Bayesian Multiple Emitter Fitting using Reversible Jump Markov Chain Monte Carlo

Mohamadreza Fazel[1], Michael J. Wester[2], Hanieh Mazloom-Farsibaf[1], Marjolein B. M. Meddens[1], Alexandra Eklund[3,4], Thomas Schlichthaerle[3,4], Florian Schueder[3,4], Ralf Jungmann[3,4] and Keith A. Lidke[1]

In single molecule localization-based super-resolution imaging, high labeling density or the desire for greater data collection speed can lead to clusters of overlapping emitter images in the raw super-resolution image data. We describe a Bayesian inference approach to multiple-emitter fitting that uses Reversible Jump Markov Chain Monte Carlo to identify and localize the emitters in dense regions of data. This formalism can take advantage of any prior information, such as emitter intensity and density. The output is both a posterior probability distribution of emitter locations that includes uncertainty in the number of emitters and the background structure, and a set of coordinates and uncertainties from the most probable model.

In single molecule localization microscopy (SMLM) super-resolution approaches[1–4], a sparse subset of single fluorescent emitters that label the target structure is activated and the position of each isolated emitter is found with a precision much better than the diffraction limit. Accumulation of enough label positions allows the reconstruction of images with high spatial resolution[5]. Dense images with overlapping emitters can be either unavoidable due to densely labeled structures, or desired to shorten data collection time. However, improper analysis of this data can lead to artifacts, such as a contrast inversion in the super-resolution image (dense areas appear sparse) [6]. One way to ameliorate this issue is to use multiple-emitter fitting approaches[6–12], which allow modeling and/or fitting of multiple overlapping emitters. Several multiple-emitter fitting methods have been reported including approaches based on maximum likelihood[6], deconvolution with L1 norm constraints[9,10], PSF radial symmetry and intermittency[11], and using a Bayesian approach to integrate over all possible positions and blinking events of emitters[7].

[1] Department of Physics and Astronomy, University of New Mexico, Albuquerque, New Mexico, USA.
[2] Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico, USA.
[3] Department of Physics and Center for Nanoscience, Ludwig Maximilian University, Munich, Germany.
[4] Max Planck Institute of Biochemistry, Martinsried, Germany.

In this work, we describe a BAyesian Multiple-emitter Fitting (BAMF) analysis that uses Reversible Jump Markov Chain Monte Carlo (RJMCMC)[13,14]. The Bayesian formalism allows the inclusion of strong prior information such as the photophysics of the probe and the emitter density. RJMCMC allows classification uncertainty, i.e., uncertainty in the true number of emitters to be incorporated in the emitter location probability distribution. BAMF also couples background estimation and its uncertainty with inference of emitter locations and intensities. The result is a posterior probability distribution for emitter positions that considers both prior knowledge and sources of uncertainty that are often ignored.

Markov Chain Monte Carlo (MCMC) is a computationally efficient method for sampling from a multi-dimensional posterior probability distribution[15,16]. RJMCMC takes the concept of MCMC further and allows jumps between parameter spaces with different numbers or types of parameters. The acceptance probability for inter-space jumps is given by an extension of the Metropolis-Hasting formula (**Supplementary Note 1**)[13,14] , resulting in a chain that spends time in each space proportional to the posterior probability of that space. The histogram of the returned chain can be interpreted as a probability distribution for the parameters of interest, which in the multiple-emitter fitting problem are the emitters' positions, whereas the other parameters and states can be marginalized out.

The entire BAMF algorithm consists of several steps (Fig. 1 a): (1) converting raw data to photon counts, (2) estimation of the intensity prior, (3) division of each image into sub-regions, (4) the core RJMCMC algorithm, (5) using the RJMCMC chain to initialize MCMC within the most probable space, (6) using the MCMC chain to calculate the parameters and their associated uncertainties, and (7) making the final reconstructions by removing the localizations in the overlapping areas of the sub regions and combining the results.

The RJMCMC step is used within a fitting sub-region and calculates a posterior distribution to make inferences about a set of parameters. This requires both a likelihood model and prior distributions. The likelihood is calculated assuming a model consisting of a set of emitter positions, a PSF model (2D Gaussian[6,17] or provided by the user), a tilted plane as unstructured background and Poisson statistics. The emitters model both apparent emitters (signal) or structured background. Each parameter has a corresponding prior distribution that is given in **Supplementary Note 1 & Supplementary Table 1**.

We allow three within-space moves (no change in number or type of emitters): 1) A single-emitter move that changes the position and intensity of one or more emitter; 2) A group move that makes correlated changes in two or more emitters, and 3) A background move, which changes the parameters of a tilted plane background model (**Supplementary Note 1**). We permit four pairs of reversible jump types between parameter spaces: (birth, death), (split, merge), (generalized split, generalized merge) and (signal, background) (Fig. 1b, c, d, f and **Supplementary Fig. 1, 5**). Birth (death) allows the addition (deletion) of an emitter anywhere in the model. Split and merge allows a split and merge between two emitters. Generalized split and merge splits or combines *N* emitters. Signal (background) converts an emitter from a PSF shaped kernel of a background structure to a detected emitter.

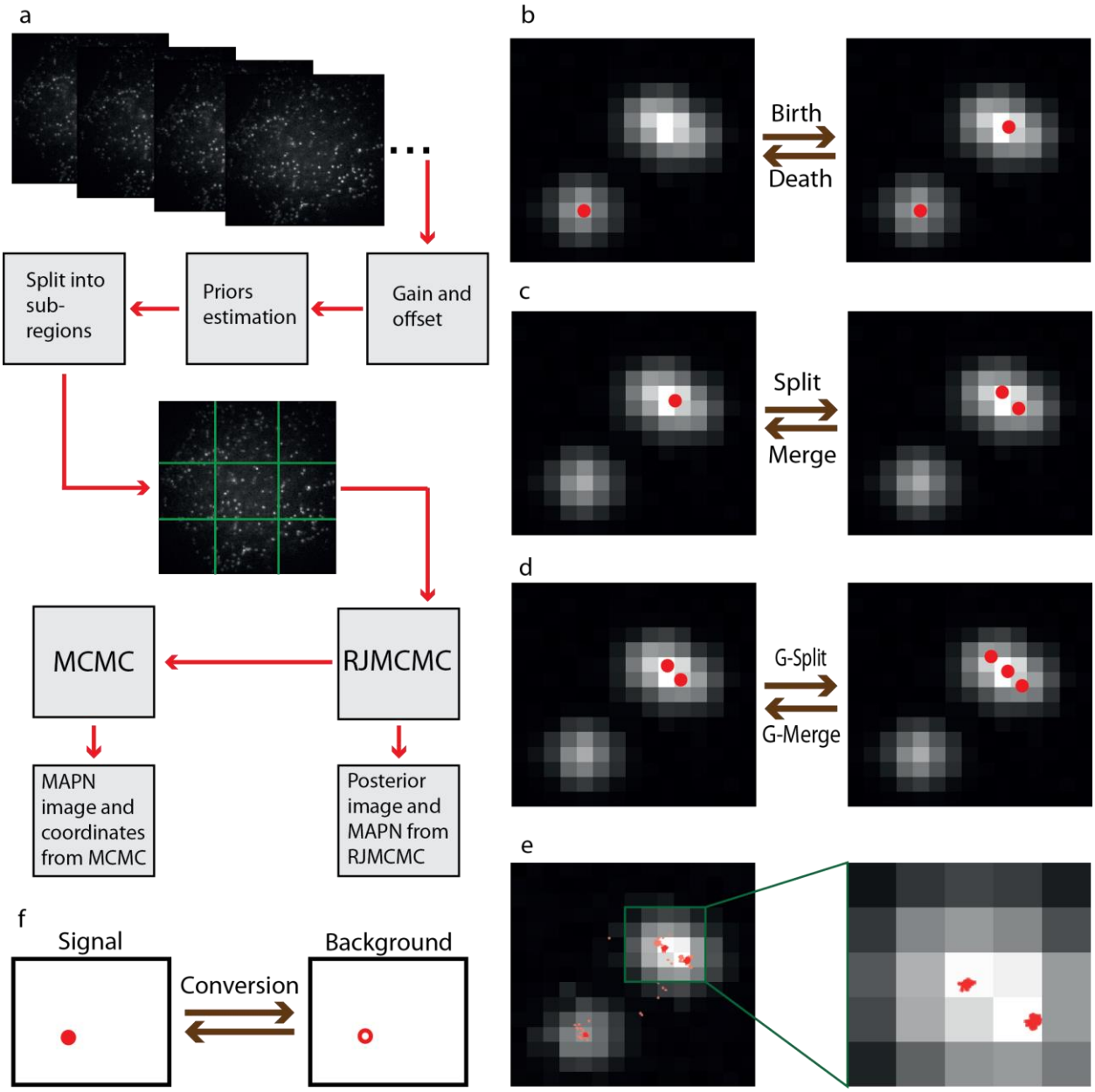


Figure 1: Data flow, jump types and the chain. (a) The data flow. Boxes show stages of the analysis. (b) From left to right, a new spot is detected through a birth event. From right to left, a death event is proposed and an existing emitter is removed. (c) From left to right, an existing emitter splits into two emitters. From right to left, two adjacent emitters merge into a single one. (d) From left to right, photons are taken from *N* existing emitters to make a new one. Right to left, an existing emitter breaks into N pieces which are added to *N* nearby emitters. G-split and G-merge stand for generalized split and generalized merge. (e) Left, the plot of a chain of 8000 jumps, where lighter red shows the burn-in part and the darker red shows the chain after convergence. Right figure depicts the chain after convergence inside the green box. (f) The conversion jump classifies the detected emitters as either signal or background emitters, using the calculated priors on intensities.

The output of the RJMCMC step is a parameter chain whose histogram can be interpreted as a probability density landscape of the emitters that considers all possible numbers and positions of emitters. For example, a single emitter appears as a blob shaped feature in the histogram image of the chain of positions (Fig. 1e), where the width of the blob can be used to calculate the standard error for the position estimation. Combining the chains from all the sub-regions, we build the posterior image for each time frame and then combine all time frames to produce the final posterior reconstruction image. To generate a set of positions and uncertainties from the elements of the RJMCMC chain from the most probable model, the Maximum a Posteriori model of Number of emitters (MAPN), is either used directly or used to initialize a MCMC chain for the MAPN model. The results are used to calculate the positions and associated uncertainties. These returned localizations are then used to reconstruct an image. The posterior probability image includes uncertainty over the number of emitters, whereas the MAPN result can provide locations and standard errors that can be used in subsequent analysis (**Supplementary Fig. 2**).

To assess the performance of BAMF, we analyzed several types of synthetic data and compared the results with that from FALCON[10], SRRF[11] and single-emitter fitting[17]. Jaccard Index (JAC) and localization accuracy are two standard measures to assess the performance of SMLM fitting algorithms[18]. JAC is defined as the ratio of the number of the matched emitters from the sets of found and true emitters to the number of the emitters in the union of those two sets: $\mathrm{JAC}=\frac{\mathrm{ME}}{\mathrm{FE}+\mathrm{TE}}$ where ME, FE and TE refer to the number of matched emitters, found emitters and true emitters, respectively.

Localization accuracy is given by the mean distance between the matched pairs. We used the MAPN result to calculate JAC and accuracy for BAMF. JAC and accuracy were also calculated for FALCON and the single-emitter algorithm. SRRF returns images but not coordinates and therefore was not included. Figure 2 depicts JAC and localization accuracy for the three algorithms. BAMF outperforms the other approaches in both JAC and accuracy.

We compared the results of these algorithms on simulated sequences of data representing two nearby emitters with various separation distances and photons/frame. At 2000 photons/frame, BAMF could distinguish two emitters down to a separation of about $0.25\sigma_{PSF}$, much better than FALCON and SRRF, which could only recognize the data as two close emitters when separated by more than σ_{PSF} (Fig. 2 **& Supplementary Fig. 3**). Here the prior information on emitter intensity helps constrains BAMF to the correct number of emitters. The trend continues to lower photon counts, however the effectively wider intensity prior distribution gives less constraint and the result is a mix of one and two emitter models.

We simulated and analyzed sequences of data with circular test structures of four different radii. FALCON returned more false emitters in the middle of circles where no true emitters reside. SRRF returned disks rather than rings. The single-emitter code returned a circle structure, but much fewer emitters (**Supplementary Fig. 4**).

To evaluate the ability of these methods to deal with structured background, we simulated a dataset with a static ring-like background structure along with in-focus emitters randomly distributed over a cross-like structure. BAMF and SRRF were able to distinguish signal and background structures, however, FALCON attempted to model the background with emitter locations (**Supplementary Fig. 5**).

We tested BAMF performance on dSTORM and DNA-PAINT experimental data (Fig. 3, **Supplementary Fig. 6 and 7**). Figure 3 shows the reconstructions from BAMF, FALCON, SRRF and the single-emitter code on actin imaged using dSTORM. In the two bottom rows in Fig. 3, the arrows show very fine actin filaments. BAMF reveals these actin filaments much better than the other algorithms. The single-emitter algorithm found much fewer localizations in those areas. FALCON does not show as many details as BAMF and has a grid-like artifact that is likely due to the grid used in the deconvolution step in FALCON. The reconstruction from SRRF is missing much of the fine detail. **Supplementary Fig. 6 and 7** shows similar trends in the results from BAMF, FALCON and single-emitter algorithm on actin imaged using DNA-PAINT.

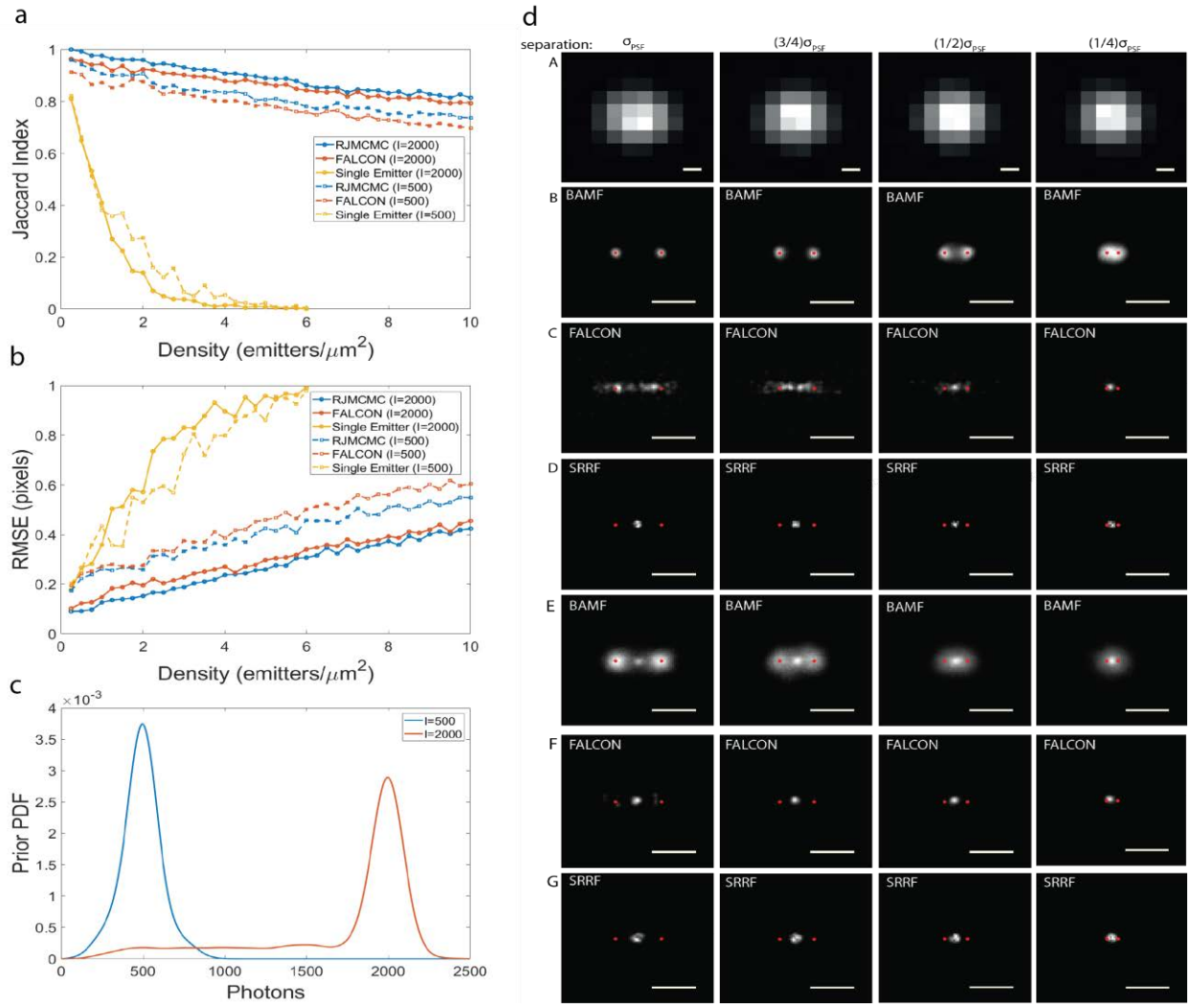


Figure 2: Jaccard index and localization error (accuracy). (a) Jaccard index, (b) localization errors for BAMF, FALCON and single-emitter fitting using 2000 and 500 photons per frame. (c) The intensity priors used to analyze the data. To make these plots for BAMF, the MAPN was used from RJMCMC. (d) represents the ability of BAMF, FALCON and SRRF to distinguish two nearby emitters with different separations. Row A shows a frame of simulated data with different separations. Rows B, C and D result from emitters with 2000 photons, and rows E, F and G show the results from 500 photon emitters. The BAMF super-resolved images are the posterior images containing all the possible models. The scale bars are σ_{PSF}.

The BAMF algorithm takes advantage of prior information to improve the classification of the number of emitters and includes the effect of uncertainty in both classification of number of emitters and the background structure. BAMF generates both a posterior image that contains all sources of uncertainty and a MAPN result that provides coordinates and standard errors of the most probable model. BAMF outperforms other common fitting models both quantitatively on synthetic data and subjectively on experimental data. The BAMF concept could be extended to include temporal information and/or 3D imaging approaches.

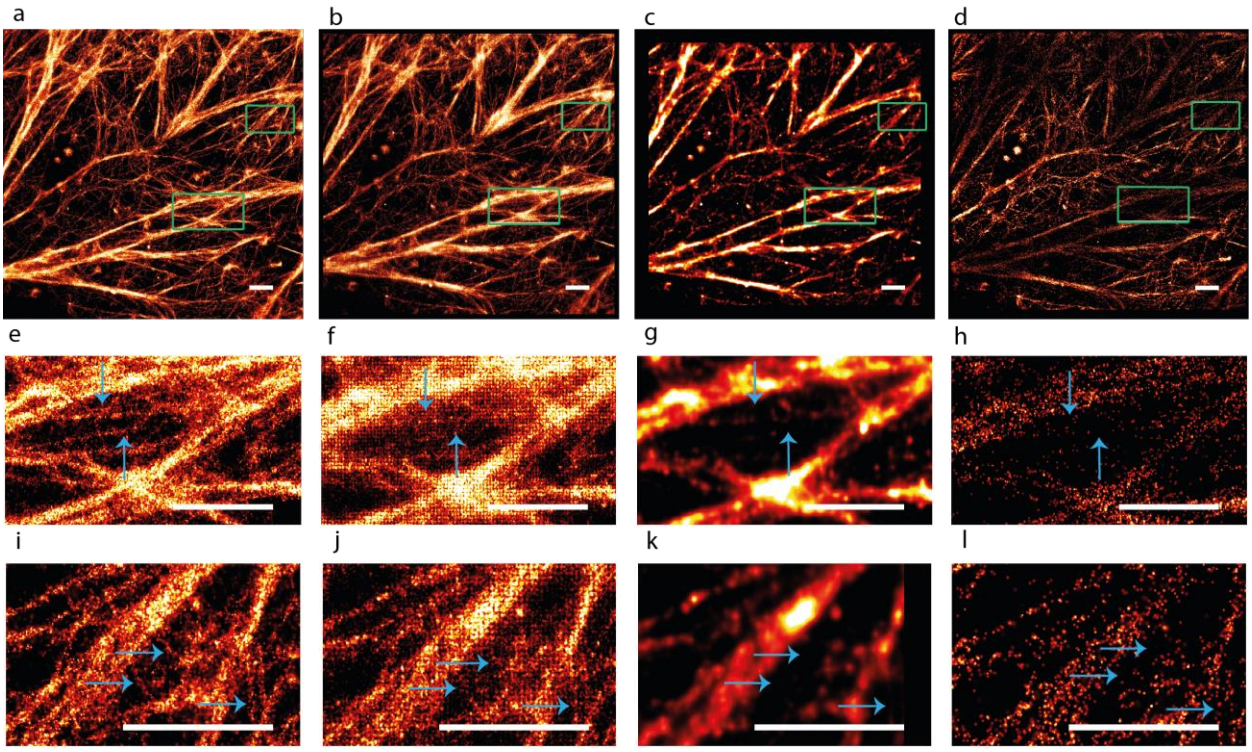


Figure 3: Reconstructions from BAMF, FALCON, SRRF and single-emitter fitting for dSTORM data of actin filaments. Reconstructions from (a) BAMF, (b) FALCON, (c) SRRF, (d) single-emitter fitting. The reconstructions for BAMF were made using the MAPN from MCMC. (e-h) Zoom of the top green square. (i-l) Zoom of the bottom green square. The blue arrows point to some fine details in the magnified regions. The scale bars are 1 µm.

Online Methods

BAMF's parameters

The jump sizes for each parameter within the RJMCMC step were adjusted to yield an acceptance rate of 25% to 50% (**Supplementary Note 1**). For the moves in position, intensity, and the offset background, jumps were selected from zero-mean normal distributions with the sigma ranging from 0.05 to 0.1 pixel, 5 to 10 photons, and 1 photon, respectively. For the burn-in chain, 3000 jumps and jump probabilities of ($P_{\text{In-model}}$, P_{Birth}, P_{Death}, P_{Split}, P_{Merge}, $P_{\text{G-split}}$, $P_{\text{G-merge}}$, $P_{\text{Conversion}}$)=(0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1) were used, while for the post-burn-in chain, we

had 2000 jumps and jumps probabilities of ($P_{In\text{-}model}$, P_{Birth}, P_{Death}, P_{Split}, P_{Merge}, $P_{G\text{-}split}$, $P_{G\text{-}merge}$, $P_{Conversion}$)=(0.4, 0.05, 0.05, 0, 0, 0.15, 0.15, 0.2). For the JAC measurements and two-emitter simulations, the density of emitters required 20000 jumps and 10000 jumps for burn-in and post-burn-in respectively, to guarantee the chain convergence.

Implementation

Image pre-processing and computational analyses were performed in MATLAB by employing the image processing, statistics and machine learning and parallel toolboxes (MathWorks Inc.). The C++-codes for RJMCMC were compiled into mex-files that could be called from inside MATLAB. All codes were CPU based and were parallelized using the MATLAB parallel computing toolbox. The single-emitter code was implemented on GPUs using CUDA codes compiled into ptx-files that could be called inside MATLAB. An i7, 3.64 GHz CPU with a GTX 750 GPU was used to process the simulated data and part of the experimentally acquired data. Part of the experimental data was also analyzed in a cluster employing 16 core Intel Xenon, 2.6 GHz CPUs, available at the UNM Center for Advanced Research Computing (CARC).

Synthetic data generation

To generate synthetic data, emitters were placed in random positions with the uniform density ρ, except where mentioned. A trace of the blinking events of each emitter was produced using the duty cycle parameters, k_{on} and k_{off}, which are respectively the rate of emitters going from off to on and on to off, such that the density of the on-emitters is proportional to the ratio of k_{on} to $k_{on} + k_{off}$. To imitate realistic conditions, random times for the emitters to turn on and on-durations were chosen, using exponential distributions with mean values of $k_{on} + k_{off}$ and k_{on}, respectively. Next, a uniform background was added to the generated data and corrupted with Poisson noise.

Tests on synthetic data

Jaccard index (JAC) and accuracy were calculated by making use of synthetic data generated in a region of 24x24 pixels, where 2 pixels at the edges were left empty, with the pixels of width 100 nm. A ground truth of 1000 emitters per μm^2 was generated and the duty cycle parameters were adjusted to provide a desired final per-frame density. 40 sequences of 100 frames of data were generated with an average density of on-emitters ranging from 0.25 to 10 emitters per μm^2 over a uniform background. The width of the PSF and the background were, respectively, 1.2 pixel and 20 photons. The intensity of the emitters that were on during an entire frame exposure was 2000 photons per frame and less if they were on for a fraction of the exposure time.

The localization accuracy was measured by the root mean square error (RMSE) to the true locations. In order to calculate JAC, matched pairs between the MAPN result (used directly from RJMCMC chain) and the true emitters were found. To discover the pairs of the matched emitters, the cost matrix of the found emitters with the true emitters were minimized using the Hungarian algorithm[19], and those pairs where the corresponding cost element was smaller than the PSF size (1.2 pixel) were used in the JAC[18].

For the synthetic circles, data sequences of 2000 frames with a size of 10x10 pixels were produced with circles of radii of $0.416\sigma_{PSF}$, $0.625\sigma_{PSF}$, $0.833\sigma_{PSF}$ and $1.041\sigma_{PSF}$, and with a PSF width, mean intensity and background of 1.2 pixel, 2000 photons and 20 photons, respectively. Uniformly distributed emitters at 1000 per µm were used to generate the circles and then by adjusting the duty cycle parameters brought to an average density of 4.5 on-emitters per frame. The localizations returned by the single-emitter code, FALCON and BAMF were used to reconstruct the final images of circles. SRRF does not return any localizations but does return a reconstruction which was included in **Supplementary Fig. 4**. Since FALCON does not return any localization accuracy, the same accuracy (σ=0.06 pixel, which is the mode of localization accuracy returned by BAMF) was used to reconstruct the final images for the three algorithms. The reconstructions from BAMF with localization accuracy better than 0.25 pixel are also included in **Supplementary Fig. 4**.

For the two emitters test, two sets of 100 frames of data were synthesized using two constantly on emitters for each separation. The PSF width, intensity and background were 1.2 pixel, 500 or 2000 photons (representing dim and bright emitters in empirical data), and 20 photons respectively. The priors used for JAC and accuracy measurements with average intensities of 500 (dim) and 2000 (bright) photons (Fig. 2) were employed because the two emitters were constantly on and heavily overlapped and the single-emitter code was not able to estimate their intensities. This is the only exception to the protocol described in the supplement to obtain the intensity priors.

A sequence of data of 2000 frames with size of 32x32 pixels was generated for the test of separation of signal emitters from background emitters (**Supplementary Fig. 5**). The structured background was produced by placing 18 constantly on-emitters on positions equally spaced on a ring with a radius of 10 pixels. The PSF size and intensity of these emitters were 1.5 pixel and 400 photons per frame. For the signal, we synthesized 600 uniformly distributed emitters per μm^2 inside a cross with PSF size and average intensity of 1.2 pixel and 2000 photons and obtained 6.5 activated emitters per frame by tuning the duty cycle parameters. The final data set was produced by adding the two synthesized data sets with an offset background of 20 photons corrupted with Poisson noise.

Experimental data analysis

dSTORM actin data: The single-emitter code was used to find the PSF size and the prior distribution for the photons/emitter/frame (intensity) parameter. The PSF size was used for BAMF and FALCON. *DNA-PAINT actin data:* The single-emitter code was used to find the PSF size and the prior distribution. The provided library function *findPSF_SMA* (**Supplementary Note 3**) was used to calculate the PSF for BAMF. The found PSF size was used for FALCON.

The returned coordinates from the single-emitter code and BAMF were then filtered, employing the coordinates' uncertainties, in order to reconstruct the final images. FALCON does not return any uncertainty and hence the returned coordinates were used to produce the reconstructions directly. SRRF returns neither coordinates nor uncertainties, so only the returned reconstructions from it were used.

Cell lines and reagents

HeLa cells were cultured in Dulbecco's Modified Eagle Medium (Life Technologies # 10313-v021) supplemented with 10% fetal bovine serum (HyClone), penicillin-streptomycin, and 2 mM L-glutamine at 37 ˚C and 5% CO_2. Actin microfilaments were labeled with 0.56 µM Alexa Flour™ 647 Phalloidin (A22287, ThermoFisher Scientific) diluted in PBS.

Cell fixation and labeling

dSTORM actin imaging: All labeling and washing steps were carried out at room temperature unless stated otherwise. Cells were seeded onto #1.5 coverslip glass 6 well chambers (LabTek) to adhere for 4-24 h. Cells were fixed for 1 hour in a 3% Glyoxal + 20% Ethanol+ 0.75% Ascetic Acid in DI water[20]. pH was adjusted to 5 by adding drops of 1 M NaOH. Cells were washed 2x in PBS and kept in $NaBH_4$ for 10 min to reduce background fluorescence, followed by 2x wash with PBS. To quench reactive crosslinkers, the samples were kept in 10 mM Tris for 10 min, followed by 2 washes with PBS. Finally, samples were blocked in 4% BSA + 0.1% Triton X-100 for 1 hour, followed by 2 more PBS washes. Primary antibody was incubated at 2 µg ml^{-1} in 5% BSA + 0.05% Triton X-100 in PBS for 15 min. Samples were washed 1x with PBS and labeled with 0.56 µM Alexa Flour™ 647 Phalloidin for 4 hours. *DNA-PAINT actin imaging*: Cos7 cells were fixed and labeled with an actin-binding affimer linked to a DNA-PAINT docking strand as described previously[23].

Super-resolution imaging

dSTORM actin imaging: Imaging was performed in a standard dSTORM imaging buffer[21] with an enzymatic oxygen scavenging system and primary thiol: 50 mM tris, 10 mM NaCl, 10% w/v glucose, 168.8 U/ml glucose oxidase (Sigma #G2133), 1404 U/ml catalase (Sigma #C9322), and 60 mM 2-aminoethanethiol (MEA), pH 8.5. To mount the samples prepared on 25 mm coverslips, an Attofluor cell chamber (A-7816, Life Technologies) was used and 1.5 ml of

dSTORM imaging buffer was added. To prevent oxygen permeation into the buffer, a clean 25 mm coverslip was used to seal the chamber. The sample was mounted on the stage of the microscope with a custom designed chamber holder. The imaging system was built on an inverted microscope (IX71, Olympus America Inc.). An *xyz* piezo stage (Mad City Labs, Nano-LPS100) mounted on a *x-y* manual stage was installed on the microscope for cell location and brightfield registration. A mounted LED with the wavelength of 850 nm (M850L3, Thorlabs) was used for brightfield illumination. Brightfield images was collected on a complementary metal-oxide semiconductor (sCMOS) camera (DCC1545M, Thorlabs) after reflecting by a short-pass dichroic beam splitter (FF750-SDi02, Semrock) and passing through a single-band bandpass filter (FF01-835/70-25, Semrock). A 638 nm laser was used (collimated from a laser diode, L638P200, Thorlabs) coupled into a single mode fiber and focused onto the back focal plane of the 1.49 NA objective lens (UAPON 100XOTIRF, Olympus America Inc.). Emission for super-resolution data was collected through a short-pass dichroic beam splitter (FF750-SDi02, Semrock) and a single-band bandpass filter (FF01-692/40-25, Semrock) on an iXon 860 electron-multiplying charge-coupled device (EM CCD) camera (Andor Technologies, South Windsor, CT). The EMCCD gain was set to 90, and frames were 128 × 128 pixels with a pixel size of 0.1192 µm. All the instruments were controlled by custom-written software in MATLAB (MathWorks Inc.). Imaging was performed with TIRF illumination. Images were acquired at 5 ms exposure time for a total of 12000 frames. Brightfield registration was performed to correct for drift after every 3000 frames as previously described[22]. *DNA-PAINT actin imaging*: Data was collected with 50 ms exposure time for 300k frames using 800 pM P1 imager strand concentration and 3.3 kW/cm^2 laser power at 561 nm [23].

References

1. Hell, S. W. & Wichmann, J. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. *Opt. Lett.* **19,** 780 (1994).

2. Betzig, E. *et al.* Imaging Intracellular Fluorescent Proteins at Nanometer Resolution. *Science (80-.).* **313,** 1642–1645 (2006).

3. Lidke, K. A., Rieger, B., Jovin, T. M. & Heintzmann, R. Superresolution by localization of quantum dots using blinking statistics. *Opt. Express* **13,** 7052 (2005).

4. Rust, M. J., Bates, M. & Zhuang, X. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM). *Nat. Methods* **3,** 793–796 (2006).

5. Nieuwenhuizen, R. P. J. *et al.* Measuring image resolution in optical nanoscopy. *Nat. Methods* **10,** 557 (2013).

6. Huang, F., Schwartz, S. L., Byars, J. M. & Lidke, K. A. Simultaneous multiple-emitter fitting

for single molecule super-resolution imaging. *Biomed. Opt. Express* **2,** 1377 (2011).

7. Cox, S. *et al.* Bayesian localization microscopy reveals nanoscale podosome dynamics. *Nat. Methods* **9,** 195–200 (2011).

8. Quan, T. *et al.* High-density localization of active molecules using Structured Sparse Model and Bayesian Information Criterion. *Opt. Express* **19,** 16963–16974 (2011).

9. Zhu, L., Zhang, W., Elnatan, D. & Huang, B. Faster STORM using compressed sensing. *Nat. Methods* **9,** 721–723 (2012).

10. Min, J. *et al.* FALCON: fast and unbiased reconstruction of high-density super-resolution microscopy data. *Sci. Rep.* **4,** 4577 (2015).

11. Gustafsson, N. *et al.* Fast live-cell conventional fluorophore nanoscopy with ImageJ through super-resolution radial fluctuations. *Nat. Commun.* **7,** 12471 (2016).

12. Small, A. & Stahlheber, S. Corrigendum: Fluorophore localization algorithms for super-resolution microscopy. *Nat. Methods* **11,** 971–971 (2014).

13. Green, P. Reversible Jump Markov Chain Monte Cralo computation and Bayesian model determination. *Biometrika* **82,** 711–732 (1995).

14. Richardson, S. & Green, P. J. On Bayesian analysis of mixtures with an unknown number of components. *J. R. Stat. Soc. Ser. B* **59,** 731–792 (1997).

15. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **21,** 1087–1092 (1953).

16. Hastrings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57,** 97–109 (1970).

17. Smith, C. S., Joseph, N., Rieger, B. & Lidke, K. A. Fast, single-molecule localization that achieves theoretically minimum uncertainty. *Nat. Methods* **7,** 373 (2010).

18. Sage, D. *et al.* Quantitative evaluation of software packages for single-molecule localization microscopy. *Nat. Methods* **12,** 717–724 (2015).

19. Kuhn, H. W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **2,** 83–97 (1955).

20. Richter, K. N. *et al.* Glyoxal as an alternative fixative to formaldehyde in immunostaining and super-resolution microscopy. *EMBO J.* **37,** 139 LP-159 (2018).

21. Dempsey, G. T., Vaughan, J. C., Chen, K. H., Bates, M. & Zhuang, X. Evaluation of fluorophores for optimal performance in localization-based super-resolution imaging. *Nat. Methods* **8,** 1027 (2011).

22. Valley, C. C., Liu, S., Lidke, D. S. & Lidke, K. A. Sequential Superresolution Imaging of

Multiple Targets Using a Single Fluorophore. *PLoS One* **10,** e0123941 (2015).

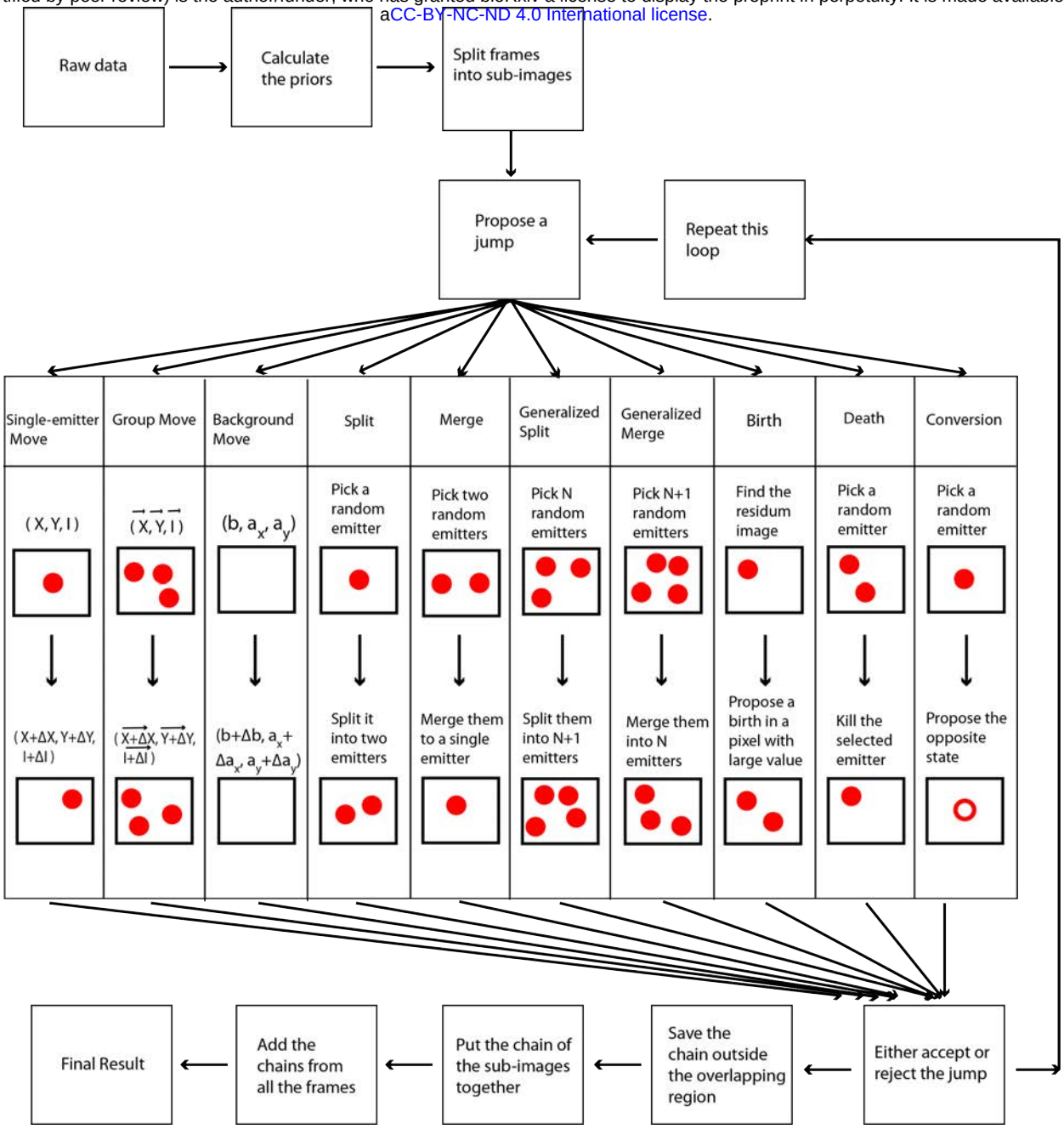
23. Schlichthaerle, T. *et al.* Site-Specific Labeling of Affimers for DNA-PAINT Microscopy. *Angew. Chemie Int. Ed.* **57,** 11060–11063 (2018).

Supplementary Information
Bayesian Multiple Emitter Fitting using Reversible Jump Markov Chain Monte Carlo

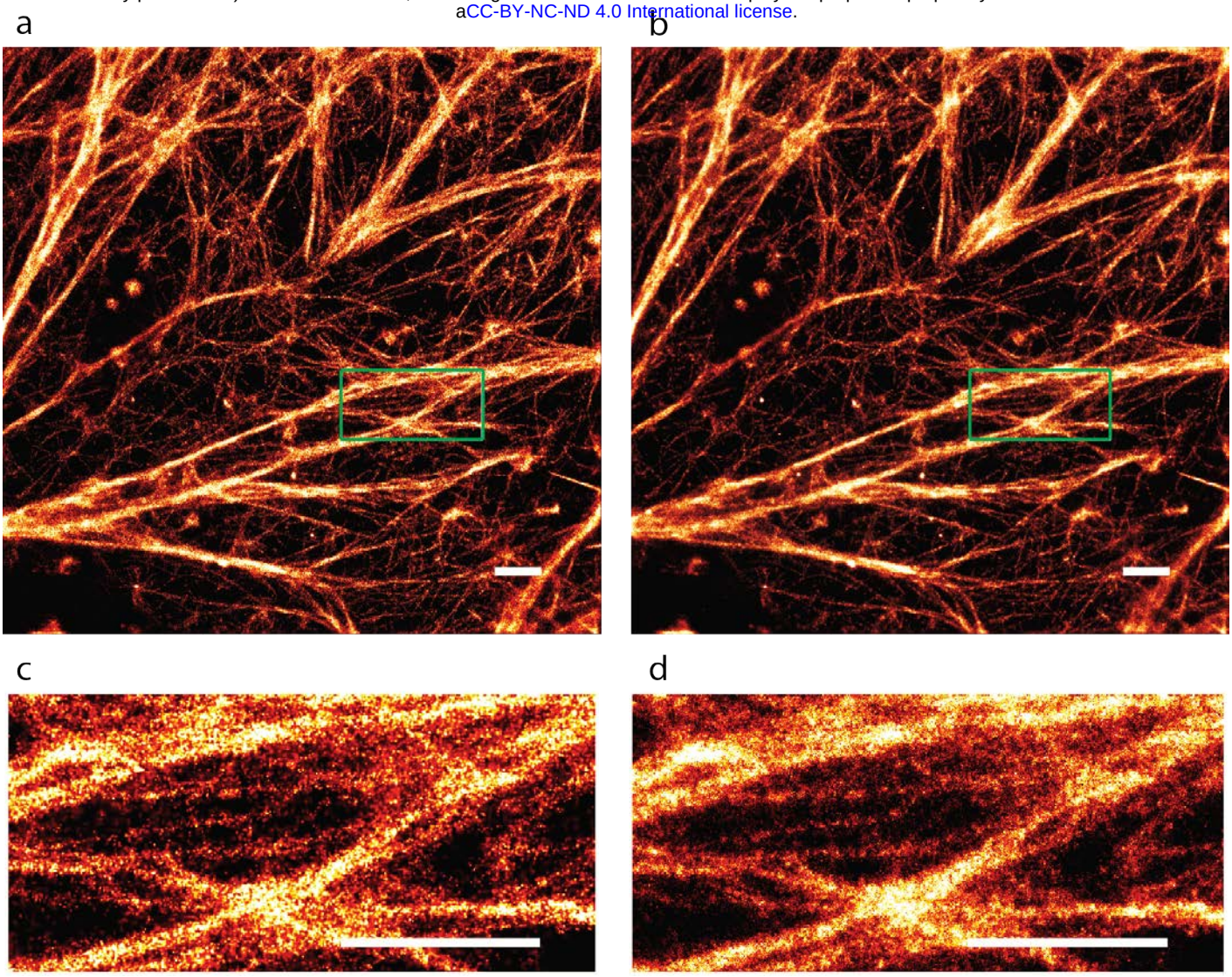
Mohamadreza Fazel[1], Michael J. Wester[2], Hanieh Mazloom-Farsibaf[1], Marjolein B. M. Meddens[1], Alexandra Eklund[3,4], Thomas Schlichthaerle[3,4], Florian Schueder[3,4], Ralf Jungmann[3,4] and Keith A. Lidke[1]

[1] Department of Physics and Astronomy, University of New Mexico Albuquerque, New Mexico, USA.
[2]Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico, USA.
[3]Department of Physics and Center for Nanoscience, Ludwig Maximilian University, Munich, Germany.
[4]Max Planck Institute of Biochemistry, Martinsried, Germany.

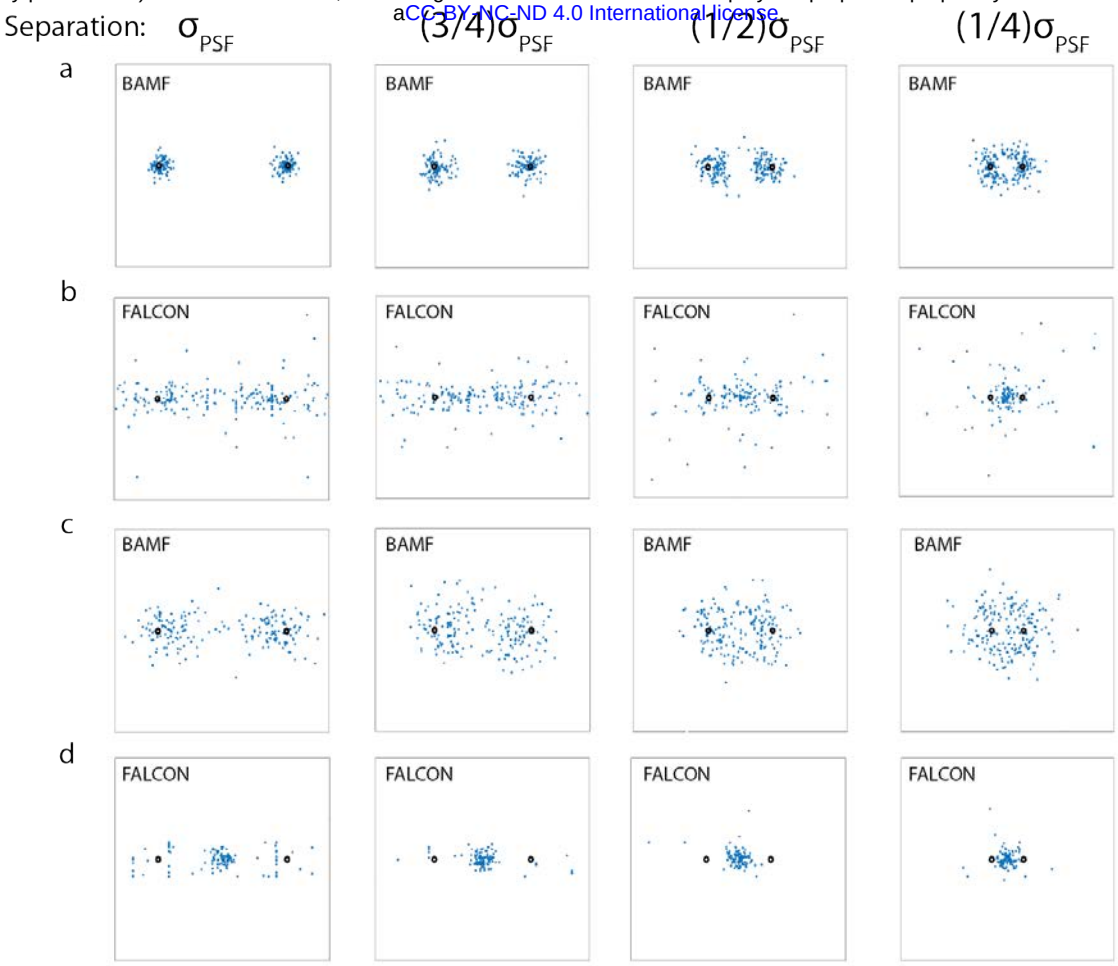
January 24, 2019



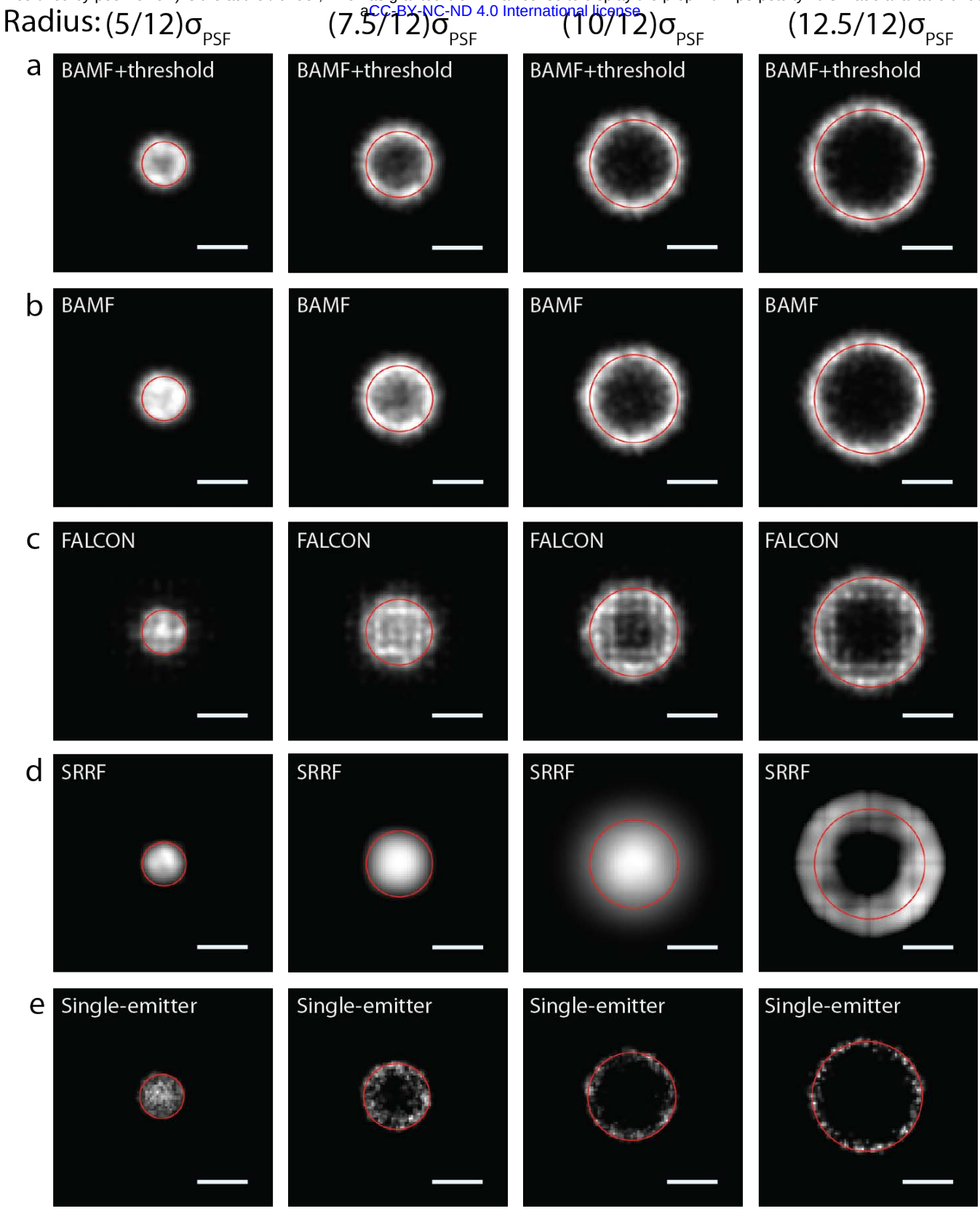
Supplementary Figure 1: BAMF concept: Schematic description of the BAMF algorithm. The flow of the data is described and the ten different jump types used in RJMCMC are depicted in the large box.



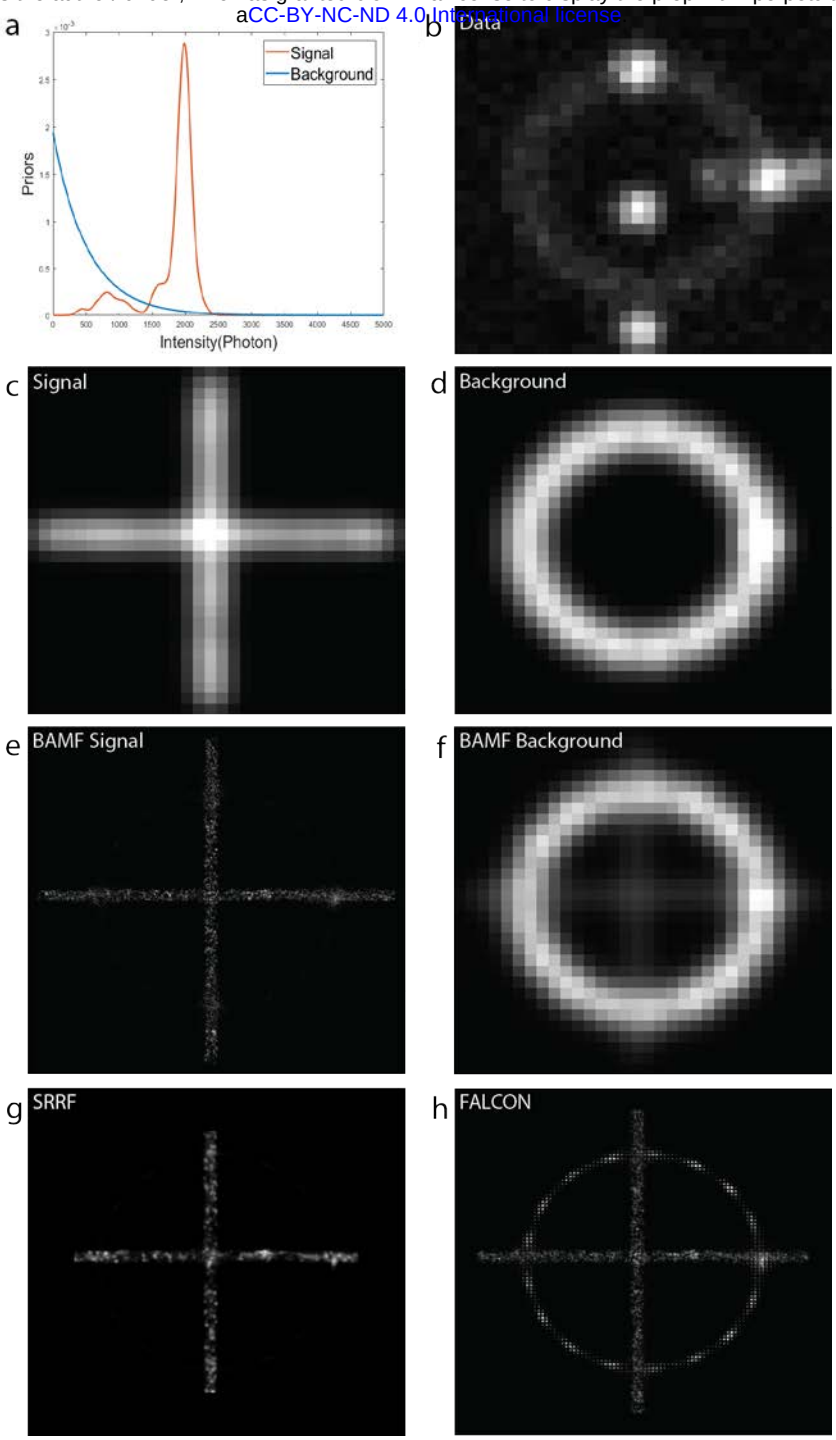
Supplementary Figure 2: Comparison of MAPN image and posterior image for dSTORM data of actin filaments from Hela cells. a and b, respectively, show the MAPN and posterior images of the actin filaments from BAMF algorithm. c and d are zooms of the green squares. The scale bars are 1 μm.



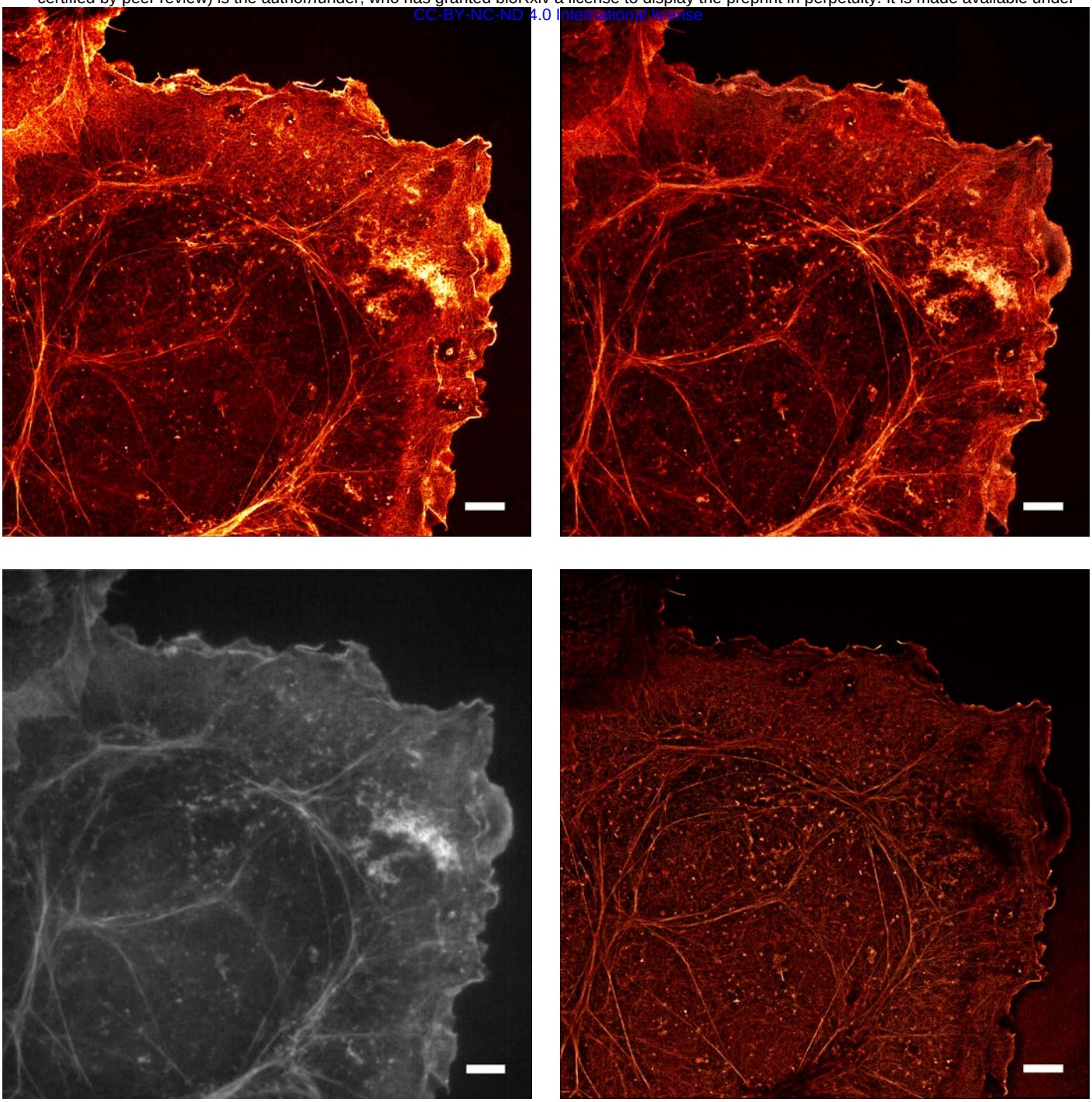
Supplementary Figure 3: Plot of found positions for two nearby emitters with different separations and intensities. The black circles represent the true locations while the blue dots stand for the found positions by BAMF and FALCON. The true locations of the first, second, third and fourth columns respectively, have separations of σ_{PSF}, $\frac{3}{4}\sigma_{\mathrm{PSF}}$, $\frac{1}{2}\sigma_{\mathrm{PSF}}$ and $\frac{1}{4}\sigma_{\mathrm{PSF}}$, where $\sigma_{\mathrm{PSF}} = 1.2$ pixels. Rows a and c show the plots of MAPN from RJCMC for intensities of 2000 and 500 photons, respectively. Rows b and d, respectively, depict the results from FALCON for intensities of 2000 and 500 photons.



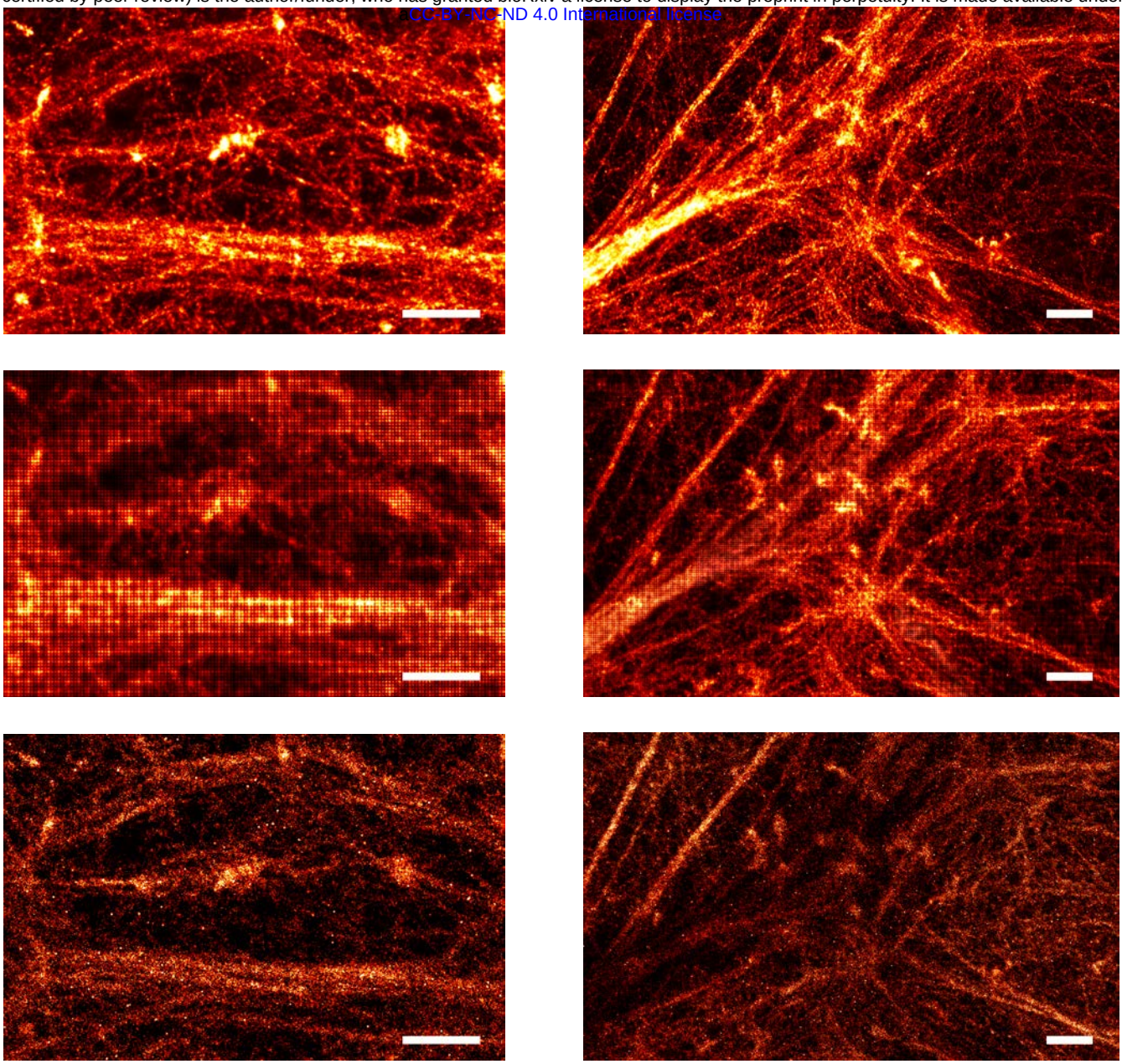
Supplementary Figure 4: Reconstructions for circles of emitters from FALCON, SRRF and the single-emitter code are depicted in rows c, d and e, respectively. BAMF reconstructions are shown in rows a and b, which are MAPN from MCMC. Row b represents the circles reconstructed using all the returned localizations, while row a shows reconstructions from localizations with accuracy better than 0.25 pixel. The circles in the first, second, third and fourth columns, respectively, have radii of $\frac{5}{12}\sigma_{\mathrm{PSF}}$, $\frac{7.5}{12}\sigma_{\mathrm{PSF}}$, $\frac{10}{12}\sigma_{\mathrm{PSF}}$ and $\frac{12.5}{12}\sigma_{\mathrm{PSF}}$. The average density of on-emitters is 4.5 emitters per frame. The scale bars are σ_{PSF}.



Supplementary Figure 5: Separation of the signal from the structured background. a) Plots of the priors used for signal and background emitters. b) One frame of the simulated data. c) Noise-free simulated signal. d) Noise-free simulated structured background. e) Super-resolved reconstruction from BAMF. f) The structured background reconstructed using the background emitters found by BAMF. g) Super-resolved reconstructions by SRRF. h) Super-resolved reconstructions by FALCON. Each image was scaled independently.



Supplementary Figure 6: Actin filaments in Cos7 cells using DNA-PAINT. Reconstructions from BAMF, FALCON and the single-emitter fitting code for DNA-paint data. (top left) Posterior image from BAMF, (top right) reconstruction from FALCON, (bottom left) wide field image, (bottom right) reconstruction from the single-emitter fitting code. The scale bars are 5 μm.



Supplementary Figure 7: Selected magnified regions from BAMF, FALCON and the single-emitter code reconstructions in Supplementary Fig. 6. The first, second and third rows display the magnified regions from BAMF, FALCON and the single-emitter algorithm, respectively. Each column represents the same regions from different fitting procedures. FALCON and single-emitter code do not reveal as many details as BAMF. FALCON has a grid-like artifact. The scale bars are 1 μm.

1 Supplementary Note 1: Reversible Jump Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are restricted to problems where the number of the parameters is fixed [15,16]. Reversible Jump Markov Chain Monte Carlo (RJMCMC) is an extension of MCMC, in which jumps between parameter spaces with different numbers of parameters are permitted and hence makes inferences about the number of parameters as well as the parameters themselves [13,14]. In other words, the number of the parameters is one of the unknowns in RJMCMC. For the multiple-emitter fitting problem that we will address here, the number of parameters and the parameters correspond to the number of emitters and their locations and intensities, an offset background and two slopes of the offset plane along the X and Y axes. There are ten different types of jumps in BAMF (Supplementary Fig. 1). The conversion jump classifies the found emitters as either background or signal. There are three different types of inside model moves (single-emitter move, group move, background move). These moves make inferences about the positions and intensities of the emitters and do not hop between different models. The other six jumps allow removing or adding emitters to the models and jumping to a new model with either one less or one more emitter. The first jump is split, which tests if there is possibility for an existing emitter to split into two adjacent emitters. Merge is the inverse of split. It considers the chance of two close emitters to really be a single emitter. Generalized split investigates the possibility of N nearby emitters to actually be $N+1$ emitters. Generalized merge calculates the probability of $N+1$ adjacent emitters to consolidate into N emitters. The generalized split and generalized merge jumps are especially useful in dense super-resolution data. Birth explores different parts of the data to see if a new emitter can be added to the current model. Death is the opposite of birth and examines the feasibility of eliminating one of the existing emitters from the current model.

1.1 PSF model, Likelihood, Priors and Posterior

RJMCMC can be used to implement a Bayesian approach that samples from the posterior of a system in order to learn about that system. The posterior is proportional to the product of the likelihood and priors.

1.1.1 PSF model

Photons from a single emitter have an approximate spatial Gaussian distribution on the camera, where (1) gives photon counts based on this approximation. For cases where the Gaussian function is not a reasonable approximation, the PSF can be acquired experimentally and employed to calculate the likelihood numerically (Supplementary Note 4). The integral of the Gaussian distribution over the kth pixel gives the average number of photons from the ith emitter in that pixel.

$$\Delta_{k,i} = \frac{I_i}{2\pi {\sigma_{\mathrm{PSF}}}^2} \int_{x_k-0.5}^{x_k+0.5} \int_{y_k-0.5}^{y_k+0.5} \exp\left[\frac{(x-x_i)^2+(y-y_i)^2}{2\pi {\sigma_{\mathrm{PSF}}}^2}\right] dy\, dx \tag{1}$$

where $\Delta_{k,i}, \sigma_{\mathrm{PSF}}, I_i, x_k, y_k, x_i$ and y_i are, respectively, the number of the photons in the kth pixel from the ith emitter, the half width of the Gaussian distribution, the total number of the photons from the emitter, the center of the kth pixel and the position of the emitter.

1.1.2 Likelihood

The total photon count in the kth pixel is the sum of the photons from all the existing emitters and the background.

$$\lambda_k(N) = a_x x + a_y y + b + \sum_{i=1}^{N} \Delta_{k,i} \tag{2}$$

where λ_k, a_x, a_y, b and N, respectively, denote the total number of the photons in the kth pixel, the slope of the background along the X and Y axes, the offset background and the number of emitters. Equation (2) yields the average photon counts for the pixel k for a fixed exposure time. Consequently, the number of the photons in pixel k has a Poisson distribution

$$L_k(D|\theta) = \frac{\lambda_k(N)^{D_k} e^{-\lambda_k(N)}}{D_k!} \tag{3}$$

where D, D_k and θ represent data, the number of the photons in pixel k of the acquired data, and the set of the parameters ($\theta = (\vec{x}, \vec{y}, \vec{I}, N, b, a_x, a_y)$). Due to the independence of the pixels, the likelihood of the frame is given by the product of the likelihoods of all the pixels in that frame [6,17]

$$L(D|\theta) = \prod_k L_k(D|\theta) \tag{4}$$

Parameter	Prior
position	uniform
number of emitters	Poisson
signal intensity	kernel density estimator
background intensity	exponential
offset background	gamma
offset background slopes	normal

Supplementary Table 1: Table of prior functions.

1.1.3 Priors and Posterior

The product of the likelihood with the prior of the parameters $P(\theta)$ is proportional to the posterior

$$p(\theta|D) = \frac{L(D|\theta)P(\theta)}{P(D)} \tag{5}$$

where $P(D)$ is called the evidence. Evidence is the normalization coefficient of the posterior:

$$P(D) = \int L(D|\theta)P(\theta)d\theta \tag{6}$$

We employ RJMCMC to estimate the positions, the intensities of the emitters, the number of the emitters, the offset background and its slopes, and therefore their priors have to be included in the calculations. We take the prior on the positions to be a uniform distribution over the frame. Because there might be some emitters outside the frame but still close enough to its edges so that portions of the PSFs are still observable on the frame, we allow the detection of emitters that are located up to 2 pixels away from the edges outside the frame (this can be modified by user). Hence the prior is a uniform distribution over this extended range. The number of the emitters inside the region of interest has a Poisson distribution with the mean value ρW^2, where ρ is the density of the emitters per pixel, given by the user as an input, and W is the width of the region of interest in pixels. The offset background and its slopes have gamma and normal distributions as priors, respectively. The intensity distribution of the emitters heavily depends on several conditions such as the on and off rate of the emitters, the labeling method, etc. Therefore, it is not feasible to consider a specific prior distribution for general data. Because of that, the signal intensity prior should be provided by the user. The signal intensity prior is given as a numeric array, enabling the code to deal with any intensity distribution. We utilized an exponential distribution for the background intensity prior. A MATLAB library consisting of several methods is provided along with the BAMF code that calculates the aforementioned priors; see Supplementary Note 3 and Supplementary Table 1. In this library, the intensity and background priors are obtained by fitting a smoothed kernel density estimator and gamma probability function to the intensities and offset background returned by a very fast single-emitter code [17].

1.2 Jump Types

BAMF defines three main types of jumps to sample the posterior of the system in the core RJMCMC algorithm. The first type of jump is the within model jump, which optimizes the parameters while not changing the number of the parameters or the number of the emitters. The second type of jump are those that allow the chain to move between different parameter spaces, varying the number of emitters for the given data. The third type of jump is between the background and signal states, which sorts the signal emitters out from the background emitters.

1.2.1 Proposing a Jump

In the following, we explain how these jumps are proposed and how we accept or reject them. First, the chain is initialized to an arbitrary state, containing an arbitrary number of emitters with random locations and intensities. We then pick a random number from a uniform distribution over the interval $[0, 1]$, and based on this random number propose a jump. The occurrence probability of each jump is given by the user, which we call $P_J, P_S, P_M, P_{GS}, P_{GM}, P_B, P_D$ and P_C respectively for the probabilities of proposing a within model jump, a split, a merge, a generalized split, a generalized merge, a birth, a death or a conversion (for instance $P_J = P_S = P_M = P_{GS} = P_{GM} = P_B = P_D = P_C = 1/8$). Note that we have

$$P_J + P_S + P_M + P_{GS} + P_{GM} + P_B + P_D + P_C = 1. \tag{7}$$

A random number *rand* is taken from the interval $[0, 1]$. The jump n (where the jumps above are labeled sequentially) is then proposed if

$$\sum_{i=1}^{n-1} P_i \leq \text{rand} < \sum_{i=1}^{n} P_i \tag{8}$$

After proposing the jump, we calculate the model and the acceptance probability. Next, another random number is picked in the interval $[0, 1]$. The jump will be accepted if the acceptance probability is larger than this random number; otherwise it will be rejected.

1.2.2 Within Model Moves

A within model move does not change the number of the emitters or the number of parameters. It only changes the parameters related to the background, positions and intensities of the existing emitters. The move sizes for different parameters in BAMF algorithm are taken from normal distributions. The widths of these normal distributions determine the size of the moves as well as the acceptance rates of the within model moves [23].

Single-emitter Move

An emitter or a set of neighboring emitters is taken from the list of current emitters at random. The new parameters, which we show by prime, are given by

$$\vec{x'} = \vec{x} + \Delta\vec{x}, \quad \vec{y'} = \vec{y} + \Delta\vec{y}, \quad \vec{I'} = \vec{I} + \Delta\vec{I} \tag{9}$$

where $\Delta x, \Delta y$ and ΔI are taken from normal distributions with centers at the origin and the widths are provided by the user for determining the jump sizes. The acceptance probability of a move from θ to θ' for in-model updates is given by

$$P = \min\left\{1, \frac{p(\theta'|D)}{p(\theta|D)}\right\} \tag{10}$$

where $p(\theta'|D)$ and $p(\theta|D)$ denote the posterior of the proposed parameters and the current parameters. The ratio in (10) can be expanded as

$$\frac{p(\theta'|D)}{p(\theta|D)} = \left(\prod_k e^{\lambda_k - \lambda'_k} \left(\frac{\lambda'_k(N)}{\lambda_k(N)}\right)^{D_k}\right) \left(\prod_{n=1}^{N} \frac{P(I'_n)}{P(I_n)}\right) \tag{11}$$

where k and n count the pixels and emitters, respectively. The first parenthesized expression is the ratio of the likelihoods, while the second one is from the prior intensity in which the position priors are canceled. Note that the evidences cancel, simplifying the calculations tremendously because obtaining the evidence involves computing very complicated integrals (6).

Group Move

In a group move, a group of neighboring emitters is found, and then we take a random subset of that group. Next, the intensities of the chosen emitters are redistributed among themselves, and we move the emitters so that their center of mass is conserved. In other words, we keep the center of mass and the number of photons conserved in this move. This jump is especially worthwhile for denser data sets as it helps to escape from local maximums in the dense regions where a single-emitter move fails. The acceptance probability is calculated utilizing (10,11).

Background Move

A background move does not deal with any of the emitters' parameters. Instead, it attempts to optimize the offset background and its slopes along the X and Y axes. The jumps are taken from normal distributions where the step size in offset background is provided by the user.

$$b' = b + \Delta b, \quad a'_x = a_x + \Delta a_x, \quad a'_y = a_y + \Delta a_y \tag{12}$$

The acceptance probability is computed as before.

1.2.3 Split and Merge

Split and merge are two complementary reversible mechanisms that allow for exploring the possibility of a different number of emitters in a local area. Basically, in each step single emitters are allowed to divide into two, or two neighboring emitters are allowed to combine.

Split

On proposing a split, an emitter is chosen at random. After splitting this emitter into two emitters, we will have two sets of parameters rather than one, and thus we choose to calculate these new sets of parameters based on conservation principles. Our rules for the parameters of the new emitters are that the total intensity (zero moment) and center of mass (first moment) are conserved.

$$\begin{aligned} I_j &= I_{j^1} + I_{j^2} \\ I_j x_j &= I_{j^1} x_{j^1} + I_{j^2} x_{j^2} \\ I_j y_j &= I_{j^1} y_{j^1} + I_{j^2} y_{j^2} \end{aligned} \tag{13}$$

where the subscripts j, j^1 and j^2 stand for the randomly chosen emitter and the two new emitters after splitting. There are still degrees of freedom in how to do this, so we generate a random vector $\vec{u}$ whose elements are used to calculate the specific parameters. Select u_1 from Beta(1,1), and u_2 and u_3 from $N(0, \sigma^2_{\text{PSF}})$. Solving for the new parameters

$$\begin{aligned} I_{j^1} &= I_j u_1 \\ I_{j^2} &= I_j (1 - u_1) \\ x_{j^1} &= x_j + u_2 \\ y_{j^1} &= y_j + u_3 \\ x_{j^2} &= x_j - \frac{u_1 u_2}{1 - u_1} \\ y_{j^2} &= y_j - \frac{u_1 u_3}{1 - u_1} \end{aligned} \tag{14}$$

The acceptance probability for a cross-dimensional jump in RJMCMC is [13,14]

$$P = \min\{1, A\} \tag{15}$$

where

$$A = \frac{p(\theta'|D) r_m(\theta')}{p(\theta|D) r_m(\theta) q(u)} \left| \frac{\partial(\theta')}{\partial(\theta, u)} \right| \tag{16}$$

$p(\theta'|D)$ is the posterior; D is data; θ is the current parameter set; θ' means either the deterministic function that calculates the new parameters $\theta'(\theta, u)$ or the result of that calculation; $r_m(\theta)$ is the probability of choosing the move type m when in state θ; and $q(u)$ is the density function of u. The term on the right is the Jacobian for transforming variables from (θ, u) to θ'. The ratio $\frac{p(\theta'|D)}{p(\theta|D)}$ is determined from the details given in Section 1.1, and for split can be written as

$$\frac{p(\theta'|D)}{p(\theta|D)} = \left(\prod_k \frac{L_k(D|\theta')}{L_k(D|\theta)} \right) \left(\frac{\prod_{n=1}^{N+1} P(I'_n)}{\prod_{n=1}^{N} P(I_n)} \right) \frac{1}{(W+2)^2} \left(\frac{\rho W^2}{N+1} \right) \tag{17}$$

where k counts the pixels and n indexes the emitters. In addition, L_k is the likelihood given in (3), the second parenthesized expression gives the ratio of the intensity priors, the third factor is the ratio of the position priors where W is the width of the given frame in pixels, and the last parenthesized expression is the ratio of the priors for the proposed and current number of emitters. The number of emitters has a Poisson distribution with mean value ρW^2, where ρ is the density of emitters and N is the current number of emitters. The ratio $\frac{r_m(x')}{r_m(x)}$ is $\frac{P_M}{P_S}$, because the probabilities for proposing a split and merge are given by P_S and P_M. $q(u)$ is the probability density of u, that is, $p(u_1)p(u_2)p(u_3)$, where these probabilities are calculated from the PDFs used to generate $\vec{u}$. The term $\left| \frac{\partial(\theta')}{\partial(\theta,u)} \right|$ is written out here. Only the values of θ' related to the split have been affected, so all other elements of this $3(N+1) \times 3(N+1)$ matrix are diagonal except those relating to the 6 changed elements of θ'. Putting these elements in the order listed above, $\theta' = (..., I_{j^1}, x_{j^1}, x_{j^2}, I_{j^2}, y_{j^1}, y_{j^2})$ and $(\theta, u) = (..., I_j, u_1, x_j, u_2, y_j, u_3)$. The Jacobian matrix is

$$\frac{\partial(\theta')}{\partial(\theta, u)} = \begin{pmatrix} 1-u_1 & 0 & 0 & -I_j & 0 & 0 \\ u_1 & 0 & 0 & I_j & 0 & 0 \\ 0 & 1 & 0 & \frac{-u_2}{1-u_1} + \frac{-u_2 u_1}{(1-u_1)^2} & \frac{-u_1}{1-u_1} & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & \frac{-u_3}{1-u_1} + \frac{-u_3 u_1}{(1-u_1)^2} & 0 & \frac{-u_1}{1-u_1} \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

and the determinant is therefore

$$\left|\frac{\partial(\theta')}{\partial(\theta,u)}\right| = \frac{I_j}{(1-u_1)^2} \tag{19}$$

The complete expression for A is

$$\begin{aligned} A \quad &= \left(\prod_k \frac{e^{-\lambda_k(N+1)}\lambda_k(N+1)^{D_k}}{e^{-\lambda_k(N)}\lambda_k(N)^{D_k}}\right) \frac{1}{(W+2\sigma)^2} \frac{\prod_{n=1}^{N+1} P(I'_n)}{\prod_{n=1}^{N} P(I_n)} \left(\frac{\rho W^2}{N+1}\right) \\ &\times \frac{P_M}{P_S}(\text{Beta}(u_1,1,1)N(u_2,0,\sigma^2_{\text{PSF}})N(u_3,0,\sigma^2_{\text{PSF}}))^{-1}\frac{I_j}{(1-u_1)^2} \end{aligned} \tag{20}$$

Merge

After proposing a merge, we randomly pick an emitter to combine with one of its neighbors. If the neighboring emitter of the picked emitter is further than $2\sigma_{\text{PSF}}$, we reject the proposed merge, otherwise we calculate the acceptance probability given by

$$P = \min\{1, A^{-1}\} \tag{21}$$

where u_1, u_2, u_3 are deterministically calculated from (14) and A is given by (20).

1.2.4 Generalized Split and Merge

Split and merge allow the move from one emitter to two emitters and vice versa. In more dense data sets, a generalized version of those jumps proved to be pivotal, where they permit moving from N emitters to $N+1$ emitters and vice versa. Therefore, we can explore the probability of N nearby emitters to be $N+1$ emitters and the opposite. The group of N nearby emitters are selected by picking an emitter at random and then finding the emitters that are closer than $2\sigma_{\text{PSF}}$ to it.

Generalized Split

A random group of N adjacent emitters is picked. A new emitter is then formed by taking a few photons from each of the picked emitters. This takes us to a model with one more emitter. The parameters of the new emitters are calculated so that the number of photons and the center of mass are conserved.

$$\begin{aligned} \sum_{i=1}^{N} I_i &= \sum_{j=1}^{N+1} I'_j \\ \sum_{i=1}^{N} I_i x_i &= \sum_{j=1}^{N+1} I'_j x'_j \\ \sum_{i=1}^{N} I_i y_i &= \sum_{j=1}^{N+1} I'_j y'_j \end{aligned} \tag{22}$$

where prime indicates the parameters of the proposed emitters. The primed parameters are $3(N+1)$ unknowns. The three equations given in (22) can be used to reduce the degrees of freedom, however, this still leaves $3N$ degrees of freedom for the primed parameters. To eliminate these freedoms, we propose the following equations

$$\begin{aligned} I'_i &= (1-u_1)I_i \\ x'_i &= \frac{x_i - u_1\left(\frac{1}{N}\sum_{i=1}^{N} x_i + u_2\right)}{1-u_1} \\ y'_i &= \frac{y_i - u_1\left(\frac{1}{N}\sum_{i=1}^{N} y_i + u_3\right)}{1-u_1} \end{aligned} \tag{23}$$

where $i = 1, ..., N$. The first equation above says that the percentage of photons taken from each existing emitter is equal. The second and third equations in (23) are derived by assuming that the new emitter lies in the vicinity of the center of mass of the existing emitters. However, this still leaves 3 degrees of freedom to be set. The random variables

$u_1 \sim \text{beta}, u_2 \sim \text{normal}, u_3 \sim \text{normal}$ are then introduced to satisfy them. Using these random variables and (22,23), we can deterministically solve for the parameters of the new emitter

$$\begin{aligned} I'_{N+1} &= u_1 \sum_{i=1}^{N} I_i \\ x'_{N+1} &= \frac{1}{N}\sum_{i=1}^{N} x_i + u_2 \\ y'_{N+1} &= \frac{1}{N}\sum_{i=1}^{N} y_i + u_3 \end{aligned} \tag{24}$$

The acceptance probability for generalized split is given by

$$P = \min\{1, A\} \tag{25}$$

where A is given by (16) and the Jacobian can be calculated as follows

$$\frac{\partial(\theta')}{\partial(\theta,u)} = \begin{bmatrix} \frac{\partial I'_1}{\partial I_1} & \cdots & \frac{\partial I'_1}{\partial I_N} & \frac{\partial I'_1}{\partial u_1} & \frac{\partial I'_1}{\partial x_1} & \cdots & \frac{\partial I'_1}{\partial x_N} & \frac{\partial I'_1}{\partial u_2} & \frac{\partial I'_1}{\partial y_1} & \cdots & \frac{\partial I'_1}{\partial y_N} & \frac{\partial I'_1}{\partial u_3} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial I'_N}{\partial I_1} & \cdots & \frac{\partial I'_N}{\partial I_N} & \frac{\partial I'_N}{\partial u_1} & \frac{\partial I'_N}{\partial x_1} & \cdots & \frac{\partial I'_N}{\partial x_N} & \frac{\partial I'_N}{\partial u_2} & \frac{\partial I'_N}{\partial y_1} & \cdots & \frac{\partial I'_N}{\partial y_N} & \frac{\partial I'_N}{\partial u_3} \\ \frac{\partial I'_{N+1}}{\partial I_1} & \cdots & \frac{\partial I'_{N+1}}{\partial I_N} & \frac{\partial I'_{N+1}}{\partial u_1} & \frac{\partial I'_{N+1}}{\partial x_1} & \cdots & \frac{\partial I'_{N+1}}{\partial x_N} & \frac{\partial I'_{N+1}}{\partial u_2} & \frac{\partial I'_{N+1}}{\partial y_1} & \cdots & \frac{\partial I'_{N+1}}{\partial y_N} & \frac{\partial I'_{N+1}}{\partial u_3} \\ \frac{\partial x'_1}{\partial I_1} & \cdots & \frac{\partial x'_1}{\partial I_N} & \frac{\partial x'_1}{\partial u_1} & \frac{\partial x'_1}{\partial x_1} & \cdots & \frac{\partial x'_1}{\partial x_N} & \frac{\partial x'_1}{\partial u_2} & \frac{\partial x'_1}{\partial y_1} & \cdots & \frac{\partial x'_1}{\partial y_N} & \frac{\partial x'_1}{\partial u_3} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial x'_N}{\partial I_1} & \cdots & \frac{\partial x'_N}{\partial I_N} & \frac{\partial x'_N}{\partial u_1} & \frac{\partial x'_N}{\partial x_1} & \cdots & \frac{\partial x'_N}{\partial x_N} & \frac{\partial x'_N}{\partial u_2} & \frac{\partial x'_N}{\partial y_1} & \cdots & \frac{\partial x'_N}{\partial y_N} & \frac{\partial x'_N}{\partial u_3} \\ \frac{\partial x'_{N+1}}{\partial I_1} & \cdots & \frac{\partial x'_{N+1}}{\partial I_N} & \frac{\partial x'_{N+1}}{\partial u_1} & \frac{\partial x'_{N+1}}{\partial x_1} & \cdots & \frac{\partial x'_{N+1}}{\partial x_N} & \frac{\partial x'_{N+1}}{\partial u_2} & \frac{\partial x'_{N+1}}{\partial y_1} & \cdots & \frac{\partial x'_{N+1}}{\partial y_N} & \frac{\partial x'_{N+1}}{\partial u_3} \\ \frac{\partial y'_1}{\partial I_1} & \cdots & \frac{\partial y'_1}{\partial I_N} & \frac{\partial y'_1}{\partial u_1} & \frac{\partial y'_1}{\partial x_1} & \cdots & \frac{\partial y'_1}{\partial x_N} & \frac{\partial y'_1}{\partial u_2} & \frac{\partial y'_1}{\partial y_1} & \cdots & \frac{\partial y'_1}{\partial y_N} & \frac{\partial y'_1}{\partial u_3} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial y'_N}{\partial I_1} & \cdots & \frac{\partial y'_N}{\partial I_N} & \frac{\partial y'_N}{\partial u_1} & \frac{\partial y'_N}{\partial x_1} & \cdots & \frac{\partial y'_N}{\partial x_N} & \frac{\partial y'_N}{\partial u_2} & \frac{\partial y'_N}{\partial y_1} & \cdots & \frac{\partial y'_N}{\partial y_N} & \frac{\partial y'_N}{\partial u_3} \\ \frac{\partial y'_{N+1}}{\partial I_1} & \cdots & \frac{\partial y'_{N+1}}{\partial I_N} & \frac{\partial y'_{N+1}}{\partial u_1} & \frac{\partial y'_{N+1}}{\partial x_1} & \cdots & \frac{\partial y'_{N+1}}{\partial x_N} & \frac{\partial y'_{N+1}}{\partial u_2} & \frac{\partial y'_{N+1}}{\partial y_1} & \cdots & \frac{\partial y'_{N+1}}{\partial y_N} & \frac{\partial y'_{N+1}}{\partial u_3} \end{bmatrix} \tag{26}$$

$$\left|\frac{\partial(\theta')}{\partial(\theta,u)}\right| = \frac{\sum_{i=1}^{N} I_i}{(1-u_1)^{N+1}} \tag{27}$$

Note that the above equations can be reduced to the equations in the split section when $N = 1$.

Generalized Merge

An emitter is chosen at random and then we pick N random emitters closer that $2\sigma_{\text{PSF}}$. The picked emitter (emitter $N+1$) is annihilated and its photons are distributed among the set of the neighboring emitters. Note that we jump from $N+1$ emitters to N emitters. The emitters' parameters and the random u's may be calculated in a deterministic way

$$\begin{aligned} u_1 &= \frac{I_{N+1}}{\sum_{i=1}^{N+1} I_i} \\ I'_i &= \frac{I_i}{1-u_1} \\ x'_i &= u_1 x_{N+1} + (1-u_1)x_i \\ y'_i &= u_1 y_{N+1} + (1-u_1)y_i \\ u_2 &= x_{N+1} - \frac{1}{N}\sum_{i=1}^{N} x'_i \\ u_3 &= y_{N+1} - \frac{1}{N}\sum_{i=1}^{N} y'_i \end{aligned} \tag{28}$$

where parameters with the prime represent the parameters associated with the proposed model (one less emitter). The acceptance probability for this jump is given by

$$P = \min\{1, A^{-1}\} \tag{29}$$

where A is described in the previous section.

1.2.5 Birth and Death

Birth and death are complementary reversible processes. They allow addition and subtraction of emitters globally. Birth inspects the data to find the spots where an emitter might exist. Death removes one of the existing emitters at random to examine the probability of the model with one less emitter.

Birth

To find a suitable spot for the new emitter, we subtract the current model from the data to obtain the residuum image. The residuum image can be interpreted as a probability distribution where pixels with higher values indicate higher probability for a missing emitter. We pick a location at random from this probability distribution for the new emitter. An alternative approach would be proposing a random location for the new emitter across the image where all the pixels have the same probability; however most of the proposed births in this procedure would be rejected. The residuum image scheme facilitates spotting the missing emitters and hence the chain converges faster. The intensity of the new emitter is chosen from the intensity prior at random and therefore $q(u)$ in (16) contains the intensity prior, which cancels the intensity prior from the posterior, and so the intensity prior is not involved in the calculations. The acceptance probability is given by (15), where

$$A = \left(\prod_k \frac{e^{-\lambda_k(N+1)} \lambda_k (N+1)^{D_k}}{e^{-\lambda_k(N)} \lambda_k (N)^{D_k}} \right) \frac{1}{P_{\text{res}}(W+2\sigma)^2} \left(\frac{\rho W^2}{N+1} \right) \frac{P_D}{P_B} \tag{30}$$

Here, the first factor is the likelihood ratio, the second factor is the location prior ratio with P_{res} standing for the residuum image distribution, the third factor is the prior ratio for the number of emitters, and the last factor represents the ratio of the probabilities to propose a death or a birth.

Death

Proposing a death, we pick one of the existing emitters at random and remove it from the model. This is the opposite of birth and its acceptance probability is given by

$$P = \min\{1, A^{-1}\} \tag{31}$$

where

$$A = \left(\prod_k \frac{e^{-\lambda_k(N+1)} \lambda_k (N+1)^{D_k}}{e^{-\lambda_k(N)} \lambda_k (N)^{D_k}} \right) \left(\frac{\rho W^2}{N+1} \right) \frac{P_D}{P_B} \tag{32}$$

The ratio of the position priors is missing here because they exert no preference in picking an emitter.

1.2.6 Conversion

We chose an emitter at random and if it is part of the signal, then we will propose it to be a background emitter (Fig. 1f), where the conversion probability is given by

$$P = \min\{1, A\} \tag{33}$$

and

$$A = \frac{P_{\text{Bg}}(I)}{P_{\text{Signal}}(I)} \tag{34}$$

$P_{\text{Signal}}(I)$ and $P_{\text{Bg}}(I)$ represent the signal and background priors (Supplementary Fig. 6a). If the picked emitter is part of the background, then we propose it to be a signal emitter (Fig. 1f) and the acceptance probability can be obtained as follows:

$$P = \min\{1, A^{-1}\} \tag{35}$$

Note that, the conversion jump is the only jump type where we use both the signal and background intensity prior value of an emitter in the acceptance probability calculation. In the other jumps, only one of the intensity priors will be used based on the current state (signal or background) of the emitter.

1.3 Chain setup and generation

The above jumps are proposed inside an iterative loop for a user specified number of iterations. Larger regions require more iterations. The chain is built as follows. First, it is initialized to an empty state where there are no emitters and the offset background initial value is taken as the mode of the prior. Next, we propose random states via the available jumps, which are embedded in RJMCMC, and allow exploration of all possible states. Calculating the acceptance probability, which was described above, we either accept or decline the jump. If the jump is accepted, we take it as the next state, otherwise we take the current state as the next one. The beginning part of the chain before its convergence is called the burn-in and will not appear in the final reconstruction. The chain makes many random jumps and attempts to spot the emitters in the given data during the burn-in. The post-burn-in part of the chain is returned by the code and is utilized to generate the final reconstruction.

1.4 Posterior and MAPN outputs

In the post-burn-in chain, the emitters have been detected and the chain has settled to a near equilibrium. There are only small deviations from the real values of the parameters, reflecting the uncertainty (Fig. 1e). For each emitter, a blob is returned, where the width is a measure of the uncertainties in the estimated parameters. RJMCMC returns the chain and the posterior image. The posterior image is built by placing a dot on the found emitters' locations in each jump. The posterior images from each subregion are put together to obtain the final chain reconstruction. This chain contains the number of emitters, their positions, intensities, offset background and slopes along the X and Y axes, jump type, likelihood ratio and posterior ratio for each attempted jump. Furthermore, the most repeated model in the chain (MAPN from RJMCMC) is extracted, and the k-means algorithm is used to find the emitter locations and accuracies. This step is necessary because the emitters associated with a given model will have slightly different positions and are ordered arbitrarily in different states. The results are used to initialize an MCMC chain which is employed to calculate the emitter parameters and their uncertainties, which is MAPN coordinates (see Supplementary Note 3). The locations and uncertainties returned by the MCMC algorithm are then used to generate the MAPN reconstruction from MCMC.

2 Supplementary Note 2: Noise characterization

Two main types of noise in super-resolution data are read-out noise and shot noise. Shot noise comes from the particle nature of photons and can be modeled by a Poisson process. Read-out noise comes from the electronics of the camera and has a Gaussian distribution with the mean value zero. In other words, it is the fluctuations of the detector when there is no signal. For an EMCCD camera, read-out noise can be ignored (3), however, it has a higher value in the data acquired using a sCMOS camera and cannot be neglected. The entire noise can be modeled as the convolution of the shot noise (Poisson distribution) and read-out noise (Gaussian distribution) [25]

$$P_i(D_i) = N \sum_{q=0}^{\infty} \frac{1}{q!} e^{-u_i} u_i^q \frac{1}{\sqrt{2\pi \mathrm{var}_i}} e^{-\frac{(D-qg_i-o_i)^2}{2\mathrm{var}_i}} \tag{36}$$

where N, D, u, g, o and var are, respectively, the normalization constant, the number of counted photons (Data), the mean photon count, gain, offset and variance with i counting pixels. In the case of EMCCD, the ratio $\frac{\mathrm{var}_i}{g_i^2}$ is small and equation (36) approximates to

$$P(D_i) \approx N' e^{-u_i} u_i^{(D_i-o)/g} \tag{37}$$

where N' is the normalization constant and gain and offset are the same across the EMCCD camera. We used this approximation to obtain the likelihood (3). In the case of sCMOS camera, the above approximation is not valid and one should use the analytical expression which can be computationally expensive. An elegant solution to this problem is given in [25], where an additional source of signal is added to the problem with the mean intensity of $\frac{\mathrm{var_i}}{g_i^2}$. Using the fact that the Gaussian distribution asymptotically goes to a Poisson distribution, the read-out noise can also be modeled by a Poisson process. Lastly, the likelihood for the data taken by an sCMOS camera is as follows:

$$L_i(D|\theta) \approx \frac{(\lambda_i(N) + \mathrm{var}_i/g_i)^{D_i} e^{-\lambda_i(N)+\mathrm{var}_i/g_i}}{D_i!} \tag{38}$$

Therefore, adding the term var_i/g_i to the mean number of the photons in the likelihood will take care of the read-out noise in sCMOS cameras.

3 Supplementary Note 3: MATLAB BAMF Library

The RJMCMC algorithm is implemented in C++ as a mex-function which can be called from inside MATLAB. We developed a MATLAB class consisting of several methods and tools along with a user friendly interface (GUI) to facilitate using the algorithm. In this section, a detailed description of the data flow and the methods used at different points of the process are provided. At the end, some of the diagnosis and visualization tools are depicted. Some of the visualization methods can be used to get more insight into RJMCMC and its jumping processes. We place empty parentheses at the end of the method names to make it clear that we are referring to a method.

The main class is called RJ. The input parameters required by the RJMCMC algorithm, such as PSF size, jump sizes, probabilities of proposing different jumps, zoom factor, numerical priors and number of jumps, are all stored in a structure called RJStruct. To find the intensity and offset background priors, we use a fast single-emitter code which also needs a few input parameters, like mean number of photons, minimum number of photons and minimum p-value, which are stored in a structure called SMAStruct. The post-processing of the data consists of thresholding, frame connection and drift correction. The required parameters for these post-processing steps are stored in structures called Threshold and FrameConnect. Some other inputs like the gain and offset are properties of the RJ-class too. Moreover, some other properties give the user the option to use certain tools or return more outputs, usually for diagnosis, such as using the MATLAB parallel computing toolbox, using a numerical PSF, saving the chain, etc. The structure, ClustInfo, contains the found positions, intensities and backgrounds. JumpStat stores the statistics of all the accepted jumps. PChain and Chain are MATLAB cell arrays containing the proposed and accepted chains. Note that the size of the chains can be tremendously large for real data sets, so the chain should only be saved for small diagnostic simulations.

The user can work with the GUI or write his or her own script to call different functions from the class. The gui displays eight panels. The DATA-panel has buttons to load the raw data and drifts. The RJStruct-panel tabulates default values for the RJMCMC parameters, which can be altered by the user. The P_Burnin-panel and P_Trial-panel let the user change the probabilities of proposing different jumps in the burn-in part of the chain and the trial portion afterwards. The SMA-panel displays the parameters required by the single-emitter code. Some input parameters of the code are included in the Parameters-panel. The Threshold-panel and FrameConnect-panel contain the post-processing parameters. The post-processing can be disabled by unchecking the boxes next to their panels.

processData()

By pressing the Run-button in the gui, the method processData() is called inside a for-loop which loads a data set in every iteration and sends it to the analyzeData() method. At the end, it saves all the output structures and the posterior reconstruction.

analyzeData()

analyzeData() is the main method in the RJ-class from which all the other analysis methods are called. The first function called inside this method is gainCorrection(), which corrects for the gain and offset noise. Next, findPriors() is called to find the intensity priors for the signal and background emitters and also the prior for the offset background. Then, findPSF_SMA() calculates the numerical PSF using the raw data. After that, the sequence of the frames are divided into a number of subsequences equal to the number of parallel workers available on the machine. A parfor-loop is then called and the workers process each subsequence of the frames individually. The user has the option to use a simple for-loop as well. The method makeSubRegions() is called inside the for-loop, dividing the frames into subregions for speed purposes. Next, the subregions are sent to analyzeROIs() to be processed. The subregions overlap to remove artifacts at their edges, so the found emitters in the overlapping area need to be removed. This task is done by the removeOverlapping() method. Then, stitchROIs() stitches together all the subregions in a frame, and then assembles the frames to get the final results for a subsequence. Finally, when the parfor-loop is done, the results are retrieved from the different workers and collected together.

findPriors()

findPriors() runs a fast single-emitter algorithm [17] to find the emitters' intensities and the offset background. It then fits a smoothed kernel density estimator to the found intensities. Because the single-emitter algorithm sometimes fits two or more very close emitters by a single emitter with higher intensity, we might get smaller peaks at higher intensities than the main peak, which is unrealistic. To eliminate those following unrealistic peaks in the returned intensity prior, we calculate the gradient of the prior and use that to remove the spurious smaller peaks described. The resulting distribution is returned as the numerical prior for signal intensities. The background emitters' intensity prior is taken as an exponential distribution with an average equal to the mean of the found intensities divided by a user provided parameter (BgPriorRatio). Moreover, this method returns a gamma distribution fit to the found offset backgrounds.

findPSF_SMA()

findPSF_SMA() calculates the numerical PSF for each subregion. This is important because the PSF might be a function of the position. It makes use of the single-emitter code to fit a portion of the data after which the outputs are filtered to find isolated emitters. Next, the PSF is created by shifting and averaging over more than 100 high signal, found single emitters. A 4× sub-sampled PSF is created by padding the Fourier transform. findPSF_SMA() returns a MATLAB cell containing the numerical PSF model for each subregion.

analyzeROIs()

The subregions and found priors are sent to analyzeROIs() for processing. Inside this function, each subregion is analyzed individually in every iteration of the for-loop described in analyzeData(). First, the rjmcmc() method is called to implement the RJMCMC algorithm for a subregion. It returns the chain which is the input to the findMap() method. Next, findMap() finds the most likely model in the chain and returns the parameters associated with that model. This model is used next to initialize an MCMC chain inside the mcmc() method. MCMC only tries within model moves to find more accurate positions and intensities of the found emitters. Finally, the results from mcmc() plus the posterior image of the subregion are returned as outputs of analyzeROIs().

findMap()

findMap() takes the chain from the rjmcmc() method and uses the histogram of the number of the found emitters to find the most repeated model. It then extracts the states of the chain corresponding to that model and uses the k-means algorithm to find the positions, intensities and their associated errors, which are used later to initialize the MCMC algorithm. The k-means step is necessary because adding and subtracting many emitters from the chain makes it impossible to recognize a particular emitter in different states of the chain. The assembleResults() method is called to put together the results from all the different data sets and performs the post-processing. In the following, a short description of some of the visualization and diagnosis methods are given.

chainAnimation()

chainAnimation() takes the chain and the subregion number plus the true positions (for simulated data). When the user call this function, a gui pops up, which can then be used to look at different states of the accepted chain and the proposed chain, the overlay of the data with the model for the accepted and proposed chains, the emitters' parameters for both chains and also a plot of the emitters' positions. It is a very useful tool and can be employed for either diagnostic purposes or for getting a better grasp of the RJMCMC algorithm.

makeModel()

makeModel() is a method for diagnosis and also for getting better insight into the RJMCMC algorithm. It takes a subregion and its corresponding chain and makes a 3D stack of images, where each slice of the stack is an overlay of the data with a model from a different state of the chain. This is a useful tool to look at the overlay of models produced from different states of the chain with experimental data, where the true positions are not known and the user cannot use the chainAnimation() method.

4 Supplementary Note 4: Numerical PSF interpolation

In the case of bright emitters, such as that from DNA-PAINT data, the 2D Gaussian approximation or theoretical models of the microscope PSF are not adequate for multiple-emitter fitting [26]. In these situations, one can use experimentally acquired PSFs [26–31]. Our approach to estimating the numerical PSF was described in the previous section (findPSF_SMA()). The N subsampled PSF is stored in a $4D$ array where the third dimension contains N samples along the X-axis and the fourth dimension contains N samples along the Y-axis. By linear interpolation along the X and Y axes and scaling by the number of photons, one obtains the PSF with its center at a desired location. This procedure is repeated for all the existing emitters to generate the model.

References

[24] Brooks, S., Gelman, A., Jones, G. L. & Meng, X.-L. Handbook of Markov Chain Monte Carlo. (CRC Press, 2011).

[25] Huang, F. *et al.* Video-rate nanoscopy using sCMOS camera-specific single-molecule localization algorithms. *Nat Methods* **10**, 653-658 (2013).

[26] Liu, S. Kromann, E. B., Krueger, W. D., Bewersdorf, J. & Lidke, K. A., Three dimensional single molecule localization using a phase retrieved pupil function, *Opt. Express* **21** 29462-29487 (2013).

[27] Quirin, S., Pavani, S. R. P. & Piestun, R., Optimal 3D single-molecule localization for superresolution microscopy with aberrations and engineered point spread functions, *Proc. Natl. Acad. Sci. USA* **109**, 675-679 (2012).

[28] Baddeley, D., Cannell, M. B. & Soeller, C., Three-Dimensional Sub-100 nm Super-Resolution Imaging of Biological Samples Using a Phase Ramp in the Objective Pupil, *Nano Res.* **4**, 589-598 (2011).

[29] Tahmasbi, A., Ward, E. S & Ober, R. J., Determination of localization accuracy based on experimentally acquired image sets: applications to single molecule microscopy, *Opt. Express* **23**, 7630-7652 (2015).

[30] Babcock, H. P., & Zhuang, X., Analyzing Single Molecule Localization Microscopy Data Using Cubic Splines, *Sci. Rep.* **7**, 552 (2017).

[31] Li, Y., Mund, M., Hoess, P., Deschamps, J., Matti, U., Nijmeijer, B., Sabinina, V. J., Ellenberg, J., Schoen, I., & Ries, J., Real-time 3D singlemolecule localization using experimental point spread functions, *Nat Methods* **15**, 367-369 (2018).