

Subpixel super resolution algorithm for *PALM microscopy*

M. Gostiaux Gabriel

*M. Gabriel Gostiaux, Master of Science student, Institute of Optics,
Palaiseau, 91 120, France*

gabriel.gostiaux@institutoptique.fr

<https://github.com/GabrielGst?tab=repositories>

Abstract: In this study, we introduce a computationally efficient approach for generating super-resolution frames in Photoactivated Localization Microscopy (PALM) by accelerating the processing pipeline. The method is designed to optimize both the localization accuracy and processing speed, making it particularly suitable for medium-large datasets.

Keywords: PALM Microscopy, sub-pixel superresolution, python.

OCIS codes: (000.0000) General.

References and links

1. F. Goudail, "Fundamental of Estimation and Detection," IOGS **chap. 5**, 59–69 (2023).
2. F. Goudail, D. Bloch, O. Leveque, "FED labworks and projects," IOGS **lab 3-4-5**, (2024)
3. T. Molle Heredia, M. Denain, "Microscopie PALM," IOGS **sect. 2.2, 2.3** 4–5, (2023)

1. Introduction

PALM (Photoactivated Localization Microscopy) is a high-resolution imaging technique that surpasses the diffraction limit. It relies on the isolation of emitters selectively tagged with a fluorescent molecule, and the fitting of their Point Spread Function (PSF). The fluorescent molecules emit light when stimulated by a laser. This approach enables the precise localization of isolated emitters with an accuracy beyond the Rayleigh criterion, thus providing "super-resolution" down to around 10 nm. This technique, developed by Eric Betzig and his team in 2006, was awarded the Nobel Prize in Chemistry in 2014. It is of great importance due to its ability to enhance the resolution of existing images. Because this method is implemented with noisy frames, one should rely on estimation and detection theory to retrieve the precise coordinates of the emitters.

The workflow begins with a rigorous noise analysis of acquired frames, enabling the subsequent likelihood computation steps to operate with enhanced precision. Our algorithm leverages a matched filter for likelihood calculation across frames, a critical step in improving computation time.

To accelerate the localization process, we identify initial guesses for molecule positions using a simple peak detection strategy via the `numpy.max` function. This initial estimation significantly reduces the computational complexity at the cost of sacrificing accuracy. For precise sub-pixel localization, we then employ the Maximum Likelihood Estimator (MLE) method, with optimization executed through the `scipy.fmin` function. This choice provides a computational efficiency and high-resolution localization, as it can then swiftly converges to optimal coordinates.

2. Precision analysis

We have a test image presenting 10 PSF and an array containing the true coordinates of the emitters producing such PSF. In this section we present the analysis of the noise and the influence of the imaging parameters (location of the emitters onto the pixel array and PSF FWHM).

The data used in this study is stored as .mat files which can be loaded in our python environment through the function loadmat of the dependency scipy.io.

2.1. Noise analysis

Because the acquired frames are noisy, one should begin by understanding the noise before going forward in the computation of the likelihood. So, we start by extracting a noisy region of the test image (see Fig. 1, one could have used a frame from the PALM sequence equivalently), and after flattening the array, we plot it as a 1D signal (see Fig. 2a and Fig. 2b).

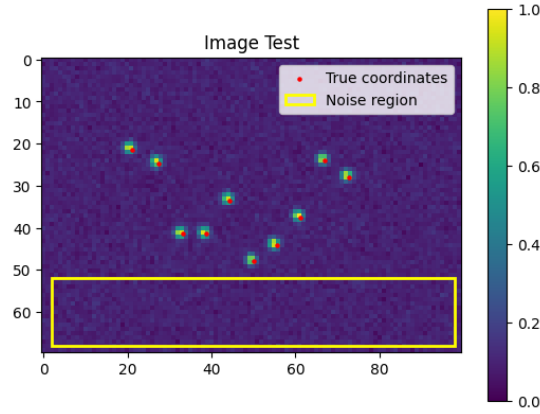


Fig. 1: Test Image

This unidimensional signal seems to present a gaussian noise, so we plot a histogram of its values because if we are right, we want to fit a gaussian curve to it to determine its amplitude, mean and standard deviation.

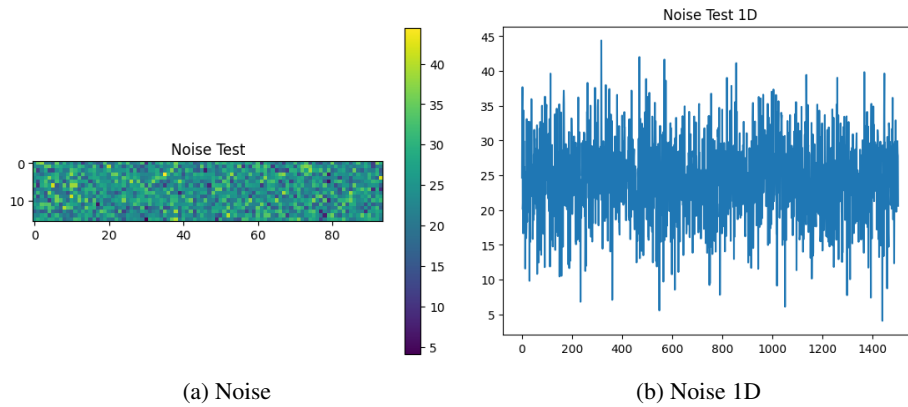


Fig. 2: Noise

The histogram confirms that the noise is gaussian, and we obtain its parameters through the fitted curve: $[a = 0.06, m = 0.06, \sigma_r = 5.78]$. We then plot the cross correlation of the noise with itself to observe its spatial coherence : since the FWHM length is one (1) pixel, we understand that the noise is white.

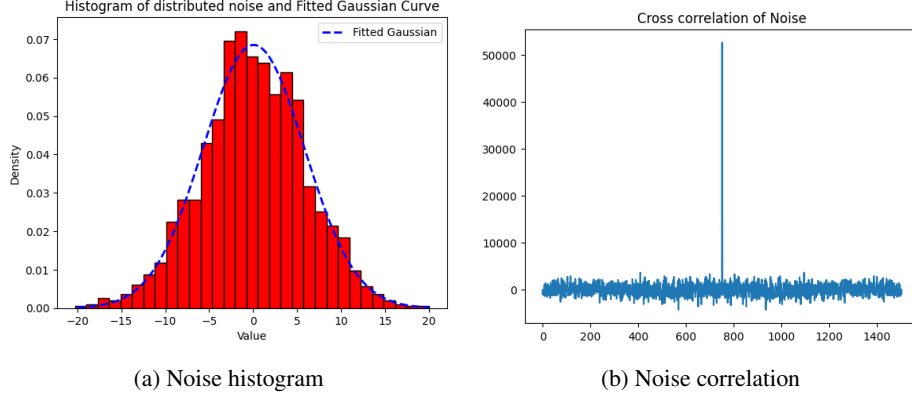


Fig. 3: Noise properties

This preliminary analysis concludes that the acquired frames are polluted by a white normal noise of parameters $[a = 0.06, m = 0.06, \sigma_r = 5.78]$ that will be of use in the computation of likelihood.

2.2. Accuracy analysis

For simplicity reasons and without any loss of generality, we consider the location from a 1D point of view. Let x be the spatial coordinate, θ the position of an emitter. It is then assumed that the signal given by this single emitter on the image plan can be written as $s(x, \theta) = ar(x, \theta) + b$ where r is a Gaussian function centered on θ so that

$$r(x, \theta) = \frac{1}{\sqrt{2\pi}\sigma_r} \exp\left[-\frac{(x - \theta)^2}{2\sigma_r^2}\right] \quad (1)$$

where $\sigma_r = w/(2\sqrt{2\ln(2)})$ is expressed as a function of the full width at half maximum (FWHM) w of the signal on the sensor. In this study, b is assumed to be an additive normal white noise of standard deviation σ_b .

We define r_i the integration of $r(x, \theta)$ over the pixel i with $i \in [0, N - 1]$

$$r_i = \int_{i\Delta x}^{(i+1)\Delta x} r(x, \theta) dx \quad (2)$$

where Δx is the discretization step. We can then compute the expression of r_i as a function of the parameters θ and w . We remind that

$$\frac{2}{\sqrt{\pi}} \int_0^z e^{-u^2} du = \text{erf}(z)$$

Let's compute r_i with a integration by substitution :

$$r_i = \int_{i\Delta x}^{(i+1)\Delta x} \frac{1}{\sqrt{2\pi}\sigma_r} \times \exp \left[- \left(\frac{x - \theta}{\sqrt{2}\sigma_r} \right)^2 \right] dx \quad (3)$$

$$= \frac{1}{\sqrt{2\pi}\sigma_r} \times \sqrt{2}\sigma_r \int_{u^-}^{u^+} e^{-u^2} du \quad (4)$$

$$= \frac{1}{\sqrt{\pi}} \left[\int_0^{u^+} e^{-u^2} du - \int_0^{u^-} e^{-u^2} du \right] \quad (5)$$

$$r_i = \frac{1}{2} \left[\operatorname{erf} \left[\frac{(i+1)\Delta x - \theta}{\sqrt{2}\sigma_r} \right] - \operatorname{erf} \left[\frac{i\Delta x - \theta}{\sqrt{2}\sigma_r} \right] \right] \quad (6)$$

Our goal is to estimate the parameter θ_0 that maximizes the gaussian signal.

2.3. Position estimation without nuisance parameter: a known

$$P(s_i) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp \left[- \frac{(s_i - ar_i)^2}{2\sigma_b^2} \right] \quad (7)$$

$$l_\theta = -N \times \ln \left(\sqrt{2\pi}\sigma_b \right) - \frac{1}{2\sigma_b^2} \sum_{s_i \in \Omega} (s_i - ar_i)^2 \quad (8)$$

$$\frac{\partial l_\theta}{\partial \theta} = \frac{1}{\sigma_b^2} \sum_{s_i \in \Omega} (s_i - ar_i) \times a \times \frac{\partial r_i}{\partial \theta} \quad (9)$$

$$\frac{\partial r_i}{\partial \theta} = - \frac{1}{\sqrt{2\pi}\sigma_r} \times \left[\exp - \left(\frac{(i+1)\Delta x - \theta}{\sqrt{2}\sigma_r} \right)^2 - \exp - \left(\frac{i\Delta x - \theta}{\sqrt{2}\sigma_r} \right)^2 \right] \quad (10)$$

$$\frac{\partial^2 l_\theta}{\partial^2 \theta} = - \frac{1}{\sigma_b^2} \sum_{s_i \in \Omega} \left(-a \left(\frac{\partial r_i}{\partial \theta} \right)^2 - (s_i - r_i) \frac{\partial^2 r_i}{\partial^2 \theta} \right) \quad (11)$$

$$\left\langle \frac{\partial^2 l_\theta}{\partial^2 \theta} \right\rangle = - \frac{1}{\sigma_b^2} \sum_{s_i \in \Omega} \left(-a \left(\frac{\partial r_i}{\partial \theta} \right)^2 - (\langle s_i \rangle - ar_i) \frac{\partial^2 l_\theta}{\partial^2 \theta} \right) \quad (12)$$

$$\left\langle \frac{\partial^2 l_\theta}{\partial^2 \theta} \right\rangle = \frac{a}{\sigma_b^2} \sum_{s_i \in \Omega} \left(\frac{\partial r_i}{\partial \theta} \right)^2 \quad (13)$$

$$CRLB = - \frac{\sigma_b^2}{1} \times \left(\sum_{s_i \in \Omega} \left(\frac{\partial r_i}{\partial \theta} \right)^2 \right)^{-1} \quad (14)$$

$$CRLB = 2\pi\sigma_b^2\sigma_r^2 \times \left(\sum_{s_i \in \Omega} \left(e^{-u_+^2} - e^{-u_-^2} \right)^2 \right)^{-1} \quad (15)$$

2.4. Position estimation with nuisance parameter: a unknown

2.5. Choices of imaging parameters: θ and w

Influence of θ over precision of estimation

Influence of w over precision of estimation

3. Implementation of the algorithm

We have a sequence of 999 frames each containing 10 PSF spreaded over the frame. We are presented with the 999 frames overlayed so that it produces a blurred image (see Fig. 4) which seems to be a text. Our goal is to extract precisely the locations of the emitters producing the PSF and to arrange them into a super-resoluted grid (say 10 times wider).

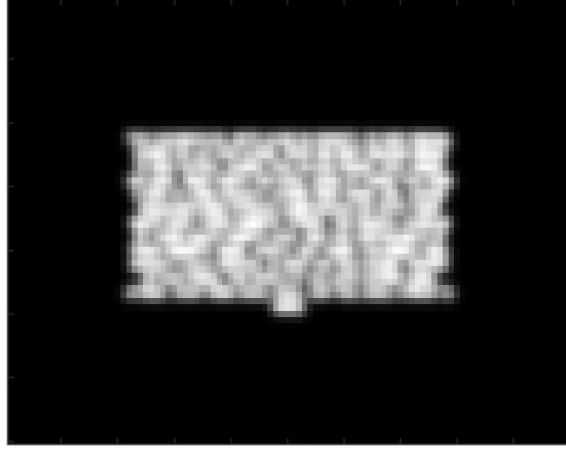


Fig. 4: Image floue

To compute the Maximum Likelihood Estimator (MLE), one should first define the model function. Here, it will be a two-dimensional gaussian function as defined in 1D in subsection 2.2. Thus, we have the following model function, after pixel-wide integration :

$$r_{ij}(\theta_x, \theta_y) = \frac{1}{4} \left[\operatorname{erf} \left(\frac{(i+1) - \theta_x}{\sqrt{2} \cdot \sigma_r} \right) - \operatorname{erf} \left(\frac{i - \theta_x}{\sqrt{2} \cdot \sigma_r} \right) \right] \times \left[\operatorname{erf} \left(\frac{(j+1) - \theta_y}{\sqrt{2} \cdot \sigma_r} \right) - \operatorname{erf} \left(\frac{j - \theta_y}{\sqrt{2} \cdot \sigma_r} \right) \right] \quad (16)$$

With that in mind, we can proceed to compute the log-likelihood :

$$l(\theta_x, \theta_y) = -I \cdot J \cdot \ln(\sqrt{2\pi}\sigma_b) - \frac{1}{2\sigma_b^2} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} (s_{ij} - ar_{ij})^2 \quad (17)$$

This function will unravel its full capacity when addressing the optimization step. However, in regard to the computation time improvement, we will proceed in computing the log-likelihood through the ambiguity function using the matched filter. The matched filter, computed as a correlation in Fourier's domain between a model function centered in its frame and the 999 acquired images, is 10 times faster than the numerical computation with the analog expression of equation 17.

We can then export the 999 log-likelihood arrays to import them later. They will be referred to as frames from now on.

On each frame, we first locate the 10 pixel-scale global maxima and store them in a list as initial guesses for the solving function. Then, we use the fmin solving function from scipy.optimize module to locate the local maxima in a sub-pixel scale. This workflow allows

to compute the log-likelihood function according to its analog expression only in a neighbourhood of the global maxima, which significantly reduces the computation time. We plot the initial guesses and the super-resolved emitters coordinates on Fig. 5.

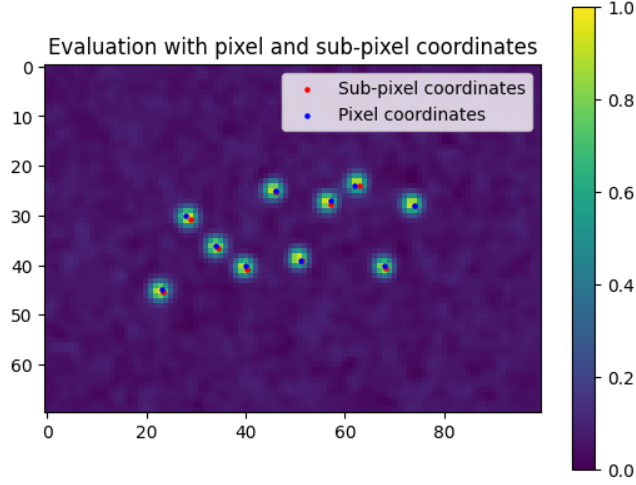


Fig. 5: Evaluation of algorithm accuracy

The coordinates seems relevant in a significant subset of the acquire image, therefore we decide now to proceed to the analysis of the 999 acquired images. On Fig. 6a, one could recognize the shape of the blurred image, suggesting that the maxima should be well computed (at pixel scale and therefore at sub-pixel scale too). Fig. 6b shows reconstruction on its way...

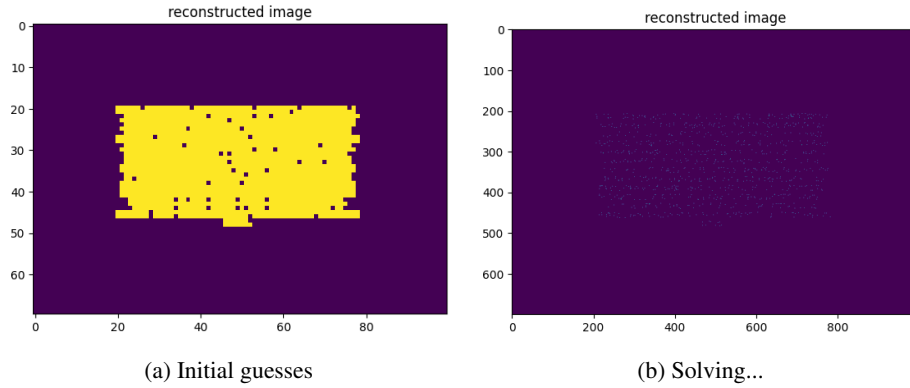


Fig. 6: Computing pix and subpix coordinates

The algorithm finally produces Fig. 7 on which one could read a passage of the famous "Romeo and Juliette" piece written by M. William Shakespeare. This proves the described fast-algorithm still produces super-resolved frames without significant loss in accuracy compared to the basic ones. The figure was produced in fifteen (15) minutes total, including five (5) minutes of log-likelihood computation and export, and ten (10) minutes of super-resolution estimation, on custom desktop hardware including intel core i7 14700KF processor and NVIDIA RTX 4060 Ti 8Gb graphic card (which was not used for computation).

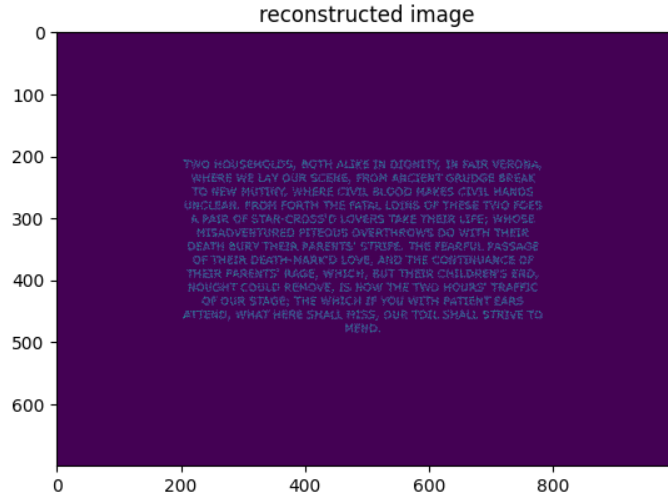


Fig. 7: Super resolution image

4. Conclusion

This study presented a specific workflow to optimize computation time for PALM microscopy. The proposed method consist in using a matched filter to compute and save the log-likelihood arrays from the acquired frames, then use the numpy max function on those to find pixel-wide initial guesses for the scipy fmin optimization function. The time efficiency is then improved by a factor of 10, leading to compute the super-resoluted image presented at the end of section 2 in 10 minutes, while it was expected to be computed in 100 minutes with the basic fmin MLE optimization method.

The workflow produces a super-resoluted image on which one could read a passage of the famous "Romeo and Juliette" piece written by M. William Shakespeare, proving the accuracy of the method.