Software Documentation

As described in **Supplementary Figure 2**, the data analysis workflow was divided into subsections, for which we provide required software as well as a number of test data sets. Please also look at the section **required dependencies** at the end of **Supplementary Note 4.**

1) **Channel registration 1**

Required files (folder channel registration):

Calculate_AffineT_from_Beads.m
Apply_Affine_Trans.m

After image acquisition and single molecule localization, the resulting localization maps need to be corrected for differences and aberrations (rotation, magnification) of the emission path between both detection channels. To this end, at the beginning of each imaging day, we acquired two wide field images of FluoroSpheres visible in both emission channels (*Channel_registration/Ch642_1 and Ch750_1*).

a) These two tiff files can be loaded into *Calculate_AffineT_from_Beads.m* to determine the affine transformation required to transform the localizations in the DL755 emission channel onto those in the A647 emission channel.

Input: 2 Tiff files
Output: Affine transformation (*.mat file*)

b) Next, we apply the affine transformation on the DL755 channel of the dataset. Please use the file *Apply_Affine_Trans.m* and follow the instructions therein.

Input: Affine transformation (.mat file), DL755 localization file
Output: affine corrected DL755 localization file (**affine_applied.mat*)

2) **Lateral drift correction**

Required files:

Drift_correction.ipynb

Sample drift in directions parallel to the plane of the coverslip results in a time-dependent bias in the localization datasets. It is corrected by first inferring the drift trajectory—i.e. the coverslip's lateral displacement with time—from gold fiducials that are present on the coverslip. The values of these trajectories—one for each lateral coordinate—at the corresponding time-point are subtracted from each localization to remove the bias.

The drift correction code use our custom analysis library called B-Store[16].

Input: A647 localization file, DL755 localization file after affine correction (**affine_applied.csv*)

a) Open the Jupyter notebook (*Drift_correction.ipynb*) and follow the directions, executing the cells by first clicking on them and clicking the *Run cell button* or hitting *Shift + Enter*. In the cell where a dataset is loaded, change the string stored in *pathToData*'s Path object to point towards the dataset to be corrected.
b) After executing the cell that calls the FiducialDriftCorrection processor, a window will appear showing a 2D histogram in which the color encodes the number of localizations within a bin. Bins containing a fiducial should contain a number of localizations that is approximately equal to the number of frames that comprised the dataset. With the mouse, drag a small square around one such bin and press the *Spacebar* to add it to the list of fiducials. Continue this procedure until 3-5 fiducials have been selected. Close the window when you are finished selecting fiducials.
c) After executing the cell containing the *plotFiducials()* method call, a number of plots will appear. Each plot shows the x- and y-coordinates of the localizations from a single fiducial as a function of time. The smooth solid line is the final drift trajectory that is computed by averaging over all sets of fiducials.
d) If the average trajectory does not visually correspond well to the individual trajectories, you can try to modify the fit doing one or more of the following.
    a. Specify a subset of the fiducials to use by adding their integer identifiers to the *useTrajectories* list.
    b. Change the size of the Gaussian smoothing window *smoothingWindowSize* or the Gaussian kernel's width *smoothingFilterSize*. Each trajectory is smoothed by this moving average and fit with a smoothing spline. The individual splines are then averaged together to produce the final trajectory.
    c. Change frame number at which the individual x- and y-trajectories are made to cross the horizontal axis by changing the *zeroFrame* parameter.
    d. Remove outlier localizations by setting the maxRadius parameter. Localizations that lie farther than this value from the localizations' center of mass are not included in the fit are appear as grey data points in the trajectory plots.

Rerun the cells that apply the drift correction to the dataset and plots the fiducial trajectories each time the trajectory fitting parameters are changed until a satisfactory curve is obtained. Then, run the cell that saves the corrected (_DC) dataset at the end.

More help may be found in the example notebook (https://github.com/kmdouglass/bstore/blob/master/examples/Fiducial-based%20Drift%20Correction.ipynb) and the B-Store documentation (http://b-store.readthedocs.io/en/latest/).

Output: drift corrected DL755 localization file (**affine_applied_DC.mat*), drift corrected A647 localization file (**affine_applied_DC.mat*), fiducial trajectories (**Localizations_affineApplied_Fid.csv*), fiducial averages (**Localizations_affineApplied_AVG.csv*)

3) Channel Registration 2

Required files (folder channel registration):

Calculate_RigidT_from_Fiducials.m

Apply_Rigid_Trans.m

To finish the correction of the DL755 localization file, we calculate and match the centroid positions of fiducial beads localized in both channels. To this end, we load the fiducial trajectories from step 2, find the centroids in both channels and calculate the linear shift to match their position.

Input: DL755 localization file (**affine_applied_DC.mat*), fiducial trajectories (**Localizations_affineApplied_Fid.csv*), fiducial averages (**Localizations_affineApplied_AVG.csv*)

Open *Calculate_RigidT_from_Fiducials.m* and load the required files as indicated in the first section and follow the code and the instructions therein.

Output: corrected DL755 localization file (**affine_applied_DC_corrected.mat*), estimate of the residual registration error

4) Particle Segmentation

Required file:

particle_segmentation_2C.m

At this step, the individual particles are extracted from the localization file. This step serves multiple proposes: 1) it distributes the data into particles, which facilitates all further processing and 2) it critically serves as a filter to select only densely- and in both channels labelled particles.

Input: DL755 localization file (**affine_applied_DC_corrected.mat*), DL755 wide field image
A647 localization file (**Localizations_DC.mat*), A647 wide field image

After opening *particle_segmentation.m* and loading the data files as indicated, the particles from both localization files are segmented by generating a thresholded mask from one of the respective wide field images. We typically use the channel with the highest contrast. While OTSU segmentation gives a satisfying result, we also implemented manual thresholding. The segmented area is then transformed into a rectangle and expanded to adjust for small differences between the wide field image and the localization cluster. Finally, the localization within each segment are stored in a new structure.

Output: Structure array of all extracted centrioles in 2 channels (**extracted_Particles.mat*)

5) Particle filtering and image generation

Required file:

particle_filter.m

Input: Structure array of all extracted centrioles in 2 channels (**extracted_Particles.mat*)

The filtering is critical for the generation of the Scipion input and hence the outcome of the 3D reconstruction. First load the extracted particles and define a new name for the filtered dataset (eg. *extracted_Particles_filtered.mat*). Then follow the instructions given in the file.

Initially the particle set is filtered according to the length of the respective localization file. This step removes large areas of clustered particles that cannot be separated during later filtering steps. The localizations are then filtered according to minFrame, MinPhotons, MaxUncertainty. The localizations from both datasets are then combined to allow separation of dual-labeled particles using density-based filtering (Supplementary Figure 3). This step is crucial, as it increases the number of single particles and removes unspecific localizations. We then calculate the resolution (using Fourier ring correlation) as well as 12 shape descriptors for both particle datasets. After a visualization step showing diverse correlations between these descriptors, the user can define a final filter to select only the best (e.g. most densely labelled) particles. We then define a unifying bounding box to put all the particles into the same space, append them to the localization list and render each particle with a given pixel size (e.g 10 nm). The rendered image library is then stitched into a single montage overview for both imaged channels (Supplementary Figure 4).

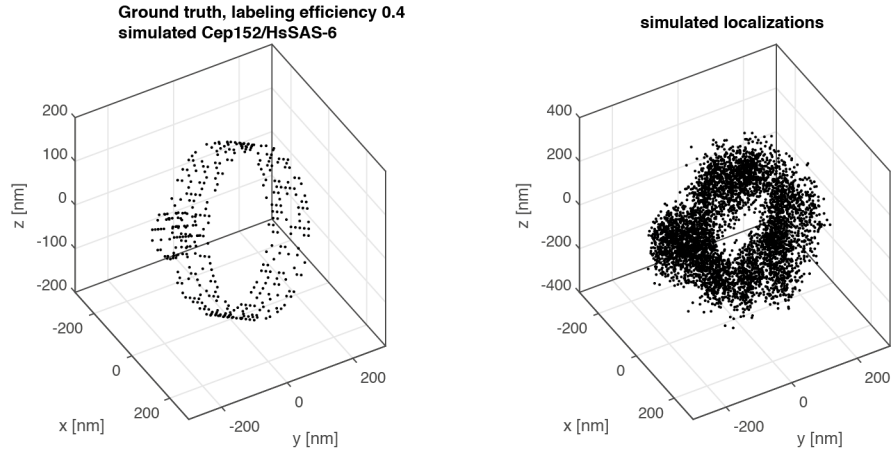
Particle Simulator

Required files:

simulate_particles_fromGT.m and called data/functions in folder *smlm_simulator*

Input: a ground truth dataset (x, y, z coordinates in nm) in MATLAB pointCloud format

We provide a simple SMLM particle simulator that uses a ground truth structure, selects a number of labelled sites (according to the selected labelling efficiency) and populates each site (i.e. each fluorophore position) with a scatter of localizations. See example below for the simulation shown in Supplementary Figure 5



Output: Structure with n cells, where each cell contains a single simulated particle with x (nm), y(nm), z(nm), photons, frame.

After finishing the simulation, the script renders each particle and generates a stitched montage that can directly be used as input for Scipion.

2D Averaging

We implemented a 2D particle averaging routine that uses image image cross-correlation to find the optimal overlap between a pair of images. This routine relies on and is an extension of the previously published efficient subpixel image registration by cross-correlation[5], which was also recently applied to study protein symmetries within mouse basal bodies[6]. We use 2D averaging to 1) find the best overlay between different centriolar side view projections (as used in Fig. 1c to determine Δz) as well as 2) unravel symmetric properties in top view projections of imaged centriolar proteins (as in Supplementary Figure 8).

Input: One or two particle image datasets (*Particles_Ch1_*.tif* and *Particle_Ch2_.tif*), an initial reference image for the first averaging iteration (i.e. the sum of all input particles in one channel)

The script aligns only the reference channel and the applies the identified translation and rotation to the second dataset.

Output: One or two matched images

Provided Test Datasets

We provide the following experimental and simulated datasets via Zenodo (https://doi.org/10.5281/zenodo.1127010) to test the functionality of our workflow. The dataset includes one fully processed field of view that can be used according to the file naming conventions introduced above to test the individual steps of the workflow.

1. Two-color bead images
2. Two-color localizations from one field of view of imaged centrioles labelled against Cep152/Cep164 together with the corresponding wide field image
3. Simulated ground truth example of Cep152/HsSAS-6