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 Estimationa 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 likelyhood 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 likelyhood. 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).

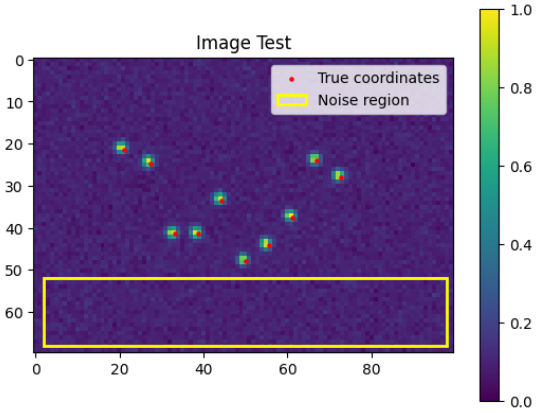
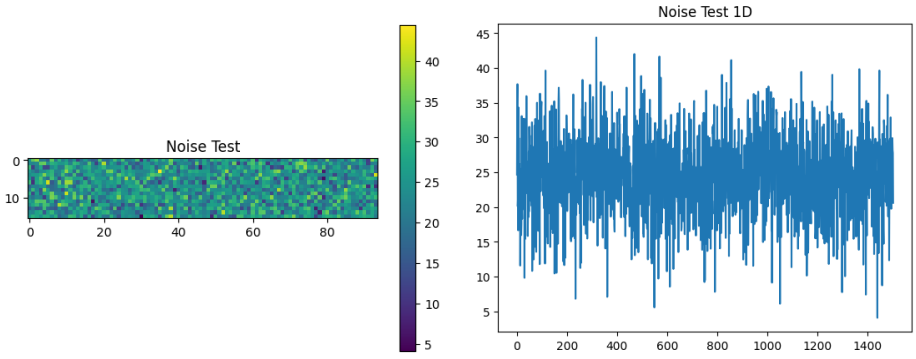


Fig. 1: Test Image

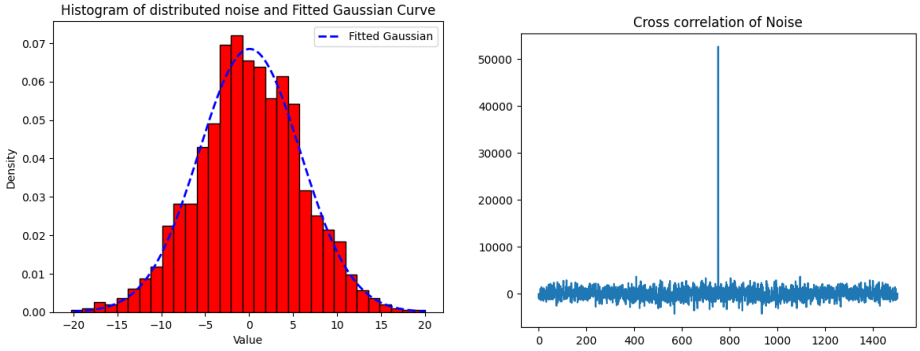
This unidimensional signal seems to present a gaussian noise, so we plot a histogram of its values because if we are right, we want to fit a gaussian curve to it to determine its amplitude, mean and standard deviation.



(a) Noise (b) Noise 1D

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.



(a) Noise histogram

(b) Noise correlation

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 likelyhood.

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

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

where $\sigma_r = w/(2\sqrt{2ln(2)})$ 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 $i \in [0, N-1]$

$$r_i = \int_{i\Delta x}^{(i+1)\Delta x} r(x,\theta)\mathrm{d}x \tag{2}$$

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

$$\frac{2}{\sqrt{\pi}} \int_0^z e^{-u^2}\, \mathrm{d}u = \mathrm{erf}(z)$$

Let's compute r_i with a integration by substitution :

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

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

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

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

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

2.3. Position estimation without nuisance parameter: a known

Here, a is known and equal to 1. The samples are statisticcally independent, they follow a normal probability law described in equation 7 which leads to the expression 8 for the log-likelyhood. We then compute the derivative along θ to equal expression 9 to zero in order to find the estimator of θ, $\hat{\theta}_{ML}$.

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

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

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

The derivativve let appear the derivative of r_i along θ which is :

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

Because expression 9 does not have any analytical solution, we solve it numerically using the fmin function from scipy.optimize dependency. We then choose a parameter $\theta_0 = 41.0050$ to estimate and we evaluate the bias and the variance of its associated estimator $\hat{\theta}_{ML}$ by using a Monte Carlo simulation over 10 000 iterations.

Here we present the results of [3] because ours (produced in lab 3, [2]) cannot be computed at the moment of the redaction of this report. The presented bias is $-3.210e-04$ and the presented variance is $2.352e-04$. In regard of the computed bias, we assume the estimator non-biased.

We then compute the Cramer Rao Lower Bound (CRLB) to see if the variance of the estimator reaches the CRLB. In this case, the estimator $\hat{\theta}_{ML}$ can be defined as efficient. In that purpose, we computes the second derivative of l_θ along θ (expression 11) and computes its mean (expressions 12 and 13).

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

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

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

To obtain the CRLB, we then take the negative inverse of the computed mean to finally get its expression :

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

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

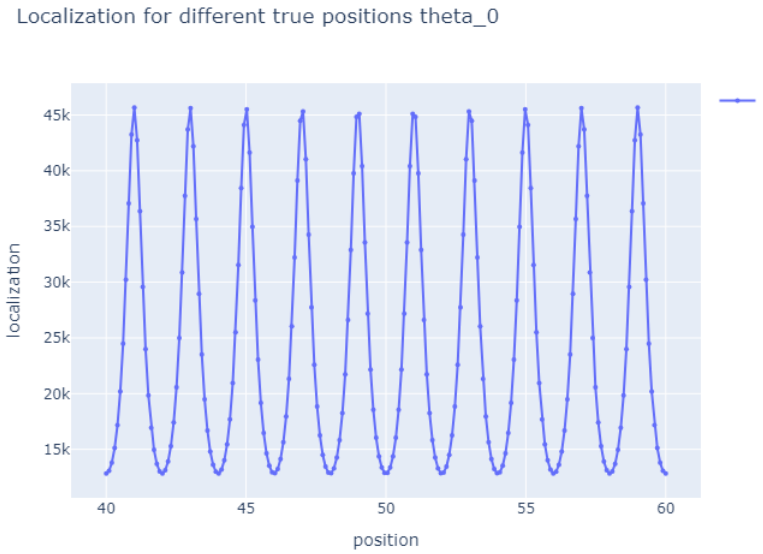


Fig. 4: CRLB as a function of θ

The result presented in the study is $2.568e-04$ so that we conclude the estimator is efficient [3]. Because the estimator is non-biased and because it belongs to the exponential family, one could have predicted thoses results [1].

2.4. *Position estimation with nuisance parameter: a unknown*

In the case where a is unknown, one should first estimate its value in order to estimate the location (θ). We set a as a parameter in the expression of the log-likelyhood.

$$l(a,\theta) = -N\ln\left(\sqrt{2\pi}\sigma_b\right) - \frac{1}{2\sigma_b^2}\sum_{i=0}^{N-1}(s_i - ar_i)^2 \tag{16}$$

We then computes the derivative along a to find the MLE $\hat{a}_{ML}$:

$$\left.\frac{\partial l}{\partial a}\right|_{a-\hat{a}_{ML}} = \frac{1}{\sigma_b^2}\sum_{i=0}^{N-1} r_i(\theta)\left(s_i - \hat{a}_{ML}r_i(\theta)\right) = 0 \tag{17}$$

We get :

$$\hat{a}_{ML} = \frac{\sum_{i=0}^{N-1} s_i r_i(\theta)}{\sum_{i=0}^{N-1} r_i(\theta)^2} \tag{18}$$

Now, we inject this expression in equation 8. As before, we solve it numerically. We then choose a parameter to estimate and we evaluate the bias and the variance of its associated estimator $\hat{\theta}_{ML}$ by using a Monte Carlo simulation over 10 000 iterations. Again (and for the same reasons as mentioned above), we present here the results of [3]. The measured bias $6.998e-05$ is and the measured variance is $2.424e-04$. Again, the estimator is non biased, and because it belongs to the exponential family, we expect it to reach the CRLB.

In this mean, we start by computing the Fisher Information matrix :

$$I(a,\theta) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{19}$$

With the following coefficients :

$$A = \frac{1}{\sigma_b^2}\sum_{i=0}^{N-1} r_i(\theta)^2 \tag{20}$$

$$B = \frac{1}{\sigma_b^2}\sum_{i=0}^{N-1} \hat{a}_{ML} r_i(\theta)\frac{dr_i}{d\theta} \tag{21}$$

$$C = B \tag{22}$$

$$D = \frac{1}{\sigma_b^2}\sum_{i=0}^{N-1} \hat{a}_{ML}^2\left[\frac{dr_i}{d\theta}\right]^2 \tag{23}$$

Because $I(a,\theta)$ is invertible [1], we compute $J(a,\theta) = I(a,\theta)^{-1}$. The CRLB is given by the coefficient (2,2) of matrix J, so that :

$$CRLB(\theta) = \frac{A}{AD - BC} \tag{24}$$

The results presented in [3] suggest that no matter the knownledge of a, the estimator $\hat{\theta}_{ML}$ is efficient.

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. 5) 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).

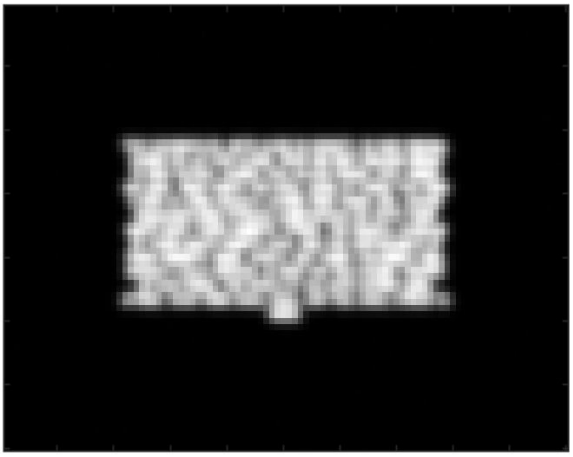


Fig. 5: Image floue

To compute the Maximum Likelyhood 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 :

$$\begin{aligned} r_{ij}(\theta_x, \theta_y) = \frac{1}{4}\left[\operatorname{erf}\left(\frac{(i+1)-\theta_x}{\sqrt{2}\cdot\sigma_r}\right) - \operatorname{erf}\left(\frac{i-\theta_x}{\sqrt{2}\cdot\sigma_r}\right)\right] \\ \times\left[\operatorname{erf}\left(\frac{(j+1)-\theta_y}{\sqrt{2}\cdot\sigma_r}\right) - \operatorname{erf}\left(\frac{j-\theta_y}{\sqrt{2}\cdot\sigma_r}\right)\right] \end{aligned} \tag{25}$$

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

$$l(\theta_x, \theta_y) = -I\cdot J\cdot \ln\left(\sqrt{2\pi}\sigma_b\right) - \frac{1}{2\sigma_b^2}\sum_{i=0}^{I-1}\sum_{j=0}^{J-1}(s_{ij} - ar_{ij})^2 \tag{26}$$

This function will unravel its full capacity when adressing the optimization step. However, in regard to the computation time improvement, we will proceed in computing the log-likelyhood 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 26.

We can then export the 999 log-likelyhood arrays to import them later. They will be refered 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-likelyhood function according to its analog expression only in a neigh-

bourhood of the global maxima, which significantly reduces the computation time. We plot the initial guesses and the super-resoluted emitters coordinates on Fig. 6.

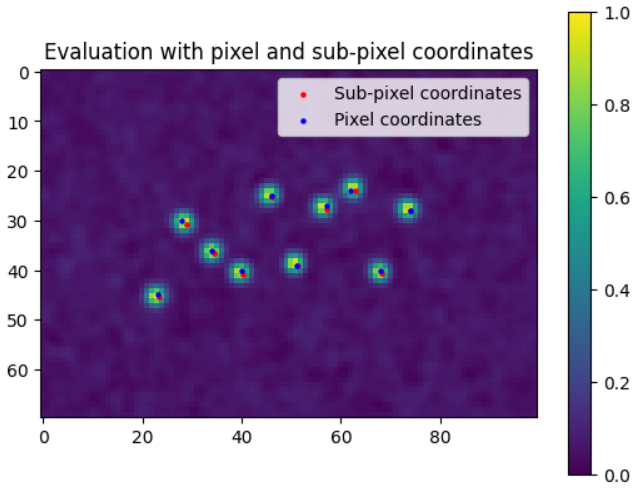
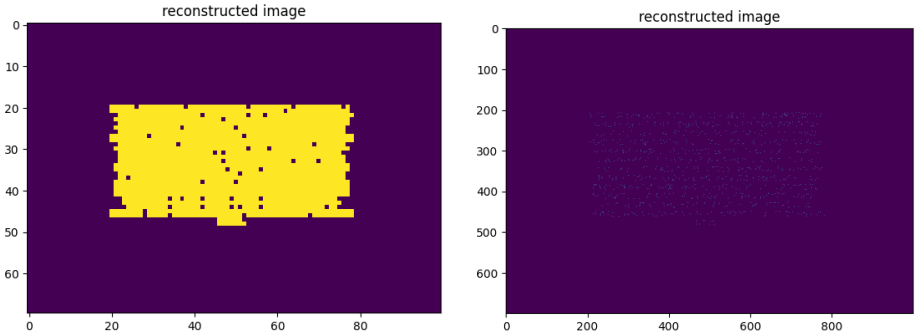


Fig. 6: Evaluation of algorithm acccuracy

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. 7a, 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. 7b shows reconstruction on its way...



(a) Initial guesses

(b) Solving...

Fig. 7: Computing pix and subpix coordinates

The algorithm finally produces Fig. 8 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-resoluted 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-likelyhood 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).

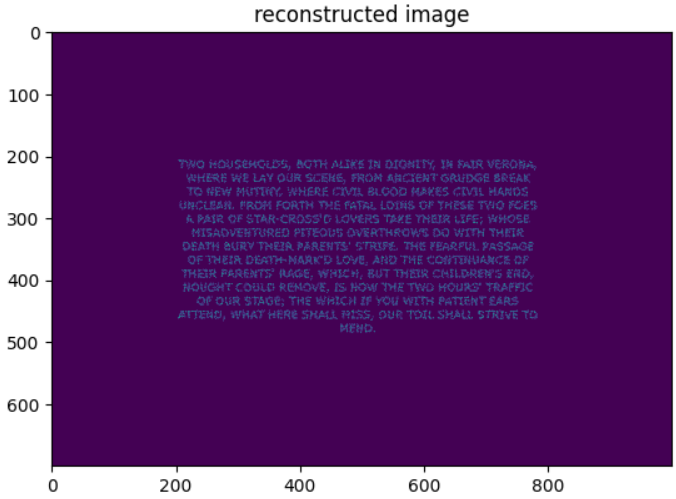


Fig. 8: Super resolution image

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-likelyhood 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.

List of Figures