$\mathbf{MLE_{bfgs}}$ localization algorithm

For single molecule localization, Gaussian function is usually use to model the PSF of an imaging system. Thus, for 2D imaging, an ideal forward imaging model is:

$$I_{i,j} = Ae^{-\frac{(i-x_0)^2+(j-y_0)^2}{2\sigma^2}} + B \tag{1}$$

Here, $I_{i,j}$ is the theoretical signal intensity at pixel (i, j), A and B are peak signal and background, respectively, (x_0, y_0) is the center position of fluorescent molecule, and σ is standard deviation of 2D Gaussian PSF. In practice, since the image is contaminated by noises (mainly by photon shot noise), the MLE optimization problem to fit molecule center position can be derived by maximizing the probability of observed signal as describe before:

$$\begin{gathered} \min L = \sum_{1\le i,j\le N} I_{i,j} - \sum_{1\le i,j\le N} q_{i,j} \ln I_{i,j} \\ s.t\ \ A > 0, x_0 > 0, y_0 > 0, \sigma > 0, B > 0 \end{gathered} \tag{2}$$

Here, q_{ij} is the observed signal intensity at pixel (i, j), and N is the size of the region of interest (ROI) for the detected molecule. Previously, this problem is widely solved by Levenberg–Marquardt optimization algorithm, which cannot achieve real-time single molecule localization for localization microscopy with large FOV (especially when sCMOS camera is used as the detector), even after combining with high performance GPU parallel computing. To accelerate single molecule localization, we adopt several optimizations to improve the efficiency of the algorithm. Firstly, we reformulate the forwarding imaging model as:

$$I_{i,j} = 128 \times Ae^{-0.1\times\sigma\left((i-x_0)^2+(j-y_0)^2\right)} + 64 \times B \tag{3}$$

Here, the coefficients (128, 64 and 0.1) are empirical scaling factor to make the dynamic range of all fitting parameters similar, which can significantly improve algorithm robustness. On the other hand, the division operation in Eq. (1) is replaced by multiplication to accelerate computation in GPU.

Then the MLE optimization problem for single molecule fitting is set to be:

$$\begin{gathered} \min L = \sum_{1\le i,j\le N}\left(I_{i,j} - q_{i,j}\right) - \sum_{1\le i,j\le N}\left(q_{i,j} \ln I_{i,j} - q_{i,j} \ln q_{i,j}\right) \\ s.t\ \ A > 0, x_0 > 0, y_0 > 0, \sigma > 0, B > 0 \end{gathered} \tag{4}$$

The main modification in Eq. (4) is that the summed observed signal level is subtracted from the cost function in Eq. (2), so that single-precision floating-point number can be used during optimization. Before this modification, the cost function L is very large and has to be represented by double-precision floating-point number, which is less efficient in GPU computing than single-precision floating-point number.

Finally, the BFGS quasi-Newton optimization algorithm is introduced to solve Eq. (4). The steps of implementing $\mathrm{MLE_{bfgs}}$ algorithm are described below.

Algorithm 1. Implementation of the MLE_{bfgs} localization algorithm.

Input: Raw image ROI and camera parameters including QE, gain and offset.
Initialization: First, the pixels of the ROI are pre-processed to convert the grey value to electron. Set the iteration counter $k = 1$. Set the column vector of the fitting parameters $\boldsymbol{x_k} = [A, x_0, y_0, \sigma, B]^T$, where the background B is estimated by the mean value of pixels at ROI edge. The amplitude A is calculated by the center pixel value minus the background. The molecule position (x_0, y_0) is directly given by the center coordinate of the ROI, and the Gaussian PSF width σ is given by the ROI size divided by 4.7 (2×2.35).

Then, initialize the approximated inverse hessian matrix $\boldsymbol{D_k}$ by identity matrix. Initialize gradient vector $\boldsymbol{g_k} = \Delta L/\Delta \boldsymbol{x_k}$ by symmetric numerical derivative, where L is the optimization target in Eq. (4). Initialize the search direction $\boldsymbol{d_k} = -\boldsymbol{g_k}$. Set the total the iteration number to be eight for 2D localization.

for $k = 1$ to total iteration number

Normalize d_k by its mean absolute value.

Find the optimal walk length λ_k by bisection method from 0 to 1 with 12 iterations to minimize $L(\boldsymbol{x_k} + \lambda_k \boldsymbol{d_k})$.

Update $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \lambda_k \boldsymbol{d_k}$, and assign $\boldsymbol{s_k} = \lambda_k \boldsymbol{d_k}$.

Update $\boldsymbol{g_{k+1}} = \Delta L/\Delta \boldsymbol{x_{k+1}}$, and assign $\boldsymbol{y_k} = \boldsymbol{g_{k+1}} - \boldsymbol{g_k}$.

Update $\boldsymbol{D_{k+1}} = \left(\boldsymbol{I} - \frac{\boldsymbol{s_k y_k^T}}{\boldsymbol{y_k^T s_k}}\right) \boldsymbol{D_k} \left(\boldsymbol{I} - \frac{\boldsymbol{y_k s_k^T}}{\boldsymbol{y_k^T s_k}}\right) + \frac{\boldsymbol{s_k s_k^T}}{\boldsymbol{y_k^T s_k}}$, where I is identity matrix.

Update $d_{k+1} = -D_{k+1} \times g_{k+1}$.

end for

Output: The fitted parameters $\boldsymbol{x_{k+1}}$. Convert the amplitude A and background B in this vector from electron to photon.

The MLE_{bfgs} localization algorithm can be extended to 3D imaging straightforwardly by adding more fitting parameters and slightly increasing the total iteration number. The algorithm is accelerated by GPU via NVidia CUDA by fitting molecular ROIs in parallel.