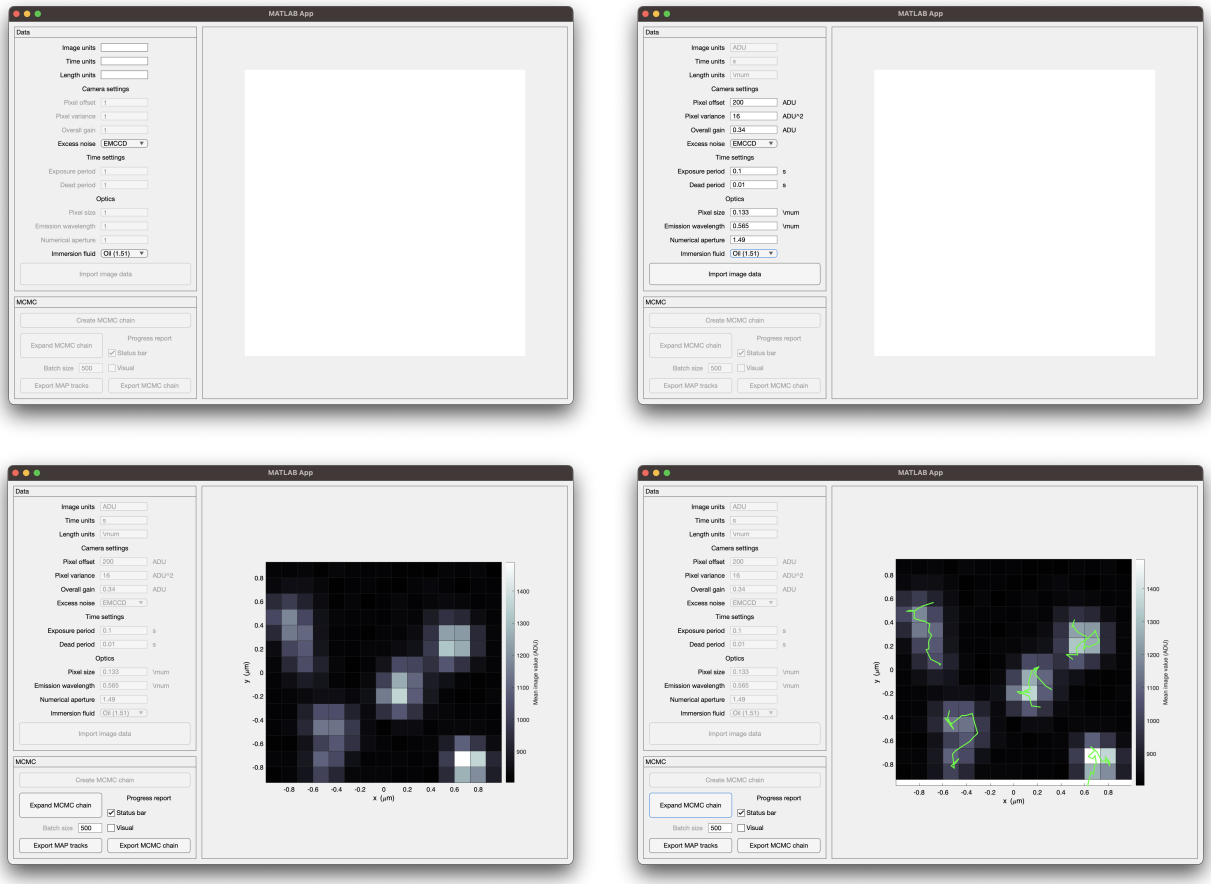

Figure 1: Graphical user interphase: (upper left) plain screen shown upon opening; (upper right) screen shown after configuring units of measurement and settings; (lower right) screen shown after importing image data and creating an MCMC chain; (lower right) screen shown during MCMC chain expansion. Can you add a, b, c, d on the panels similar to the rest of the paper?

1 BNP track software

To allow easy access to the methods developed in the study, we provide a computational implementation of our BNP tracking framework. This applies our non-parametric tracking algorithm adjusted to analyze images of Brownian particles acquired with a typical fluorescence microscope equipped with a single scientific grade camera imaging monochromatic light. For generality, in this basic implementation we consider only a uniform illumination profile and adopt standard hyper-parameter values.

Our implementation comes with a *graphical user interface* for easy use and also *source code*. The source code version is readily customizable and allows for modifications beyond our basic setup. Such modifications, among others, may include alternative hyper-parameter values, illumination profiles, point spread functions, or motion models.

Our software is written in Matlab (version R2023a) CITE which needs to be installed before BNP track can be used.

1.1 Graphical user interphase

Once started, our graphical user interphase appears as shown in fig. 1.

First, the units of measurement must be inserted. These include the units of the image values, time, and length. Typical units are ADU, s, and μm.

Then, the camera settings must be inserted. These include the values of the pixel offset, pixel variance, overall gain, and camera type. The latter allows two choices: EMCCD and sCMOS/CCD which set the excess noise factor to 2 and 1, respectively. The camera settings configure the noise statistics of the images to be analyzed.

Next, the time settings must be inserted. These include the values of the exposure period and dead time. The time settings configure the temporal frame of reference.

Next, the optical settings must be inserted. These include the values of the pixel size, imaged wavelength, numerical aperture, and objective type. The latter allows two choices: oil and water which set the index of refraction of the immersion fluid to 1.51 and 1.33, respectively. The optical settings configure the spatial frame of reference and the point spread function which, in this implementation is assumed to follow a standard 3D Gaussian model.

Finally, the image values must be imported via the "Import image data" button. This open a new dialog that allows selection of the file with the images. By convention, the images must be stored within a single 3D array named `w` and stored in a `.mat` file. Typically, the raw images generated by the microscope's camera must be converted into such a Matlab array prior to import. This array must have 3 dimensions with the individual frames arranged in the 3$^{\text{rd}}$ one. For instance, `w(:,:,1)` must contain the image values of the 1$^{\text{st}}$ frame, `w(:,:,2)` must contain the image values of the 2$^{\text{nd}}$ frame, and so on.

Once the image values are imported, an MCMC chain must be created via the "Create MCMC chain" button.

Subsequently, the MCMC chain must be expanded *until convergence*. This is achieved via *successive* applications of the "Expand MCMC chain" button. Each application expands the MCMC chain by adding a batch of new samples of the size indicated in the "Batch size" field. Generally, each expansion of the MCMC chain requires significant time. To monitor the progress of the expansion, the interface offers two choices: provide a progress report in the command window, which is activated by the "Status bar" checkbox, and show visually the MCMC trace plots, which is activated by the "Visual" checkbox. Because visualization on real time slows downs the computations significantly, it is turned off by default.

Once a chain is sufficiently expanded, the estimated particle tracks can be exported via the "Export MAP tracks" button. This exports a `.mat` file containing arrays of the maximum a posterior tracks. To obtain full estimates of every quantity of interest or allow further processing with other software, the interface export also the entire MCMC chain via the "Export MCMC chain" button.

1.2 Source code

The source code allows customization and flexibility in the analysis setup. To use this version, the driver script is `BNP_track_driver.m` which must be called from the command line within Matlab.

Within the driver script, the image values are loaded with the lines

```
load('demo_data.mat','w')

opts.w_cnt = w;
```

Here, the same formatting conventions apply as with the graphical user interphase. Namely, the image values must be contained within a single 3D array with the individual frames arranged along the 3rd dimension.

The units of measurements are inserted with the lines

```
opts.units.image = 'ADU';

opts.units.time = 's';

opts.units.length = '\mum';
```

The camera settings are inserted with the lines

```
opts.wM = 200;

opts.wV = 16;

opts.wG = 0.34;

opts.wF = 2;
```

The time settings are inserted with the lines

```
opts.t_ded = 0.01;

opts.t_exp = 0.10*ones(1,size(w,3));
```

Unlike the graphical interphase where the exposure period must be the same for all frames, the source code allows for varying periods among frames. For this reason, the exposure periods must be provided as an 1D array with the same size as the number of frames to be analyzed.

The optical settings are inserted with the lines

```
opts.x_bnd = 0.133*(0:size(w,1));

opts.y_bnd = 0.133*(0:size(w,2));

opts.lambda = 0.565;

opts.nRI = 1.51;

opts.nNA = 1.49;
```

Unlike the graphical interphase where the pixel size must be the same for all pixels and in both lateral directions, the source code allows for varying pixel sizes. For this reason, the pixel edges must be provided as 1D arrays. As they encode the position of the pixel edges, rather than the pixel widths, such arrays must have sizes 1 more than

the number of pixels in the images to be analyzed.

Finally, an MCMC chain is created with the line

```
chain = chainer_main([],0,opts,true,[]);
```

and expanded with successive calls of the line

```
chain = chainer_main(chain,500,[],true,false);
```

In particular, the latter line expands an MCMC chain by a batch size of 500 and shows only progress report in the command line. Expansions with different batch sizes are accommodated, for instance, with

```
chain = chainer_main(chain,1000,[],true,false);
```

which uses a batch size of 1000. Expansions with different progress reports are achieved with the lines

```
chain = chainer_main(chain,500,[],false,false);

chain = chainer_main(chain,500,[],false,true);

chain = chainer_main(chain,500,[],true,true);
```

The latter lines expand the MCMC chain by a batch size of 500 while showing no progress reports, only visual, and both status bar and visual, respectively.

1.3 Example dataset

For demonstration, we also provide an example image dataset configured as explained below. Our dataset is stored within `'demo_data.mat'` and contains successive frames of 5 Brownian particles diffusing at 0.01 μm^2/s with particle brightness of 10^5 photons/s and varying background around 10^6 photons/s/μm^2.

The camera settings use a pixel offset of 200 ADU, pixel variance of 16 ADU^2, overall gain of 0.34 ADU, and apply on an EMCCD type.

The time settings use an exposure period of 0.1 s and a dead period of 0.01 s.

The optical settings use a pixel size of 0.133 μm, a wavelength of 0.565 μm, a numerical aperture of 1.49, and apply on an oil immersion objective.