (FWHM).* The FWHM is a way to quantify the width of a peak in a graph. Intuitively, the FWHM is defined as the length of the intervals, where the function is above half its maximum. Usually, we only consider functions with a single peak such as in Figure 5. Now, we give a formal definition.

**Definition 5.2.1** (FWHM)**.** Let $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be some bounded, (Riemann) integrable function with $M := \sup_{x\in\mathbb{R}} f(x)$. Then, we define

$$FWHM(f) = \int_{-\infty}^{\infty} \mathbb{1}_{\{f(x)\geq M/2\}} \,\mathrm{d}x.$$

**Proposition 5.2.2.** *Let $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a function like above with $FWHM(f) = t$. Then,*

1. *it holds that $FWHM(cf) = t$ for any $c \in \mathbb{R}_{>0}$ (scale-invariant);*
2. *it holds that $FWHM(g) = t$, where $g(x) = f(x + a)$ for $a \in \mathbb{R}$ (shift-invariant).*

*Proof.* Let $M := \sup_{x\in\mathbb{R}} f(x)$. Trivially, it holds that $\sup_{x\in\mathbb{R}}(cf)(x) = cM$. Then, we have that

$$FWHM(cf) = \int_{-\infty}^{\infty} \mathbb{1}_{\{(cf)(x)\geq cM/2\}} \,\mathrm{d}x = \int_{-\infty}^{\infty} \mathbb{1}_{\{f(x)\geq M/2\}} \,\mathrm{d}x = FWHM(f) = t.$$

Futhermore, via coordinate substitution $x' := x + a$ it holds that

$$\begin{aligned} FWHM(g) &= \int_{-\infty}^{\infty} \mathbb{1}_{\{g(x)\geq M/2\}} \,\mathrm{d}x = \int_{-\infty}^{\infty} \mathbb{1}_{\{f(x+a)\geq M/2\}} \,\mathrm{d}x \\ &= \int_{-\infty}^{\infty} \mathbb{1}_{\{f(x')\geq M/2\}} \,\mathrm{d}x' = FWHM(f) = t. \end{aligned}$$

□

**Example 5.2.3.** Since we assume our PSF to be Gaussian, it is useful to explore the relation between the FWHM of the Gaussian density function and the variance parameter $\sigma^2$. By the shift invariance property we can consider a Gaussian random variable with mean 0 and variance $\sigma^2$. The density function has maximum

$$f(0) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{0^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma}.$$

The half maximum is attained when $e^{-x^2/(2\sigma^2)} = 1/2$ which happens if $x^2/(2\sigma^2) = \log 2$. It follows that $x = \pm\sqrt{2\log 2}\,\sigma$. The distance between these two points is then computed as $2\sqrt{2\log 2}\,\sigma$. △

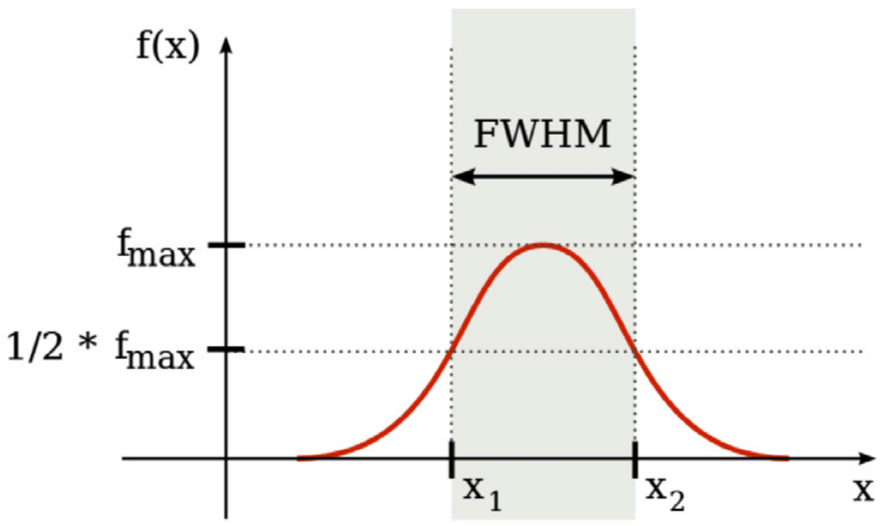


Figure 5: Visualization of the full width at half maximum. Image taken from [22].

Now, suppose we have a subimage containing precisely one emitter. We can estimate the FWHM as follows. We find the pixel containing the centroid of the subimage. That is, the average $x$- and $y$-coordinate weighted by the pixel values. Then, we consider its value as the maximum. Note that taking the maximum pixel value in the subimage would lead to bias, as the highest intensity pixel will, on average, be larger than the true maximum, due to shot noise. But even if we consider the centroid pixel, we are likely to have a poor estimate of the maximum value. To reduce variance, we can take a (weighted) average over a small region around the centroid pixel at the cost of a small bias, since the average intensity of these pixels will be slightly smaller than the true maximum intensity. To compensate for this bias, one could do a small Gaussian fit on these pixels, but we want to keep this step computationally easy as the goal of estimating the FWHM was to reduce the number of variables in the eventual fit procedure.

From the centroid pixel we walk in the left and right direction over our image until the pixel intensity drops below half of the FWHM and count the number of pixels above it. We do the same for the up and down direction and this gives us two estimations of the FWHM for each detection. Averaging these values over all detections results in a stable estimation of the FWHM. Note that FWHM is independent of the intensity of an emitter by the scale invariance property.

### 5.2.1 In multi-emitter setting

If we cannot assume a subimage contains exactly one emitter, then the above estimation will no longer be accurate. Firstly, the centroid pixel can be a very bad estimation for the maximum intensity, since this pixel can lie precisely between two emitters. For this reason we choose the maximum intensity pixel as starting point instead of the centroid pixel. Again, we can take a weighted average of a small area around this pixel to estimate the true maximum intensity.

Secondly, simply counting the pixels in $x$- and $y$- direction will sometimes overestimate the FWHM. For example, suppose we have an emitter roughly in the center and a second emitter lying right next to it. We perform the same method as above where we go in one direction until we arrive at a pixel below half the maximum. However, instead of considering the average over all these values, we now determine the histogram of all FWHM estimations (the estimations are all integer). Then, we follow the same idea of the background estimation where we determine the maximum bin $M$ and take the average of the FWHM estimates close to $M$.

The process can be summarized in the following steps. From $D$, we compute the multiset (set with duplicates) $E = E(D)$ with estimations of the FWHM. Then, with some abuse of notation, we compute the histogram $\mathcal{H}(E)$ of $E$. Similar to the background estimation, we compute $M_F = \arg\max_{n\in\mathbb{N}} \mathcal{H}(E)_n$ and subsequently, $M'_F = \frac{1}{|M_F|}\sum_{m\in M_F} m$. Then, we define $K_F = \{e \in E(D) \mid |e - M'_F| < p_F\}$ where $p_F$ is the

'FWHM parameter'. In the end, we compute

$$\widehat{FWHM} = \frac{1}{|K_F|} \sum_{e \in K_F} e$$

as an estimation for the FWHM of the PSF. Then, we compute $\widehat{\sigma} = \widehat{FWHM}/(2\sqrt{2\log 2})$.

## 6 Individual emitter localization

In this section we describe our theory and methods for low-density emitter localization. We follow the two-step structure often used in the literature, consisting of detecting *regions of interests* (ROIs) and localizing emitters in these ROIs. We use the parameter estimations for the FWHM and the background from the previous section.

### 6.1 Molecule detection

The next step is to detect the molecules in the image. Generally, we want to look for local maxima as emitters will always contain a local maximum. However, areas without emitters can also contain a (low intensity) local maximum, due to random noise. We can prevent these false detections as follows. First, we determine a threshold value $\tau$ for a local maximum to be counted as a detection. Due to the randomness of the noise, it is not unlikely that there are some background pixels with a high intensity value above $\tau$. We cannot choose $\tau$ too large, since at some point we are excluding actual emitters. In order to create a larger gap between local maxima of emitters and local maxima of background pixels, we apply a uniform filter to our image. If we consider an image $I$ as a matrix, then the uniform filter of size $\alpha$ is a function $\mathcal{U}_\alpha : \mathbb{R}_+^{W\times H} \to \mathbb{R}_+^{W\times H}$ and it outputs for each pixel the average pixel value in the window around $p = (w, h)$. If we recall $\mathcal{N}_\alpha(w, h)$ as the neighborhood around pixel $p$, then it is defined as follows:

$$\mathcal{U}_\alpha(I)_{wh} = \frac{1}{|\mathcal{N}_\alpha(w,h)|} \sum_{(i,j)\in\mathcal{N}_\alpha(w,h)} I_{ij}.$$

A uniform filter smooths bright background pixels, since the surrounding pixels will, if they are background pixels as well, have an average intensity of the background. This results in average intensity close to the background intensity. On the other hand, a bright pixel in the center of an emitter will be surrounded by other (slightly less) bright pixels, so its averaged intensity will not decrease much.

If $I$ is an image series, then the uniform filter is applied to each separate frame. To reduce extensive notation, we will simply denote the blurred image by $I'$.

Now, we only need to choose the threshold value. For this we consider the distribution and the histogram of the pixel intensities. We can split the total set of pixels of an entire image series roughly in two sets, signal pixels, which contain both background and emitter intensity, and background pixels, which only contain background noise. We denote these set by $A$ and $B$ respectively. The ratio between the sizes of these two sets will vary with the density, but as long as the density is not extremely large we will have that $|A|/|B| << 1$. The pixels in $B$ have intensities that are all realizations of the same Poisson distribution with parameter $\beta$. The intensities of the pixels in $A$ are realizations of all slightly different Poisson distributions with parameter $\beta + \alpha_k$ where $\alpha_k$ is the parameter of the Poisson variable of the intensity contributed by one or more emitters to pixel $k$. When the number of background pixels is sufficiently large compared to the number of signal pixels, we have relatively many realizations of the same discrete random variable and the histogram of the image series will approximate the probability mass function of this random variable for the lower intensity pixels (see Figure 3).

In an ideal situation, where all emitters are nicely separated, the histogram of all signal pixels will be independent of the density of the image.[2] The goal is then to remove the background pixels from the histogram and determine a suitable threshold value from just the signal pixels. From Section 5.1 we have an estimation

[2]Although density influences the separation of emitters, we want to emphasize that the number of signal pixels compared to the number of background pixels, does not influence the shape of the histogram of just the signal pixels.

$\widehat{\beta}$ of the background parameter $\beta$. We approximate the number of background pixels by considering the mode intensity $m$ again. We compute

$$p = \frac{\widehat{\beta}^m e^{-\widehat{\beta}}}{m!}$$

which approximates the probability that a random background pixel has intensity $m$. Then, the number of background pixels $B_p$ can be approximated by $N_b = \mathcal{H}(I)_m/p$. Note that in higher density pixels $\mathcal{H}(I)_m$ is smaller, so this accounts for different densities. Using this approximation, we can approximate the expected number of background pixels for the other intensities. That is, we compute the values $H_i^b = N_b \cdot \frac{\widehat{\beta}^i e^{-\widehat{\beta}}}{i!}$ for each $i \in \mathbb{N}$. Then, we compute $H_i^s = \max(0, \mathcal{H}(I)_i - H_i^b)$ as an approximation for the number of signal pixels of a given intensity $i$. As a threshold value we choose the median value of the signal pixels. Note that $H_i^s$ is most likely not integer. However, we can still compute the median intensity of this 'histogram'. We compute the threshold value $\tau$ as follows:

$$\tau := \min_{j \in \mathbb{N}} \left\{ \sum_{i=0}^{j} H_i^s > \frac{1}{2} \sum_{i=0}^{\infty} H_i^s \right\}$$

This may seem like a fairly low threshold value leading to many extra detections, but recall that we look for local maxima in the blurred image. Pixels in the center of an emitter will approximately still have the same intensity, since they are surrounded by other pixels with a high average intensity, while high intensity pixels on the edge of emitters or in the background decrease in intensity, since they are surrounded by pixels with a low average intensity.

We can improve this threshold value. A small fit on the lower intensities instead of simply considering mode intensity will better estimate the number of background pixels. Incorporating an estimate on the molecule density can improve it even further. We chose the above method mainly for simplicity and because the threshold parameter is less important than the fit parameters like the FWHM and the background.

Finally, we define the set of candidate pixels $D$. Let $I'$ be the blurred image. From $I'$ we compute pixels with intensity above $\tau$ that are a local maximum. A pixel is considered a local maximum if it takes on the maximum value in the neighborhood around it. That is, we are determining the set

$$D := \left\{ p \in [W] \times [H] \times [T] \;\middle|\; I'(p) \geq \tau, I'(p) = \max_{(w',h') \in \mathcal{N}_\rho(p)} I'(w', h', t) \right\}.$$

**Remark 6.1.1.** One might wonder why we do not consider the original image when determining whether a pixel is a local maximum. The reason for this is that the maximum intensity pixel of an emitter might be, due to shot noise, more towards the edge of the fluorophore instead of the middle. If we were to select part of the image around this pixel, the fluorophore is not nicely centered in the middle of the subimage. Taking the blurred image smooths outliers on the edge and the maximum is more likely to be closer to the center of the fluorophore.

## 6.2 Localization

When we fit individual emitters, we consider two optimization methods, least-squares fitting and maximum likelihood optimization. Now, we go into detail of these optimization functions and their gradients.

Generally, we are trying to fit three parameters for an emitter, two positional parameters $x$ and $y$, and the intensity $b$ (for brightness). We assume a two-dimensional Gaussian point spread function and we estimate the variance from the data. For each detection $d \in D$, we consider the neighborhood $\mathcal{N}_r(d)$ which we use for the localization. Since we only consider the neighborhood $\mathcal{N}_r(d)$ in the remainder of this section, we simply denote the restriction of $I$ to $\mathcal{N}_r(d)$ by $I_d := I|_{\mathcal{N}_r(d)}$. We choose $r$ such that most of the PSF is contained, being $r = \widehat{FWHM}/2 + 1$. For both optimization problems we compute the expected intensity, $\lambda_{\theta,k}^S$ of each pixel, given the parameters (see Section 3.4). For readability we denote it by $\lambda_k$. Suppose we are fitting $N$ emitters. Then, the set of parameters consists of $\theta := (x_1, y_1, b_1, \ldots, x_N, y_N, b_N)$. The expected intensity $\lambda_k$

of a pixel $k$ is then calculated as

$$\lambda_k = \sum_{i=1}^{N} b_i \int_{D_k} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left((x-x_i)^2+(y-y_i)^2\right)} \,\mathrm{d}x\,\mathrm{d}y + \beta$$

where $D_k$ is the surface of pixel $k$ and $\beta$ is the background. From this expression we can compute the partial derivatives as follows. For the intensity parameter we get

$$\frac{\partial \lambda_k}{\partial b_i} = \int_{D_k} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left((x-x_i)^2+(y-y_i)^2\right)} \,\mathrm{d}x\,\mathrm{d}y.$$

The location parameters are slightly more complicated. Let $D_k = [l_k, r_k] \times [t_k, b_k]$. Then, we can calculate the partial derivatives with respect to $x_i$ and $y_i$. Because the PSF is a two-dimensional Gaussian, we define $f_{x_i}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_i)^2}{2\sigma^2}}$ and $f_{y_i}(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-y_i)^2}{2\sigma^2}}$ as one-dimensional probability density functions and let $F_{x_i}$ and $F_{y_i}$ be the corresponding distribution functions. With these we can rewrite the integral over $D_k$ as follows:

$$\begin{aligned}\int_{D_k} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left((x-x_i)^2+(y-y_i)^2\right)} \,\mathrm{d}x\,\mathrm{d}y &= \int_{t_k}^{b_k}\int_{l_k}^{r_k} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left((x-x_i)^2+(y-y_i)^2\right)} \,\mathrm{d}x\,\mathrm{d}y \\ &= \int_{l_k}^{r_k} f_{x_i}(x)\,\mathrm{d}x \int_{t_k}^{b_k} f_{y_i}(y)\,\mathrm{d}y.\end{aligned}$$

Now, we can compute the partial derivative with respect to $x_i$ as follows:

$$\frac{\partial \lambda_k}{\partial x_i} = b_i \int_{t_k}^{b_k} f_{y_i}(y)\,\mathrm{d}y \cdot \frac{\partial}{\partial x_i}\int_{l_k}^{r_k} f_{x_i}(x)\,\mathrm{d}x$$

To solve the remaining partial derivative, we apply the substitution $x' = x - x_i$ and use the Leibniz integral rule. Then, we have that

$$\begin{aligned}\frac{\partial}{\partial x_i}\int_{l_k}^{r_k} f_{x_i}(x)\,\mathrm{d}x &= \frac{\partial}{\partial x_i}\int_{l_k}^{r_k} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_i)^2}{2\sigma^2}}\,\mathrm{d}x \\ &= \frac{\partial}{\partial x_i}\int_{l_k-x_i}^{r_k-x_i} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x'^2}{2\sigma^2}}\,\mathrm{d}x' \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_k-x_i)^2}{2\sigma^2}} \cdot (-1) - \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(l_k-x_i)^2}{2\sigma^2}} \cdot (-1) \\ &\quad + \int_{l_k-x_i}^{r_k-x_i} \frac{d}{dx_i}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x'^2}{2\sigma^2}}\right)\mathrm{d}x \\ &= f_{x_i}(l_k) - f_{x_i}(r_k).\end{aligned}$$

Using an analogous reasoning for the derivative with respect to $y$, we get the following partial derivatives:

$$\begin{aligned}\frac{\partial \lambda_k}{\partial x_i} &= b_i \left(F_{y_i}(b_k) - F_{y_i}(t_k)\right)\left(f_{x_i}(l_k) - f_{x_i}(r_k)\right) \\ \frac{\partial \lambda_k}{\partial y_i} &= b_i \left(f_{y_i}(t_k) - f_{y_i}(b_k)\right)\left(F_{x_i}(r_k) - F_{x_i}(l_k)\right)\end{aligned}$$

We can use the above partial derivatives to compute the partial derivatives for the optimization functions in MLE and LS estimation.

**MLE** For MLE we use the fact that the pixel intensity is a Poisson random variable. We get the likelihood function

$$\mathcal{L}(\theta \mid I_d) = \prod_{k\in\mathcal{N}_\rho(d)} \frac{\lambda_k^{I_d(k)} e^{-\lambda_k}}{I_d(k)!}.$$

Taking the logarithm, we obtain

$$\begin{aligned}\ell(\theta \mid I_d) &= \log \prod_k \frac{\lambda_k^{I_d(k)} e^{-\lambda_k}}{I_d(k)!} \\ &= \sum_k \log \frac{\lambda_k^{I_d(k)} e^{-\lambda_k}}{I_d(k)!} \\ &= \sum_k \left(I_d(k) \log \lambda_k - \lambda_k - \log I_d(k)!\right)\end{aligned}$$

Since $\log I_d(k)!$ is constant, the *maximum* likelihood estimation can be formulated as the following *minimization* problem:

$$\min_\theta L_{\text{MLE}}(\theta) := \min_\theta \sum_k (\lambda_k - I_d(k) \log \lambda_k)$$

The partial derivative of $L_{\text{MLE}}(\theta)$ with respect to any variable $\theta_i$, where $\theta_i \in \{x_1, y_2, b_1, \ldots, x_N, y_N, b_N\}$, can be computed as

$$\begin{aligned}\frac{\partial}{\partial \theta_i} L_{\text{MLE}}(\theta) &= \frac{\partial}{\partial \theta_i} \sum_k (\lambda_k - I_d(k) \log \lambda_k) \\ &= \sum_k \left(\frac{\partial \lambda_k}{\partial \theta_i} - \frac{I_d(k)}{\lambda_k} \cdot \frac{\partial \lambda_k}{\partial \theta_i}\right) \\ &= \sum_k \left(1 - \frac{I_d(k)}{\lambda_k}\right) \frac{\partial \lambda_k}{\partial \theta_i}.\end{aligned}$$

**LS** If we use least squares optimization, we try to minimize the distance between the image and the expected image from the $\lambda_k$. The measurements are simply the measured pixel intensities. The functions we denoted by $f_i$ in Section 3 are in this case the expected intensities $\lambda_k$. For pixel $k$ the residual becomes $r_k = I_d(k) - \lambda_k$. We get the following optimization problem:

$$\min_\theta L_{\text{LS}}(\theta) := \min_\theta \sum_k (I_d(k) - \lambda_k)^2$$

Although we will only need the partial derivatives of the $r_k$, we also present, for completeness, the partial derivative of $L_{\text{LS}}(\theta)$ with respect to any variable $\theta_i$

$$\begin{aligned}\frac{\partial}{\partial \theta_i} L_{\text{LS}}(\theta) &= \frac{\partial}{\partial \theta_i} \sum_k (I_d(k) - \lambda_k)^2 \\ &= -\sum_k 2(I_d(k) - \lambda_k) \frac{\partial \lambda_k}{\partial \theta_i}\end{aligned}$$

## 6.3 Fit procedure

The above optimization problem can be solved using a gradient-based iteration algorithm. Now, we formulate our method. Although the method allows for an arbitrary number of emitters to be fitted, we limited ourselves to two. This already results in a sufficiently complex algorithm. However, we do not know how many emitters are in a certain subimage. So, for each detection we perform both a 1-emitter fit and a 2-emitter fit. Then, we determined which solution is the most reliable.

### 6.3.1 Initial guess

The result of an iterative optimization algorithm can vary greatly between different initial guesses, since our optimization functions are non-convex [5, Section 1.4]. In the case of fitting one emitter we can choose our initial guess close to the actual solution by choosing a high-intensity pixel. A reliable optimization method will move to the optimal value from there. However, when fitting multiple emitters, we run in a problem. Suppose, for example, that we have two emitters that slightly overlap and we choose our initial two locations in the same fluorophore. Because the optimization function is non-convex, the two locations can converge to the emitter location of the fluorophore they start in. Although in the global minimum the two locations are at both emitters, this requires one of the two locations to move from one fluorophore to the other. This can require going to a worse solution first resulting in a non-optimum local minimum. On the other hand, if we chose our initial guess in such a way that both fluorophores contained a location, we probably would find the correct solution. Enforcing this is difficult, since we initially do not know where the emitters are. A simple solution is to use random starting positions. To improve the guesses we use the image itself as a distribution for guesses, since higher intensity pixels are more likely to contain an emitter. That is, let $X$ be a random variable taking values in $\mathcal{N}_r(d)$ with probability mass function $p_X(k) = I_d(k)/\left(\sum_{j\in\mathcal{N}_r(d)} I_d(j)\right)$ and $k \in \mathcal{N}_r(d)$. Then, we take two independent realizations of $X$ as initial solutions.

### 6.3.2 Combining results

Now we have a for each detection $d \in D$ a 1-fit $\theta_1^d = (x_1, y_1, b_1) \in \Theta_1 =: \mathbb{R}^2 \times \mathbb{R}_+$ for one emitter and a 2-fit $\theta_2^d = (x_{21}, y_{21}, b_{21}, x_{22}, y_{22}, b_{22}) \in \Theta_2 =: \mathbb{R}^2 \times \mathbb{R}_+ \times \mathbb{R}^2 \times \mathbb{R}_+$ for two emitters. We have to decide which emitters we return. We could simply compare the values of the optimization function, but the 2-fit simply has more degrees of freedom. For each solution $\theta_1^d \in \Theta_1$ we can find a solution in $\Theta_2$ with an equal optimization value. If we choose $(x_{21}, y_{21}, b_{21}, x_{22}, y_{22}, b_{22}) = (x_1, y_1, b_1, 0, 0, 0)$ then the optimization functions of both LS and MLE will return the same value. So, unless we pick a bad initial solution, the final solution of the 2-fit will always be at least as good as the one from the 1-fit. For this reason we chose a decision tree approach, where we try to incorporate additional information on the model. For example, if the 1-fit has an unusual large intensity, then it is likely we have actually two emitters close to each other. On the other hand, if the two localizations of the 2-fit are very close to each other and at least one has an unusual small intensity, then it is likely that the algorithm is just trying to fit two emitters while we only have one. Lastly, if either of the two emitters is placed far outside the ROI, then we have probably one emitter as well. To detect unusual intensities, we first perform all 1-fits on the detections. If we denote the 1-fit of detection $d$ by $(x^d, y^d, b^d)$, we define the (multi)set $B = \{b^d \mid d \in D\}$. We define unusual intensities using the following boundaries. We define $b_{\min} = \min(B)$ and $b_{\max} = \max(B)$. We also define $b_{\text{low}} = \min_{b\in B}\{|\{b' \in B \mid b' < b\}| > 0.05 \cdot |B|\}$ and $b_{\text{high}} = \max_{b\in B}\{|\{b' \in B \mid b' > b\}| > 0.05 \cdot |B|\}$. Now, we use the following decision tree procedure:

1. If $d(p_1, p_2) < 1$, choose 1-fit;
2. else if $b_{21} \notin [b_{\min}, b_{\max}]$ or $b_{22} \notin [b_{\min}, b_{\max}]$, choose 1-fit;
3. else if $b_1 > b_{\text{high}}$, choose 2-fit;
4. else if $d(p_1, p_2) < 2$ and either $b_{21} < b_{\text{low}}$ or $b_{22} < b_{\text{low}}$, choose 1-fit;
5. else choose solution with smallest optimization value.

Now we argue the above decision tree step-by-step. In step 1 we use the fact that there is a minimum distance at which we can confidently separate two emitters. If we find $p_1$ and $p_2$ closer to each other, we decide it is simply one emitter.

In step 2 we compare the brightnesses of the 2-fit with the minimum and maximum 1-fit brightnesses. This step is to filter two situations where we fit two emitters on a region containing one. In this situation one of the two emitters represents the actual emitter while the other is either far away and very bright to get a slightly better optimization value than the 1-fit or very dim and somewhere inside the region to also slightly decrease the optimization function. In both situations the second emitter does not represent a true emitter, but is simply a result of trying to fit two emitters where there is only one emitter.

In step 3 we use the observation that if $b_1$ is an outlier in $B$, much brighter than the average intensity of an emitter, then it is quite likely we simply have two emitters that have a large overlap, leading to a bright spot.

In step 4 we do a similar test to step 1 and 2, but with a different motivation. When the 2-fit is placed at a small distance, but not so small that it is impossible to distinguish between one or two emitters, we check in addition if either of the 2-fit emitters is unusually dim. In that case it is likely that there is only one emitter and that the position of the 2-fit emitters minimizes the optimization function better than two emitters at the same position.

Finally, if none of the above situations apply, we consider the optimization value. In most cases the 2-fit will be better, but if we are unlucky with the initial positions, we might find a larger optimization value than the one from the 1-fit. If this is the case, it means that the 2-fit is not optimal. One could then try different solutions, until we find one that is better, but for practical reasons, we simply choose the 1-fit.

If we process every detection $d \in D$, we end up with a total set of localizations $L$. As a final postprocessing step, we filter duplicate localizations. Especially when we have overlapping emitters it is possible that we find the same emitter in different detections. We go through each localization $\ell$ and determine all localizations in the same frame that are within a distance of one pixel. Then, for each of these localizations we compare the optimization function to the one of $\ell$ and remove the worst one (the largest optimization value). After this step there are no localizations within pixel distance from each other.

Pseudocode of the entire algorithm is presented in Algorithm 1. For the sake of readability we left out the user-chosen parameters which should be defined at the start and passed to functions.

**Algorithm 1** Individual emitter localization

```
Input: $I \in \mathbb{R}^{[W]\times[H]\times[T]}$
  $\widehat{\beta} \leftarrow$ computeBackground($I$)                    (See Section 5.1)
  $\tau \leftarrow$ computeThreshold($I$,$\widehat{\beta}$)
  $I' \leftarrow$ blurImage($I$)
  $D_{FWHM} \leftarrow$ findDetections($I'$, $\tau$)
  $\widehat{FWHM} \leftarrow$ computeFWHM($I$,$D_{FWHM}$)       (See Section 5.2)
  $\widehat{\sigma} \leftarrow \widehat{FWHM}/(2\sqrt{2\log 2})$
  $\rho \leftarrow \widehat{FWHM}/2$
  $D \leftarrow$ findDetections($I'$,$\tau$, $\rho$)                 (See Section 6.1)
  $L \leftarrow \emptyset$
  for $d \in D$ do
      $(x_1, y_1, b_1) \leftarrow$ 1Fit($I$,$d$,$\tau_2$,$\widehat{\beta}$,$\widehat{\sigma}$)
      $L_1(d) \leftarrow \{(x_1, y_1, b_1)\}$
  end for
  $B := (b_{\min}, b_{\max}, b_{\text{low}}, b_{\text{high}}) \leftarrow$ calculateBrightnessBounds($L_1$)
  for $d \in D$ do
      $(x_{21}, y_{21}, b_{21}, x_{22}, y_{22}, b_{22}) \leftarrow$ 2Fit($I$,$d$,$\tau_2$,$\widehat{\beta}$,$\widehat{\sigma}$)
      $L_2(d) \leftarrow \{(x_{21}, y_{21}, b_{21}, x_{22}, y_{22}, b_{22})\}$
  end for
  for $d \in D$ do
      $L \leftarrow L \cup$ ChooseFit($L_1(d)$,$L_2(d)$,$B$)
  end for
  $L \leftarrow$ filterDuplicates($L$)
  return $L$
```

## 7 Multi-emitter grid localization

In this section we take a different approach from the one in the previous section. This approach is loosely based on the paper of Gazagnes et al. [9] and more suited for high-density data compared to individual emitter fitting. The reason for this is that it does not suffer from the issue in individual fitting where we can have other emitters partly in the ROI of a detection. For simplicity we consider a single-frame image $I$ where $W = H = N$. We emphasize that a square image is not required, but is simply assumed to simplify the complexity analysis. Moreover, we view $I \in \mathbb{R}_+^{N^2}$ as a vector.

Contrary to the previous section, we assume that the emitters lie on some finer grid of size $NL \times NL$. That is, emitter positions are assumed to be in a discrete set of locations instead of a continuous space. We call the elements of this finer grid *subpixels*. This situation is illustrated in Figure 6. Just as with individual emitter fitting, we first estimate the FWHM and the background $\widehat{\beta}$ from Section 5.1. Then, recall from Example 5.2.3 that we can compute the standard deviation of a Gaussian distribution from the FWHM. So, we use the estimate $\widehat{\sigma} = \widehat{FWHM}/(2\sqrt{2\log(2)})$.

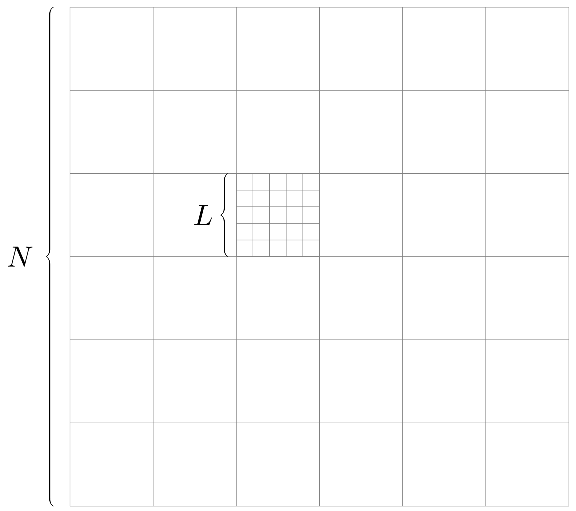


Figure 6: An $N \times N$ image with one pixel subdivided in an $L \times L$ grid of subpixels.

In this setting we allow each subpixel to have an emitter. So, our variable vector becomes $x \in \mathbb{R}_+^{N^2L^2}$. Here $x_i$ stands for the intensity of the emitter at subpixel $i \in [N^2L^2]$ where we interpret 0 intensity as the absence of an emitter. The contribution of an emitter at position $i \in [N^2L^2]$ to pixel $k \in [N^2]$ is computed similarly as before as the two-dimensional Gaussian with fixed variance $\sigma^2$ positioned in the center of the subpixel. We denote the integral of the Gaussian pdf centered at subpixel $i$ over the area of pixel $k$ by $A_{ik}$ where we use the estimation of the FWHM to approximate the parameter $\sigma$. That is, we define

$$A_{ik} = \int_{D_k} f_i(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_{D_k} \frac{1}{2\pi\widehat{\sigma}^2} e^{-\frac{1}{2\widehat{\sigma}^2}\left((x-x_i)^2+(y-y_i)^2\right)}\,\mathrm{d}x\,\mathrm{d}y$$

where $D_k$ is again the area of pixel $k$ and $(x_i, y_i)$ is the center of subpixel $i$. Then, the expected intensity of pixel $k$ can be calculated as $\lambda_k(x) = \sum_{i=1}^{N^2L^2} A_{ik}x_i + \widehat{\beta}$ where $x_i$ is the intensity of the emitter at position $i$. Gazagnes et al. use this approach and then try to minimize the LS error over $x$.

*Details of this section removed by embargo.*

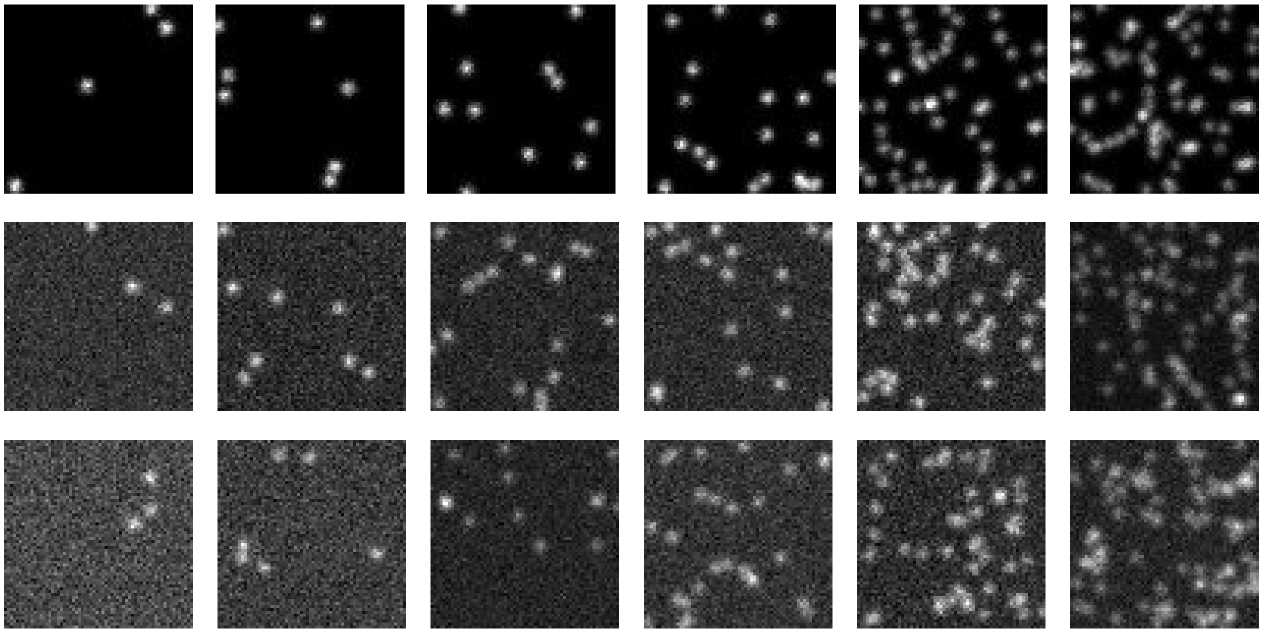


Figure 7: The first frame of each of the datasets. Each row contains images with densities 0.1, 0.2, 0.3, 0.5, 1.0 and 2.0 and each column contains the three different background levels 0, 30 and 60.

## 8 Experiments

The next step is to evaluate our two algorithms. In this section we explain the exact steps of this analysis. We start with the generation of the data sets. Then, we discuss the metrics and how we compute them. Finally, we analyze the results from our algorithms and compare them to each other and to the ThunderSTORM algorithm [32].

### 8.1 Data sets

To evaluate the algorithms, we use the interactive ImageJ plugin ThunderSTORM [32] to generate data sets and their ground truth (GT) locations. With this tool we can freely determine the parameters of the image, most importantly, the noise level and the molecule density. We generate several data sets consisting each of 1000 frames of size $64 \times 64$. The pixel size is set to 100nm. We add a baseline of 100 digital units and simply set the number of electrons per digital unit to 1.[3] We fix the FWHM at 350, as we assumed this in our algorithm to be fixed, and chose the intensity range to be $[700, 900]$.

The parameters we varied are the level of background noise and the molecule density. We choose the set of backgrounds to be $\{0, 30, 60\}$ where these values represent the Poisson parameter $\beta$ from Section 5.1. For each of these background noise levels we generate six image sets for the densities $\{0.1, 0.2, 0.3, 0.5, 1.0, 2.0\}$ where these values stand for the average number of molecules per $\mu$m. The first frame of each data set is shown in Figure 7.

### 8.2 Metrics

Before we discuss the metrics we will use, we need to match localizations to ground truth locations. We denote the set of localizations and GT locations by $A$ and $B$ respectively. The goal now is to find a matching between $A$ and $B$. That is, we want to find a set $C$ containing pairs $(a, b)$ with $a \in A$ and $b \in B$ such that each $a$ and $b$ occur at most once in $C$. Of course, we cannot choose any matching. We have to choose a criterion such that it is reasonable that a localization corresponds to the matched GT location. A sensible choice is to choose a maximum tolerance distance $r$ to count as a valid pair and try to find a matching that

[3]For an explanation on these parameters, see Section 3.3.

pairs as many localizations and locations together. There are several algorithms to compute such a matching. We will utilize the matching of the ThunderSTORM plugin [32]. They use the Gale-Shapley algorithm which requires a preference list of the set $B$ for each $a \in A$ and vice versa. It then finds a stable matching between localizations and locations. This means that there are no $a \in A$ and $b \in B$ that are not matched together, but prefer each other over their current match. In the case of ThunderSTORM the preference is determined by the distance where closer locations or localizations have a preference over ones that are further away.

**Remark 8.2.1.** The choice for the Gale-Shapley algorithm is an odd one, since there is one particular case where Gale-Shapley fails, even though Ovesný et al. emphasize that in that situation Gale-Shapley outperforms a greedy solution. This situation is shown in Figure 8. In this figure we have GT locations (red dots) and localizations (blue crosses) that we need to match together. The authors explicitly claim that Gale-Shapley finds the matching in d) as opposed to the one in c). However, this is not a stable matching, since the GT location 1 and the localization B prefer each other. However, there is also an argument to be made that we should always match closest GT-localization pairs as opposed to finding the one that maximizes the number of matchings. Since we considered this matching outside of the scope of this thesis, we chose to simply use the matching algorithm from ThunderSTORM as a black box.

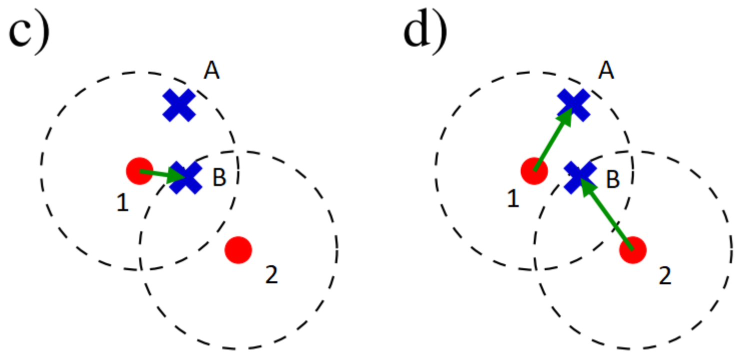


Figure 8: Image taken from the Supplementary notes of ThunderSTORM [32]. The authors claim that Gale-Shapley finds the matching in d) while it actually finds the matching in c).

Now we have our matching, we denote the number of matched localizations by $TP$ (for true positives), the number of unmatched localizations by $FP$ (for false positives) and the number of unmatched GTs by $FN$ (for false negatives). We can measure the performance of our algorithms with the following three metrics:

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP} \qquad Jaccard = \frac{TP}{TP + FN + FP}.$$

The Jaccard index punishes both outputting too many and too few localizations. This metric is well-suited to determine the detection step of an algorithm which is especially useful for high-density images or images with a lot of noise. The recall and precision rate can indicate whether an algorithm fails to find some emitters or places too many incorrect ones. In low-density low-noise images it can be expected that a reasonable algorithms will detect most emitters in each frame, but when the number of emitters per frame is high or there is a lot of background noise, finding the number of emitters and their rough locations might already be a challenge itself. The Jaccard index is then a useful metric.

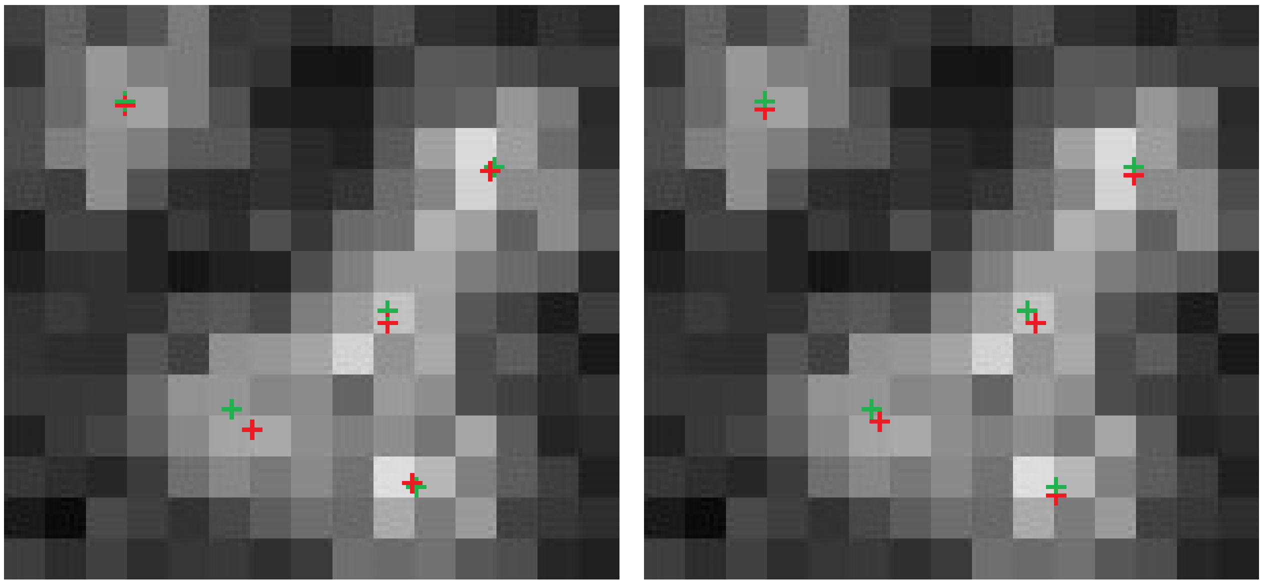
## 8.3 Results

Since we have many parameters, we have listed all parameter choices in Appendix A. For the individual fitting algorithm we chose the least squares optimization algorithm, since its results are more stable. The MLE approach suffers from some issues, if the model that is used does not match the actual situation. In particular, fitting the wrong number of emitters can lead to some bad results. For example, suppose we have a nicely centered emitter, but with a second emitter on the boundary. If we fit one emitter, putting the localization in the center of the image results in a very small likelihood, since the intensity of the second emitter is very unlikely with just one emitter. As a result, the algorithm will put the emitter somewhere in between the two emitters. The least squares method does not suffer as much from this issue, because placing a localization between the two emitters results in a larger penalty than just the penalty of ignoring the second emitter.

**Remark 8.3.1.** Before we present the results, we must make a few remarks. First, the grid fitting algorithm failed for unknown reasons on a large fraction of frames of the image series with molecule density 2. Most likely, this is due to a implementation bug and not a fault in the algorithm. Unfortunately, we could not find the exact reason for this behavior, so we left it out of the experiment. Secondly, since the algorithm is significantly slower than the individual emitter fitting algorithm, we limited the number of analyzed frames to 200.

Lastly, there are some data points missing in the tables and graphs. These are all due to bugs in the ThunderSTORM plugin. Some matchings could not be saved and the ThunderSTORM algorithm itself failed on some data sets.

Now we are ready to present the results of our two algorithms. We need to choose the maximum distance between a localization and a GT location. Since the performance is greatly dependent on this distance, we computed the results for distance 50nm and 100nm (0.5 pixel and 1 pixel respectively). The results of the experiments for both methods are presented in Tables 2, 3, 4 and 5 in Appendix B. We noticed that increasing the maximum distance from 50nm to 100nm greatly increases the number of true positives. This means that there are a significant number of localizations within a pixel distance, but not half a pixel distance. Upon further inspection these situations occurred most often when either the emitter was at the border of an image or when there at least two overlapping emitters and the algorithm did not find all of them. For example, if there are two emitters overlapping, but the algorithm decides there is only one there, it will place the emitter somewhere between the two emitters at a small distance from both. In these situations the algorithm did discover an emitter, but at slightly larger distance. For this reason we choose to use 100nm results to analyze further.



(a) Individual fitting (b) Grid fitting

Figure 9: Ground truth locations (green) and localizations (red) for a small region from the data set with density 1 and background level 30.

**Comparison of the two methods** Now, we want to compare the two methods. We have plotted the Jaccard index, precision rate and recall rate for maximum distance 100 in Figure 10. Since we only have results up to density 1 for the grid algorithm, we have omitted the results of the density 2 for the individual fitting algorithm. We immediately see that for the noise level 60 the results are not consistent and vary a lot. Furthermore, we observe that the Jaccard index for individual fitting performs slightly better than grid fitting in the lower density settings. Especially in the recall rate, there is a small difference. There is one particular case where the grid fitting algorithm will always fail to find an emitter which is when two emitters are extremely close to each other. The individual fitting algorithm then still has the choice to choose for two emitters when the intensity is too large, but the grid fitting algorithm does not have that choice. However, the small number of false negatives makes it difficult to perform a meaningful analysis on the localizations. It is also possible that we can attribute the difference between parameter choices, since these might be chosen non-optimal.

For the higher density images we see, for the noise levels 0 and 30, that the grid fitting algorithm performs significantly better than the individual fitting algorithm. The precision rate remains quite steady in both algorithms, where the grid fitting only performs slightly better, but the recall rate for the grid fitting algorithm is much better compared to individual fitting algorithm. Although the grid fitting algorithm still cannot distinguish emitters that are too close to each other, there will also be more areas where many emitters are overlapping, but not on top of each other. These situations are particularly difficult for the individual fitting algorithm, since it can at most fit two emitters at once. This also explains the declining recall rate of the individual fitting algorithm. It can only fit two emitters within a certain ROI, possibly a few extra from a ROI next to it. If the number of emitters in the ROI becomes too large, it is already guaranteed to miss some. Finally, the general better performance on higher density images of the grid fitting algorithm can also be attributed to the fact that it does not suffer from mismatches on the border of a ROI. When molecules are nicely separated in low density images, there will be few emitters intersecting the ROI of a detection, but this occurs much more often in high density images. In some of the cases these emitters are found in a different detection, but it can always miss some. Because the grid fitting algorithm processes an entire image at once, it is not affected by these issues.

**Comparison to ThunderSTORM** We also ran the ThunderSTORM algorithm itself on the data sets. The ThunderSTORM has many parameter choices as well which we will list first. The ThunderSTORM plugin contains many options for preprocessing the data in order to enhance its features helping in detecting molecules. We used the wavelet filtering using B-splines where we chose B-spline order 3 and a B-spline scale of 2.0. For the detection step we simply chose the local maximum option with a threshold value equal to the standard deviation of the first level of the wavelet transform as was suggested by them. The pixel must be above the threshold and a local maximum in the $3 \times 3$-neighborhood around it. For the localization part we chose a fitting radius (region-of-interest) of three pixels. We used the least squares fit method to better compare the results to our individual fitting method and chose the initial sigma to be 1.6 pixels.

We enabled multi-emitter fitting and limited the number molecules per region to two, since that is our maximum as well. We chose a model selection threshold of $10^{-6}$ and intensity range limit on $[700, 900]$, equal to how we generated the data. This is more information than we assumed in our algorithm, but since we also incorporated some knowledge on the range of intensity, we chose this approach for a better comparison.

**Remark 8.3.2.** The ThunderSTORM algorithm seemed to have much difficulty with detecting emitters close to the edge of the frame, resulting in a low recall rate. In order to make a fairer comparison, we decided to exclude any ground truths that were within three pixels of the border. This means that we removed both unmatched GTs (false negatives) and matched GTs (true positives). The filtered data is shown in Table 6 in Appendix B. Note that the data is missing for two images sets, background level 0 and 30 for density 0.1. This is due to a bug in the ThunderSTORM plugin where it would get stuck on these particular data sets.

Figure 11 shows the results of the ThunderSTORM algorithm next to our individual fitting algorithm. We observe that the ThunderSTORM algorithm is more stable under noise, since both the precision rate and the recall rate (and the Jaccard rate as a consequence) decrease at higher noise levels, but they remain consistent. In comparison, our individual fitting algorithm becomes very unstable at noise level 60. Moreover, the ThunderSTORM algorithm also performs more consistently in increasing densities, losing precision and recall rate gradually. The individual algorithm breaks down at density 2, performing significantly worse. Although there are several differences between the our algorithm and the ThunderSTORM algorithm, one of these differences, which could explain the described behavior, is the smaller neighborhood for finding local

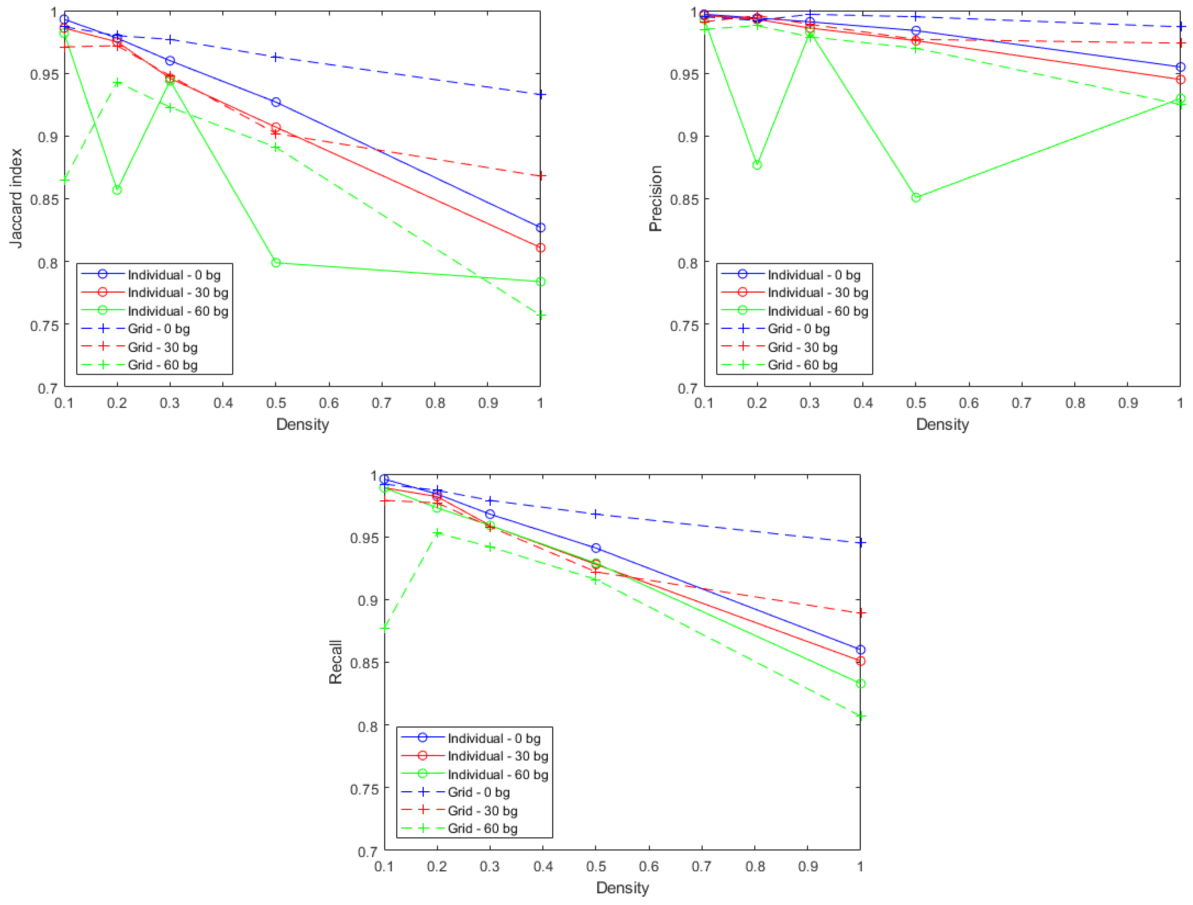


Figure 10: The Jaccard index, precision rate and recall rate of the two methods.

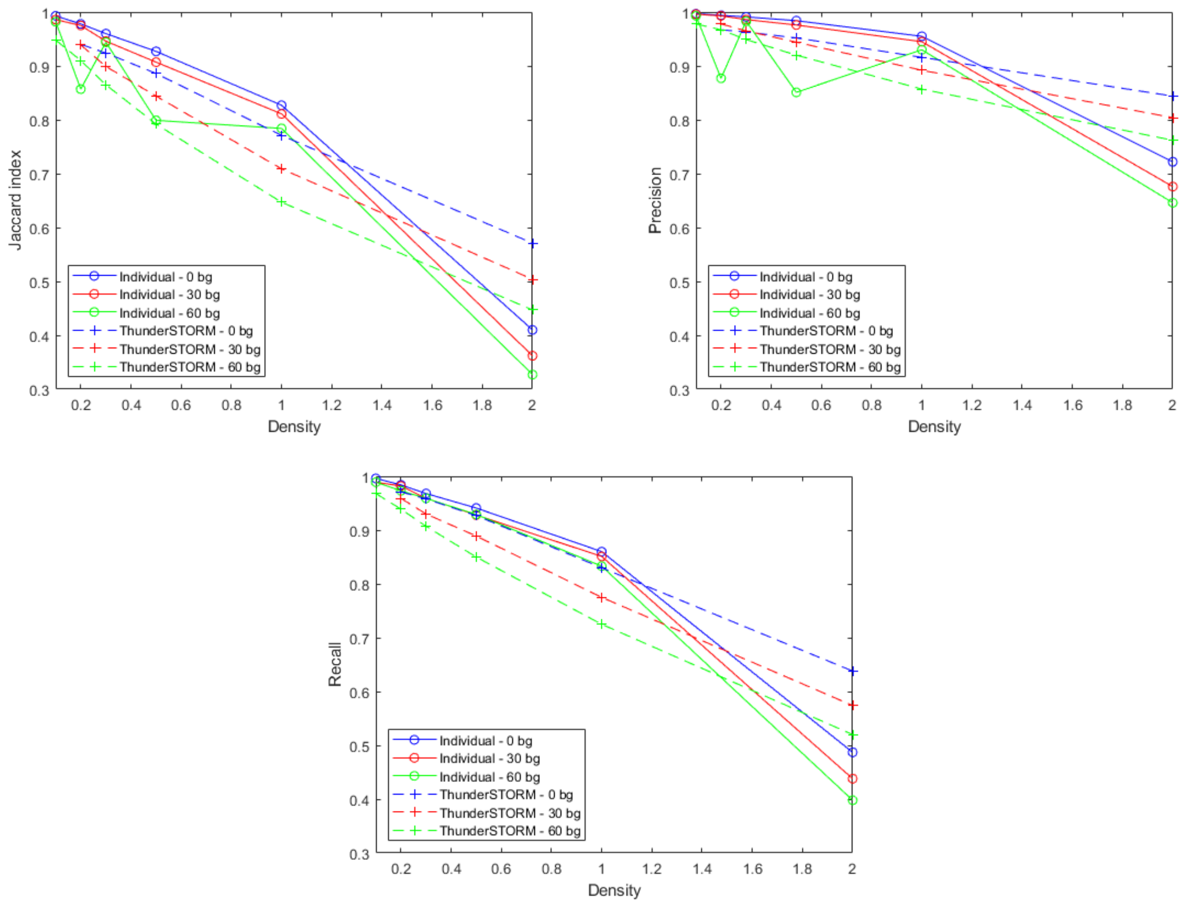


Figure 11: The Jaccard index, precision rate and recall rate of the individual fitting algorithm and ThunderSTORM [32].

maxima. ThunderSTORM has the option to either search in the 4-connected neighborhood (the 4 pixels horizontally and vertically adjacent) or the 8-connected neighborhood (including diagonal pixels). This is a much smaller neighborhood than the one in our algorithm where we chose roughly the FWHM size. This may lead to missing more emitters that are close to each other, because only the brightest attains a local maximum.

That being said, our individual fitting algorithm does outperform ThunderSTORM when the density is at most 1 and the noise level at most 30. Utilizing the intensity ranges appears to not be present in ThunderSTORM, since it caps the intensity estimated intensity values.

# 9 Discussion

The results show that for the purposes of multi-emitter fitting the grid fitting algorithm is generally a better choice. The caveat is that this algorithm is significantly slower due to large computational complexity.

## 9.1 Future research and improvements

Now, we discuss some of the future research that is still possible and improvements that can be done on the algorithms of this thesis. Our goal was to also compare the localization results with the theoretical Cramér-Rao bound. Unfortunately, we did not have time for this in the end. Computing these bounds can give us more insight in the performance of the algorithms.

For both algorithms we can greatly improve the parameter selection process. Many parameters are determined by a trial-and-error process, but a better theoretical foundation could provide more optimal parameter choices.

A useful quantity to estimate as a preprocessing step would be either the density or the number of emitters in a frame, so a good estimation for those can improve the algorithm. This allows us to better estimate the background. At the moment we consider the mode intensity $M$ and consider an average of the intensities around $M$. However, this overestimates the background parameter $\beta$, especially for high molecule density images, since some of the pixels with intensity $m$ are the sum of a smaller background intensity and a small signal intensity. Knowledge of the molecule density can compensate for this overestimation.

Both methods can also be improved in terms of runtime. In the individual fitting algorithm there is a large difference between the time usage of the 1-fit and the 2-fit. If we can detect situations where we are almost certain that there is only one emitter, we might not need to compute the 2-fit for every detection. Since this algorithm performs better for lower densities where there are relatively few overlapping emitters, such an addition can be very useful. The grid fitting algorithm is well suited for GPU parallelization, so a logical next step is to explore how large the speedup is in parallel setting.

In addition to the improvements mentioned above we can extend the individual fitting algorithm to more than two emitters. The decision process to choose between the different fits will then have to be generalized. Right now, the methodology is quite ad-hoc and similar decision steps are less suitable when we increase the number of emitters being fitted.

For the grid fitting algorithm there are also several more improvements. As mentioned before, a huge disadvantage is that we have to specify $L$ at the start. A better resolution will result in a large complexity increase, while most of the resulting image will not change at all. We could alter the algorithm in the following way. We start with a small value of $L$, say 1, for which we perform the optimization step. This leads to a small subset $P \subset [N^2L^2]$ of (sub)pixels containing emitters. Then, we increase $L$, but only refine the subpixels close to $P$. We can optimize this new set of smaller subpixels again to get a better resolution. Using this approach repeatedly, we can get an arbitrarily small precision (until floating point precision fails to distinguish neighboring subpixels) without the large increase in complexity.

Another disadvantage is that we cannot incorporate additional information into the model. If we have knowledge on the range of intensities of the emitters, then finding emitters with a larger intensity suggests multiple emitters close to each other. Our algorithm will, depending on the parameters and how close the emitters are, often simply conclude that there is one brighter emitter there. One way to address this issue is to add a post-processing step to detect these cases and enforce an extra emitter in a local fit procedure, possibly using the individual fitting algorithm.

Finally, an interesting future research direction is to find an algorithm to optimize the parameter choices for $\lambda$ and $p$, either theoretically or experimentally.

# A Parameter choices

## A.1 Preprocessing steps

We start with the preprocess parameters. We choose the background parameter $p_B = M'_B/2$ where $M'_B$ was the mode intensity. For computing the FWHM we perform the detection step with uniform filter of size $\alpha = 1$, maximum filter of size $\rho = 3$. Note that for these filters the sets $\mathcal{N}_\alpha(p)$ and $\mathcal{N}_\rho(p)$ are $3 \times 3$ and $7 \times 7$ respectively. The FWHM parameter $p_F$ is chosen to be 2 pixels. This seems quite small, but the FWHM is also just a few pixels wide.

## A.2 Individual fitting algorithm

In the actual detection step we choose the same size uniform filter, but we choose $r = \widehat{FWHM}/2 + 1$ as the neighborhood size for the region of interest. This means that the width of the ROI is always at least $\widehat{FWHM}$.

## A.3 Grid fitting algorithm

The grid fitting algorithm has also several parameters to be specified. Most importantly, we have to choose $L$ which determines how precise the coordinates of our algorithm become. We choose $L = 5$. The next parameter is the maximum distance $d$ between pixel $k$ and subpixel $i$ such that we count the contribution of an emitter in $i$ to pixel $k$. We choose $d = 4.5$ pixels, meaning that for a subpixel $i$ we consider a $9 \times 9$-window of pixels around $i$. It follows that $|C_i| = 81$.

Finally, we have the two parameters $p$ and $\lambda$ that influence the sparsity of the solution. Since these parameters depend should depend on the noise level and molecule density, we choose them for each data set separately. Choosing these parameters is ultimately a trial-and-error process, but we keep $p$ fixed for data sets with the same molecule density and tune $\lambda$ for each data set separately, following the argument in Section **??**. The chosen parameters are shown in Table 1.

| | Background | | |
|---|---|---|---|
| Density | 0 | 30 | 60 |
| 0.1 | (0.7, 0.1) | (0.7, 0.01) | (0.7, 0.01) |
| 0.2 | (0.8, 0.1) | (0.8, 0.03) | (0.8, 0.03) |
| 0.3 | (0.8, 0.1) | (0.8, 0.03) | (0.8, 0.02) |
| 0.5 | (0.85, 0.1) | (0.85, 0.05) | (0.85, 0.03) |
| 1.0 | (0.85, 0.05) | (0.85, 0.03) | (0.85, 0.03) |

Table 1: Parameter choices for the grid algorithm as pair $(p, \lambda)$

## B Tables

| (density, background) | TP | FP | FN | Jaccard | precision | recall | RMSE |
|---|---|---|---|---|---|---|---|
| (0.1, 0) | 4082 | 39 | 44 | 0.980 | 0.991 | 0.989 | 12.404 |
| (0.2, 0) | 7852 | 196 | 273 | 0.944 | 0.976 | 0.966 | 13.458 |
| (0.3, 0) | 11542 | 423 | 707 | 0.911 | 0.965 | 0.942 | 14.193 |
| (0.5, 0) | 18433 | 1315 | 2202 | 0.840 | 0.933 | 0.893 | 15.727 |
| (1.0, 0) | 31189 | 5683 | 9752 | 0.669 | 0.846 | 0.762 | 18.815 |
| (2.0, 0) | 25285 | 29831 | 56382 | 0.227 | 0.459 | 0.310 | 26.571 |
| (0.1, 30) | 3975 | 83 | 111 | 0.953 | 0.980 | 0.973 | 16.938 |
| (0.2, 30) | 7882 | 319 | 408 | 0.916 | 0.961 | 0.951 | 17.932 |
| (0.3, 30) | 11388 | 729 | 1073 | 0.863 | 0.940 | 0.914 | 18.743 |
| (0.5, 30) | 17638 | 1986 | 2992 | 0.780 | 0.899 | 0.855 | 19.948 |
| (1.0, 30) | 29965 | 7012 | 11102 | 0.623 | 0.810 | 0.730 | 22.244 |
| (2.0, 30) | 20196 | 32434 | 61025 | 0.178 | 0.384 | 0.249 | 29.322 |
| (0.1, 60) | 3949 | 143 | 163 | 0.928 | 0.965 | 0.960 | 20.483 |
| (0.2, 60) | 7439 | 1652 | 760 | 0.755 | 0.818 | 0.907 | 21.537 |
| (0.3, 60) | 11166 | 945 | 1251 | 0.836 | 0.922 | 0.899 | 21.447 |
| (0.5, 60) | 16897 | 5340 | 3469 | 0.657 | 0.760 | 0.830 | 23.876 |
| (1.0, 60) | 27661 | 8661 | 12883 | 0.562 | 0.762 | 0.682 | 24.760 |
| (2.0, 60) | 17253 | 33271 | 64657 | 0.150 | 0.341 | 0.211 | 30.530 |

Table 2: Results from individual fitting with maximum fit distance of 50nm.

| (density, background) | TP | FP | FN | Jaccard | precision | recall | RMSE |
|---|---|---|---|---|---|---|---|
| (0.1, 0) | 4108 | 13 | 18 | 0.993 | 0.997 | 0.996 | 13.506 |
| (0.2, 0) | 7998 | 50 | 127 | 0.978 | 0.994 | 0.984 | 16.160 |
| (0.3, 0) | 11860 | 105 | 389 | 0.960 | 0.991 | 0.968 | 18.007 |
| (0.5, 0) | 19426 | 322 | 1209 | 0.927 | 0.984 | 0.941 | 22.233 |
| (1.0, 0) | 35217 | 1655 | 5724 | 0.827 | 0.955 | 0.860 | 30.287 |
| (2.0, 0) | 39780 | 15336 | 41887 | 0.410 | 0.722 | 0.487 | 50.277 |
| (0.1, 30) | 4043 | 15 | 43 | 0.986 | 0.996 | 0.989 | 18.993 |
| (0.2, 30) | 8141 | 60 | 149 | 0.975 | 0.993 | 0.982 | 21.412 |
| (0.3, 30) | 11945 | 172 | 516 | 0.946 | 0.986 | 0.959 | 23.846 |
| (0.5, 30) | 19147 | 477 | 1483 | 0.907 | 0.976 | 0.928 | 27.661 |
| (1.0, 30) | 34947 | 2030 | 6120 | 0.811 | 0.945 | 0.851 | 34.055 |
| (2.0, 30) | 35601 | 17029 | 45620 | 0.362 | 0.676 | 0.438 | 54.073 |
| (0.1, 60) | 4065 | 27 | 47 | 0.982 | 0.993 | 0.989 | 23.101 |
| (0.2, 60) | 7977 | 1114 | 222 | 0.857 | 0.877 | 0.973 | 27.610 |
| (0.3, 60) | 11910 | 201 | 507 | 0.944 | 0.983 | 0.959 | 27.124 |
| (0.5, 60) | 18921 | 3316 | 1445 | 0.799 | 0.851 | 0.929 | 32.029 |
| (1.0, 60) | 33774 | 2548 | 6770 | 0.784 | 0.930 | 0.833 | 37.950 |
| (2.0, 60) | 32641 | 17883 | 49269 | 0.327 | 0.646 | 0.398 | 56.253 |

Table 3: Results from individual fitting with maximum fit distance of 100nm.

| (density, background) | TP | FP | FN | Jaccard | precision | recall | RMSE |
|---|---|---|---|---|---|---|---|
| (0.1, 0) | 788 | 46 | 49 | 0.892 | 0.945 | 0.941 | 14.701 |
| (0.2, 0) | 1506 | 68 | 76 | 0.913 | 0.957 | 0.952 | 15.045 |
| (0.3, 0) | 2286 | 106 | 150 | 0.899 | 0.956 | 0.938 | 15.685 |
| (0.5, 0) | 3878 | 199 | 316 | 0.883 | 0.951 | 0.925 | 16.803 |
| (1.0, 0) | 7313 | 544 | 890 | 0.836 | 0.931 | 0.892 | 17.876 |
| (0.1, 30) | 780 | 35 | 45 | 0.907 | 0.957 | 0.945 | 18.708 |
| (0.2, 30) | 1562 | 71 | 103 | 0.900 | 0.957 | 0.938 | 18.239 |
| (0.3, 30) | 2205 | 153 | 231 | 0.852 | 0.935 | 0.905 | 19.079 |
| (0.5, 30) | 3495 | 382 | 615 | 0.778 | 0.901 | 0.850 | 19.916 |
| (1.0, 30) | 6532 | 1005 | 1730 | 0.705 | 0.867 | 0.791 | 21.339 |
| (0.1, 60) | 664 | 47 | 134 | 0.786 | 0.934 | 0.832 | 21.159 |
| (0.2, 60) | 1512 | 101 | 160 | 0.853 | 0.937 | 0.904 | 21.718 |
| (0.3, 60) | 2193 | 238 | 332 | 0.794 | 0.902 | 0.869 | 22.200 |
| (0.5, 60) | 3346 | 433 | 657 | 0.754 | 0.885 | 0.836 | 22.794 |
| (1.0, 60) | 5451 | 1631 | 2663 | 0.559 | 0.770 | 0.672 | 24.507 |

Table 4: Results from grid fitting with maximum fit distance of 50nm.

| (density, background) | TP | FP | FN | Jaccard | precision | recall | RMSE |
|---|---|---|---|---|---|---|---|
| (0.1, 0) | 830 | 4 | 7 | 0.987 | 0.995 | 0.992 | 21.518 |
| (0.2, 0) | 1562 | 12 | 20 | 0.980 | 0.992 | 0.987 | 18.978 |
| (0.3, 0) | 2386 | 6 | 50 | 0.977 | 0.997 | 0.979 | 20.453 |
| (0.5, 0) | 4058 | 19 | 136 | 0.963 | 0.995 | 0.968 | 22.011 |
| (1.0, 0) | 7752 | 105 | 451 | 0.933 | 0.987 | 0.945 | 24.087 |
| (0.1, 30) | 808 | 7 | 17 | 0.971 | 0.991 | 0.979 | 23.111 |
| (0.2, 30) | 1626 | 7 | 39 | 0.972 | 0.996 | 0.977 | 22.525 |
| (0.3, 30) | 2333 | 25 | 103 | 0.948 | 0.989 | 0.958 | 24.603 |
| (0.5, 30) | 3788 | 89 | 322 | 0.902 | 0.977 | 0.922 | 27.676 |
| (1.0, 30) | 7341 | 196 | 921 | 0.868 | 0.974 | 0.889 | 31.294 |
| (0.1, 60) | 700 | 11 | 98 | 0.865 | 0.985 | 0.877 | 25.944 |
| (0.2, 60) | 1594 | 19 | 78 | 0.943 | 0.988 | 0.953 | 26.501 |
| (0.3, 60) | 2379 | 52 | 146 | 0.923 | 0.979 | 0.942 | 29.269 |
| (0.5, 60) | 3666 | 113 | 337 | 0.891 | 0.970 | 0.916 | 30.647 |
| (1.0, 60) | 6548 | 534 | 1566 | 0.757 | 0.925 | 0.807 | 37.464 |

Table 5: Results from grid fitting with maximum fit distance of 100nm.

| (density, background) | TP | FP | FN | Jaccard | precision | recall |
|---|---|---|---|---|---|---|
| (0.2, 0) | 6488 | 221 | 191 | 0.940 | 0.967 | 0.971 |
| (0.3, 0) | 9648 | 374 | 423 | 0.924 | 0.963 | 0.958 |
| (0.5, 0) | 15710 | 784 | 1233 | 0.886 | 0.952 | 0.927 |
| (1.0, 0) | 27928 | 2568 | 5706 | 0.771 | 0.916 | 0.830 |
| (2.0, 0) | 42831 | 7892 | 24343 | 0.571 | 0.844 | 0.638 |
| (0.2, 30) | 6455 | 142 | 275 | 0.939 | 0.978 | 0.959 |
| (0.3, 30) | 9544 | 351 | 718 | 0.899 | 0.965 | 0.930 |
| (0.5, 30) | 15027 | 887 | 1883 | 0.844 | 0.944 | 0.889 |
| (1.0, 30) | 26194 | 3161 | 7585 | 0.709 | 0.892 | 0.775 |
| (2.0, 30) | 38276 | 9337 | 28406 | 0.504 | 0.804 | 0.574 |
| (0.1, 60) | 3251 | 72 | 106 | 0.948 | 0.978 | 0.968 |
| (0.2, 60) | 6357 | 216 | 416 | 0.910 | 0.967 | 0.939 |
| (0.3, 60) | 9272 | 495 | 956 | 0.865 | 0.949 | 0.907 |
| (0.5, 60) | 14207 | 1229 | 2502 | 0.792 | 0.920 | 0.850 |
| (1.0, 60) | 24282 | 4044 | 9195 | 0.647 | 0.857 | 0.725 |
| (2.0, 60) | 34904 | 10899 | 32214 | 0.447 | 0.762 | 0.520 |

Table 6: Results from the ThunderSTORM algorithm with maximum fit distance of 100nm. The ground truths closer than 300nm from the border are excluded.

## References

[1] Anish V. Abraham, Sripad Ram, Jerry Chao, E. S. Ward, and Raimund J. Ober. Quantitative study of single molecule location estimation techniques. *Opt. Express*, 17(26):23352–23373, Dec 2009. doi: 10.1364/OE.17.023352. URL `http://opg.optica.org/oe/abstract.cfm?URI=oe-17-26-23352`.

[2] Hazen Babcock, Yaron M. Sigal, and Xiaowei Zhuang. A high-density 3d localization algorithm for stochastic optical reconstruction microscopy. *Optical Nanoscopy*, 1(1):6, Jul 2012. ISSN 2192-2853. doi: 10.1186/2192-2853-1-6. URL `https://doi.org/10.1186/2192-2853-1-6`.

[3] Arne Bechensteen, Laure Blanc-Féraud, and Gilles Aubert. New &#x2113;2&#x2009;&#x2212;&#x2009;&#x2113;0 algorithm for single-molecule localization microscopy. *Biomed. Opt. Express*, 11(2):1153–1174, Feb 2020. doi: 10.1364/BOE.381666. URL `http://opg.optica.org/boe/abstract.cfm?URI=boe-11-2-1153`.

[4] Eric Betzig, George H. Patterson, Rachid Sougrat, O. Wolf Lindwasser, Scott Olenych, Juan S. Bonifacino, Michael W. Davidson, Jennifer Lippincott-Schwartz, and Harald F. Hess. Imaging intracellular fluorescent proteins at nanometer resolution. *Science*, 313(5793):1642–1645, 2006. doi: 10.1126/science.1127344. URL `https://www.science.org/doi/abs/10.1126/science.1127344`.

[5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Jerry Chao, E Sally Ward, and Raimund J Ober. Fisher information theory for parameter estimation in single molecule microscopy: tutorial. *JOSA A*, 33(7):B36–B57, 2016.

[7] Olivier Daigle, Claude Carignan, Jean-Luc Gach, Christian Guillaume, Simon Lessard, Charles-Anthony Fortin, and Sébastien Blais-Ouellette. Extreme faint flux imaging with an emccd. *Publications of the Astronomical Society of the Pacific*, 121(882):866, aug 2009. doi: 10.1086/605449. URL `https://dx.doi.org/10.1086/605449`.

[8] Eric R Fossum and Donald B Hondongwa. A review of the pinned photodiode for ccd and cmos image sensors. *IEEE Journal of the electron devices society*, 2014.

[9] Simon Gazagnes, Emmanuel Soubies, and Laure Blanc-Féraud. High density molecule localization for super-resolution microscopy using cel0 based sparse approximation. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 28–31, 2017. doi: 10.1109/ISBI.2017.7950460.

[10] Fabian Hinterer, Magdalena C Schneider, Simon Hubmer, Montserrat López-Martinez, Philipp Zelger, Alexander Jesacher, Ronny Ramlau, and Gerhard J Schütz. Robust and bias-free localization of individual fixed dipole emitters achieving the cramér rao bound for applications in cryo-single molecule localization microscopy. *PloS one*, 17(2):e0263500, 2022.

[11] Fang Huang, Samantha L Schwartz, Jason M Byars, and Keith A Lidke. Simultaneous multiple-emitter fitting for single molecule super-resolution imaging. *Biomedical optics express*, 2(5):1377–1393, 2011.

[12] David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1): 30–37, 2004.

[13] Manuel Hüpfel, Andrei Yu. Kobitski, Weichun Zhang, and G. Ulrich Nienhaus. Wavelet-based background and noise subtraction for fluorescence microscopy images. *Biomed. Opt. Express*, 12(2):969–980, Feb 2021. doi: 10.1364/BOE.413181. URL `http://opg.optica.org/boe/abstract.cfm?URI=boe-12-2-969`.

[14] I. Izeddin, J. Boulanger, V. Racine, C.G. Specht, A. Kechkar, D. Nair, A. Triller, D. Choquet, M. Dahan, and J.B. Sibarita. Wavelet analysis for single molecule localization microscopy. *Opt. Express*, 20(3):2081–2095, Jan 2012. doi: 10.1364/OE.20.002081. URL `http://opg.optica.org/oe/abstract.cfm?URI=oe-20-3-2081`.

[15] Ivan V Kotov, Alexandra. I Kotov, James Frank, Paul O'Connor, Victor Perevoztchikov, and Peter Takacs. Ccd base line subtraction algorithms. *IEEE Transactions on Nuclear Science*, 57(4):2200–2204, 2010. doi: 10.1109/TNS.2010.2049660.

[16] Pavel Křížek, Ivan Raška, and Guy M. Hagen. Minimizing detection errors in single molecule localization microscopy. *Opt. Express*, 19(4):3226–3235, Feb 2011. doi: 10.1364/OE.19.003226. URL `http://opg.optica.org/oe/abstract.cfm?URI=oe-19-4-3226`.

[17] Mickaël Lelek, Melina T Gyparaki, Gerti Beliu, Florian Schueder, Juliette Griffié, Suliana Manley, Ralf Jungmann, Markus Sauer, Melike Lakadamyali, and Christophe Zimmer. Single-molecule localization microscopy. *Nature Reviews Methods Primers*, 1(1):1–27, 2021.

[18] Ü. Lepik. Numerical solution of differential equations using haar wavelets. *Mathematics and Computers in Simulation*, 68(2):127–143, 2005. ISSN 0378-4754. doi: https://doi.org/10.1016/j.matcom.2004.10.005. URL `https://www.sciencedirect.com/science/article/pii/S0378475404002757`.

[19] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[20] Luchang Li, Bo Xin, Weibing Kuang, Zhiwei Zhou, and Zhen-Li Huang. Divide and conquer: real-time maximum likelihood fitting of multiple emitters for super-resolution localization microscopy. *Opt. Express*, 27(15):21029–21049, Jul 2019. doi: 10.1364/OE.27.021029. URL `http://opg.optica.org/oe/abstract.cfm?URI=oe-27-15-21029`.

[21] Ariel Lipson, Stephen G Lipson, and Henry Lipson. *Optical physics*. Cambridge University Press, 2010.

[22] Daniel Locci-Lopez, Rui Zhang, Arnold Oyem, and John Castagna. The multi-scale fourier transform. 09 2018. doi: 10.1190/segam2018-2994723.1.

[23] Hongqiang Ma, Jianquan Xu, and Yang Liu. Windstorm: Robust online image processing for high-throughput nanoscopy. *Science Advances*, 5(4):eaaw0683, 2019. doi: 10.1126/sciadv.aaw0683. URL `https://www.science.org/doi/abs/10.1126/sciadv.aaw0683`.

[24] Olvi L Mangasarian. Pseudo-convex functions. In *Stochastic optimization models in finance*, pages 23–32. Elsevier, 1975.

[25] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030. URL `https://doi.org/10.1137/0111030`.

[26] Junhong Min, Cédric Vonesch, Hagai Kirshner, Lina Carlini, Nicolas Olivier, Seamus Holden, Suliana Manley, Jong Chul Ye, and Michael Unser. Falcon: fast and unbiased reconstruction of high-density super-resolution microscopy data. *Scientific Reports*, 4(1):4577, Apr 2014. ISSN 2045-2322. doi: 10.1038/srep04577. URL `https://doi.org/10.1038/srep04577`.

[27] Kim I Mortensen, L Stirling Churchman, James A Spudich, and Henrik Flyvbjerg. Optimized localization analysis for single-molecule tracking and super-resolution microscopy. *Nature methods*, 7(5):377–381, 2010.

[28] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995. doi: 10.1137/S0097539792240406. URL `https://doi.org/10.1137/S0097539792240406`.

[29] Elias Nehme, Lucien E. Weiss, Tomer Michaeli, and Yoav Shechtman. Deep-storm: super-resolution single-molecule microscopy by deep learning. *Optica*, 5(4):458–464, Apr 2018. doi: 10.1364/OPTICA.5.000458. URL `http://opg.optica.org/optica/abstract.cfm?URI=optica-5-4-458`.

[30] Frank Nielsen. Cramér-rao lower bound and information geometry. In *Connected at Infinity II*, pages 18–37. Springer, 2013.

[31] Raimund J. Ober, Sripad Ram, and E. Sally Ward. Localization accuracy in single-molecule microscopy. *Biophysical Journal*, 86(2):1185–1200, 2004. ISSN 0006-3495. doi: https://doi.org/10.1016/S0006-3495(04)74193-4. URL `https://www.sciencedirect.com/science/article/pii/S0006349504741934`.

[32] Martin Ovesný, Pavel Křížek, Josef Borkovec, Zdeněk Švindrych, and Guy M. Hagen. ThunderSTORM: a comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging. *Bioinformatics*, 30(16):2389–2390, 05 2014. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu202. URL `https://doi.org/10.1093/bioinformatics/btu202`.

[33] John A Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006.

[34] Michael J. Rust, Mark Bates, and Xiaowei Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). *Nature Methods*, 3(10):793–796, Oct 2006. ISSN 1548-7105. doi: 10.1038/nmeth929. URL `https://doi.org/10.1038/nmeth929`.

[35] Daniel Sage, Hagai Kirshner, Thomas Pengo, Nico Stuurman, Junhong Min, Suliana Manley, and Michael Unser. Quantitative evaluation of software packages for single-molecule localization microscopy. *Nature Methods*, 12(8):717–724, Aug 2015. ISSN 1548-7105. doi: 10.1038/nmeth.3442. URL `https://doi.org/10.1038/nmeth.3442`.

[36] Daniel Sage, Thanh-An Pham, Hazen Babcock, Tomas Lukes, Thomas Pengo, Jerry Chao, Ramraj Velmurugan, Alex Herbert, Anurag Agrawal, Silvia Colabrese, et al. Super-resolution fight club: assessment of 2d and 3d single-molecule localization microscopy software. *Nature methods*, 16(5):387–395, 2019.

[37] Mark J Schervish. *Theory of statistics*. Springer Science & Business Media, 2012.

[38] Alexey Sharonov and Robin M. Hochstrasser. Wide-field subdiffraction imaging by accumulated binding of diffusing probes. *Proceedings of the National Academy of Sciences*, 103(50):18911–18916, 2006. doi: 10.1073/pnas.0609643104. URL `https://www.pnas.org/doi/abs/10.1073/pnas.0609643104`.

[39] Russell E. Thompson, Daniel R. Larson, and Watt W. Webb. Precise nanometer localization analysis for individual fluorescent probes. *Biophysical Journal*, 82(5):2775–2783, 2002. ISSN 0006-3495. doi: https://doi.org/10.1016/S0006-3495(02)75618-X. URL `https://www.sciencedirect.com/science/article/pii/S000634950275618X`.

[40] P. Zelger, K. Kaser, B. Rossboth, L. Velas, G. J. Schütz, and A. Jesacher. Three-dimensional localization microscopy using deep learning. *Opt. Express*, 26(25):33166–33179, Dec 2018. doi: 10.1364/OE.26.033166. URL `http://opg.optica.org/oe/abstract.cfm?URI=oe-26-25-33166`.