Documentation for

Single-molecule Platform for Automated, Real-Time Analysis (SPARTAN)

Daniel Terry

St Jude Children’s Research Hospital

262 Danny Thomas Pl, Memphis, TN 38105

February 22, 2022 (version 3.8.0)

To inquire about use of this software, report bugs, or request features, please send an email to:

Scott.Blanchard@stjude.org

Table of Contents

I. Introduction

This documentation describes software for the analysis of single-molecule fluorescence and FRET (smFRET) data. Written in MATLAB, tools are included for extracting traces from movies, selecting traces according to defined criteria, applying corrections, hidden Markov modeling, simulation, and data visualization. This documentation provides a full description of core functions and some examples of their use. More details are available in the associated publication (*Juette and Terry et al., Nature Methods* 2016).

Recommended system requirements:

MATLAB (source code version)**:** 2018a with Image Processing, Statistics, Curve Fitting, and Distributed Computing toolboxes.

Operating System: Windows 10 (Mac OS and Linux also supported but not extensively tested)

CPU: 64-bit Intel compatible x86 with at least four physical cores.

Memory: 16 GB RAM.

Storage: 100 GB free on a modern solid-state hard drive.

Version numbers are formatted as MAJOR.MINOR.REVISION. Major versions break compatibility by changing file formats, core function syntax, or fundamental features. Minor versions add new features and fix bugs, but largely preserve compatibility. Revisions are small changes that only fix bugs.

II. Installation of the compiled version

The compiled version of SPARTAN does not require MATLAB and is the easiest to set up.

1. Uninstall any previous versions of SPARTAN. On Windows, navigate to "Start→Control Panel→Uninstall a program". On Linux and Mac OS, simply delete the installation folder.

2. Download and execute *Install_SPARTAN* and follow the on-screen instructions. On Linux, run in a terminal as root or install in your home folder.

3. Once installed, to run SPARTAN on Linux, navigate to the installation folder and execute:
   ./run_Cornell_SPARTAN.sh /path/to/MATLAB/MATLAB_Runtime/v85

III. Installation of the source code version

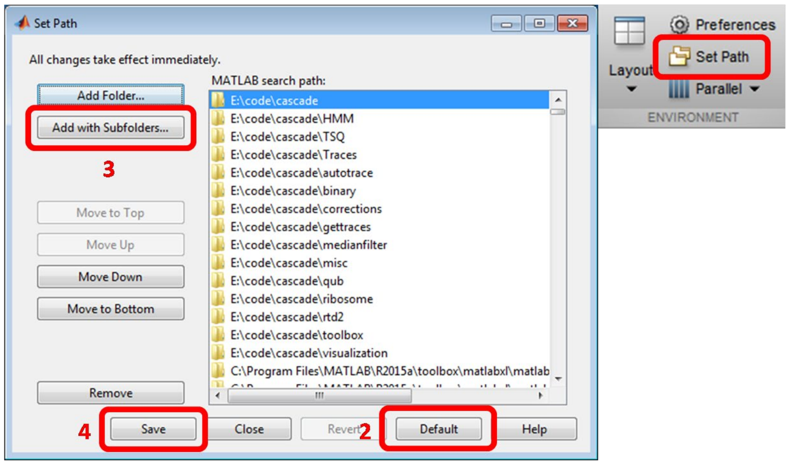
The source code version requires MATLAB to be installed, but also enables access to more functionality and the ability to write custom code that interfaces with SPARTAN functions.

1. Install 64-bit MATLAB with required toolboxes (see above).

2. Download the latest version of SPARTAN source: http://www.scottcblanchardlab.com/software

3. Unzip the folder to some location such as "My Documents/MATLAB".

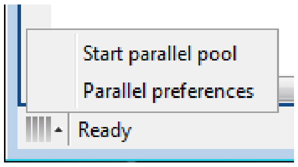
4. Open MATLAB, click on the "Set Path" button in the "HOME" toolbar panel.
   Click "Default" to removal all but the default code locations from the path.
   Click "Add with Subfolders" and select the SPARTAN source code folder.
   Click "Save" to keep the new path as the default and hit "Close".



5. *Optional:* download ebFRET version 1.1.1 and add the "src" folder to the MATLAB path. ebFRET will now be accessible in *batchKinetics* program as an additional Markov-modeling method. https://github.com/ebfret/ebfret-gui/releases/

6. Execute the command *spartan* at the MATLAB command prompt.

Known issues (source code version):



1. Computations may occasionally take a very long time while MATLAB starts the parallel pool. To fix this, click the parallel icon in the lower-left corner of the main MATLAB window and select "Parallel preferences" and set the timeout to 1000 minutes.

2. Some multi-processor computers may show long delays at the end of parfor loops. If you encounter this, set "constants.enable_parfor" to false in *cascadeConstants.m* to disable parfor.

3. Other programs and user scripts on the MATLAB path may conflict with SPARTAN. If you experience a problem, first try resetting the path (see above) and ensure *only one* version of SPARTAN is on the path and *no other files* are on the path. It is also a good idea to keep the "My Documents/MATLAB" folder clear of .m files to prevent conflicts.

4. If you encounter other problems, first try restarting MATLAB. If that does not work, contact us at Scott.Blanchard@stjude.org for assistance. Please include a full description of the problem.

IV. Main menu and core analysis workflow (*spartan*)

A main menu GUI (*spartan*) is provided for convenient access to key tools in the software suite. For compiled versions of the software, this window will appear when the program is launched. The current working directory is displayed in the textbox at the top and can changed by clicking the "Browse…" button. The general analysis workflow is as follows (described in detail in subsequent sections):

1) Acquire wide-field fluorescence movies as a TIFF format image stacks.
2) Run *gettraces* to extract fluorescence traces from the movies and calculate FRET.
3) Run *autotrace* to examine trace statistics and select a subset of useful traces.
4) Run *makeplots* to visualize the behavior of the ensemble of molecules.
5) Run *sorttraces* to examine the behavior of individual molecules and select representative traces.
6) Run *batchKinetics* to interpret FRET trajectories using hidden Markov modeling (HMM).
7) Simulate traces in *batchKinetics* to evaluate if proposed models are consistent with experiments.

For convenience, a GUI called *spartan* is provided to access all key functions from one place:

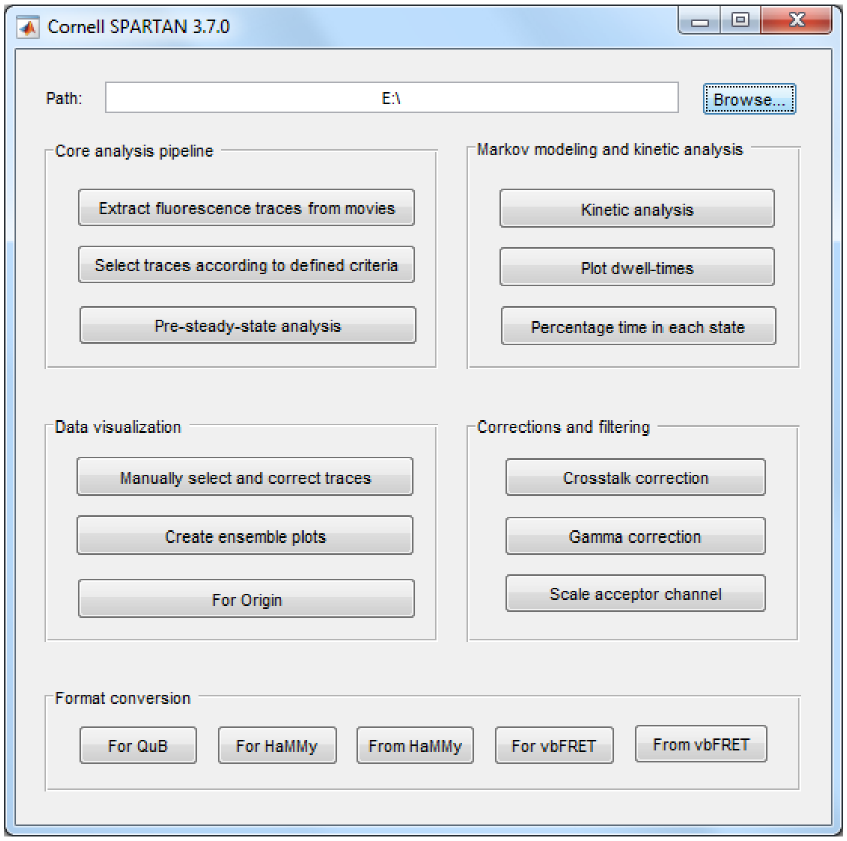


Figure 1: SPARTAN main menu.

V. Acquiring TIRF smFRET data

Fluorescence microscopy data acquisition software may be obtained from the camera vendor, from one of the sources listed below, or custom software implemented in a program like LabVIEW. All of these programs perform a similar function – to read out frames from the camera(s) and save the data as an image stack in TIFF-format files. The output of most such programs should be supported by SPARTAN, but please contact us if you find a problem.

https://www.micro-manager.org/
http://www.moleculardevices.com/systems/metamorph-research-imaging
https://cplc.illinois.edu/software/

Below is a list of suggestions for instrumentation and data formats:

1) **Fluorescence channel arrangement:** images from distinct spectral channels are generally tiled side-by-side in each frame (e.g., Cy3 left and Cy5 right). They can also be stacked sequentially (e.g., Cy3 in the first 1000 frames, Cy5 in the next 1000), but this requires modification to *cascadeConstants.m* to create a new profile for this purpose.

2) **Movie file format:** movies should be saved as monochrome TIFF image stacks. The BIG-TIFF format (http://bigtiff.org/) should be used for sCMOS movies since they often surpass the 2GB limit of the original TIFF format. Best practice is to name movies using the following format: "YYMMDD sample-name experimental-condition.tif". Each movie should have at least 50 frames and typically 1,000 to 3,000. Specify the exposure time as a float in seconds in an "ExposureTime" EXIF tag (33434). If this tag is not found in a movie, the user will be prompted for it, which will interfere with batch processing.

3) **Immobilization:** SPARTAN is tailored to experiments where donor fluorophore-labeled particles of interest are tethered to the surface and do not dissociate or move during the imaging period. Imaging of diffusing particles and/or corrections for stage drift are not supported.

4) **Alignment:** spectral channels should be manually aligned using fluorescent beads (Tetraspeck 0.1 µm, Invitrogen). SPARTAN's automatic alignment will handle any minor deviations.

5) **Microscope:** the microscope should feature stable, well-corrected optics, with each channel being *precisely parfocal and flat*. The illumination should be as even as possible within the field of view. The stage should be very stable and not drift within the imaging period. Finally, the objective correction collar should be adjusted for optimal PSF shape and intensity.

6) **Multiple cameras:** cameras should be as identical as possible – for example with sequential serial numbers to ensure they were manufactured at the same time – and properly calibrated. The user also must ensure that the cameras are temporally synchronized, generally using an external trigger, and aligned on the same axis to avoid rolling shutter artifacts (sCMOS).

Below are some additional tips for obtaining the best quality data:

1) **Hardware pixel binning:** Care must be taken when enabling hardware binning on cameras that support it. Binning pools together groups of pixels that are read together before the data are streamed to the computer, reducing data size and analysis time. (With EMCCD cameras, binning can have the added benefit of increased SNR, since noise is introduced when each pixel is read, and may also increase maximum frame rates.) Keep in mind, however, that binning also generally reduces the maximum density of particles that can be immobilized, limiting throughput.

2) **Immobilization density:** Small amounts of fluorescent background particles are often found on prepared glass imaging surfaces. When particles of interest are immobilized at low densities, background particles may contribute significantly to analysis results. To avoid this, it is best practice to occasionally take a movie under standard imaging conditions (including oxygen scavenging) to estimate the number of such particles and ensure an immobilization density at least 10-fold above this amount if possible. One should also avoid overloading the surface, with a 40% overlap rate at most (see next section).

3) **Minimum intensity:** Mean total fluorescence intensity should be at least 300 photons (20:1 SNR_{BG}) for best results. At very low signal intensity and SNR levels, some algorithms may fail. In particular, particles with a SNR_{BG} less than 5 typically are not registered because the photobleaching step is small compared to noise. To increase SNR, increase illumination intensity, time resolution, or both.

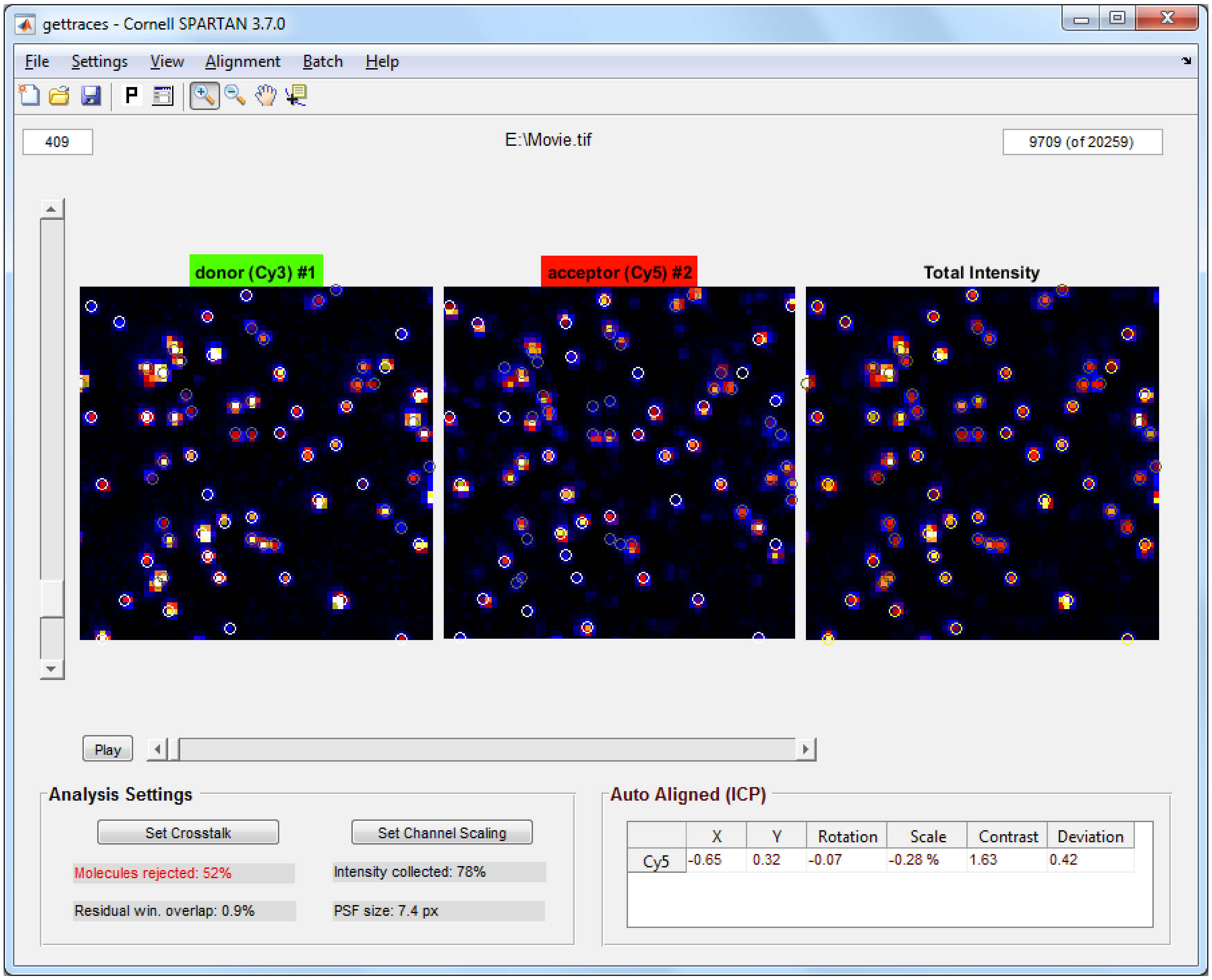
4) **Fast photobleaching:** If you acquire data in which the fluorophores photobleach substantially within 10 frames, some algorithms may not behave as expected. Specifically, *gettraces* uses a sum of the first 10 frames for detecting molecules by default. Molecules photobleaching within this window will appear dim and may not be detected, biasing the results. In such cases, you should reduce power (and/or increase integration time), use better-performing dyes, and/or improve the efficiency of oxygen scavenging. As a last resort, you can also safely decrease the number of frames summed in gettraces to 4.

5) **First frame:** charge build-up, shutter opening, and other artifacts are common in the first frame acquired by most scientific cameras. To avoid such artifacts, the first frame (and possibly more) should be removed in the acquisition software prior to analysis. Also ensure that shutter opening is optimally timed with the first frame exposure.

6) **Illumination uniformity:** *gettraces* (next section) uses a single global threshold for detecting peaks of intensity associated with fluorescent molecules. As such, it is important that illumination be as uniform as possible. Otherwise, molecules in regions with lower intensity may not be selected, possibly in a biased manner. For optimal results, illumination intensity should decrease by no more than 30% from center to edge.

VI. Extracting fluorescence traces from movies (*gettraces*)

Using TIRF illumination, wide-field fluorescence movies are obtained that contain many point-spread functions (PSFs), each associated with the fluorescence emitted by a single fluorophore-labeled particle. The first steps in smFRET data analysis are to detect these peaks of intensity, determine a mapping spectral channel function to associate PSFs that arise from the same physical particle, sum the fluorescence intensity from each PSF to obtain fluorescence-time traces, and calculate properly-corrected FRET-time traces. *gettraces* performs these analysis tasks and can be accessed with the "Extract fluorescence traces from movies" button in the *spartan* menu.



409

E:\Movie.tif

9709 (of 20259)

Molecules rejected: 52%

Intensity collected: 78%

Residual win. overlap: 0.9%

PSF size: 7.4 px

	X	Y	Rotation	Scale	Contrast	Deviation
Cy5	-0.65	0.32	-0.07	-0.28 %	1.63	0.42

Figure 2: Screenshot of the *gettraces* program.

1. Loading and viewing movies

First, choose an Imaging Profile from the "Settings" menu box at the top of the window. Profiles describe the imaging setup, including spectral channel assignments and default analysis settings. Some of these settings can be changed using the "Customize" option at the bottom of the "Settings" menu or

2. Detecting particles

Closely separated particles with potentially overlapping PSFs must be excluded from analysis because the resulting signal contamination adds noise to the fluorescence traces. Gray circles mark such excluded PSFs. The minimum distance between PSFs is specified in the “Settings→Customize” menu. The fraction of rejected molecules, shown in the bottom-left of the window, should approach 40% for optimal loading. The number of particles selected for analysis is shown in the upper-right corner. The first number is selected molecules; the second one in parentheses is all molecules.

3. Alignment of spectral channels

In most cases, cameras cannot be perfectly aligned due to optical imperfections. As such, by default, *gettraces* uses the Iterative Closest Points algorithm (ICP) to determine a mapping between corresponding molecules in each spectral channel. ICP iteratively finds a transformation that most closely overlays PSFs across all spectral channels. Once the algorithm converges, PSFs are re-detected from the aligned total fluorescence intensity image to minimize any potential bias towards the brightest channel. You can disable ICP or load a previous alignment from the “Alignment” menu.

ICP requires a strong signal on all spectral channels (0.2 to 0.8 FRET range). If one channel has little or no signal, or if if there is a large number of contaminating fluorescent particles only visible on one channel, ICP may fail. In such cases, you should record a short movie of fluorescent beads (TetraSpeck 0.1 µm, Invitrogen) at the beginning of the day, pick peaks using ICP, and select the “Alignment→Memorize” menu option to use this alignment for all subsequent movies. A table in the lower-right corner lists the alignment transformation parameters used. “Deviation” is the average residual misalignment between the donor and acceptor PSFs and should ideally be less than 0.4 pixels. “Contrast” is the Weber contrast score of the total intensity image, relative to a scrambled image. A value below 1.1 is not much better than a random alignment, while 1.4 indicates high contrast from a proper alignment.

4. Extracting fluorescence traces, apply corrections, and calculate FRET

Background levels after donor photobleaching are subtracted to more precisely zero the baseline. Donor to acceptor crosstalk is removed by subtracting a set fraction of donor intensity from the acceptor channel – this value can be changed by clicking the “Set Crosstalk” button. To correct for unequal acceptor channel brightness, the acceptor channel is scaled by a specified factor – this can be changed by clicking on the “Set Scaling” button. (For more information on these corrections, see the end of the next section). FRET-time traces are then calculated from the fluorescence traces, with FRET set to zero whenever the donor is dark. By default, donor blinking is detected with a threshold, but a more robust Markov modeling approach (SKM) can be selected from the Settings→Customize menu. The final fluorescence and FRET traces are then saved in a “.rawtraces” file with the same name as the movie.

5. Batch processing and automation

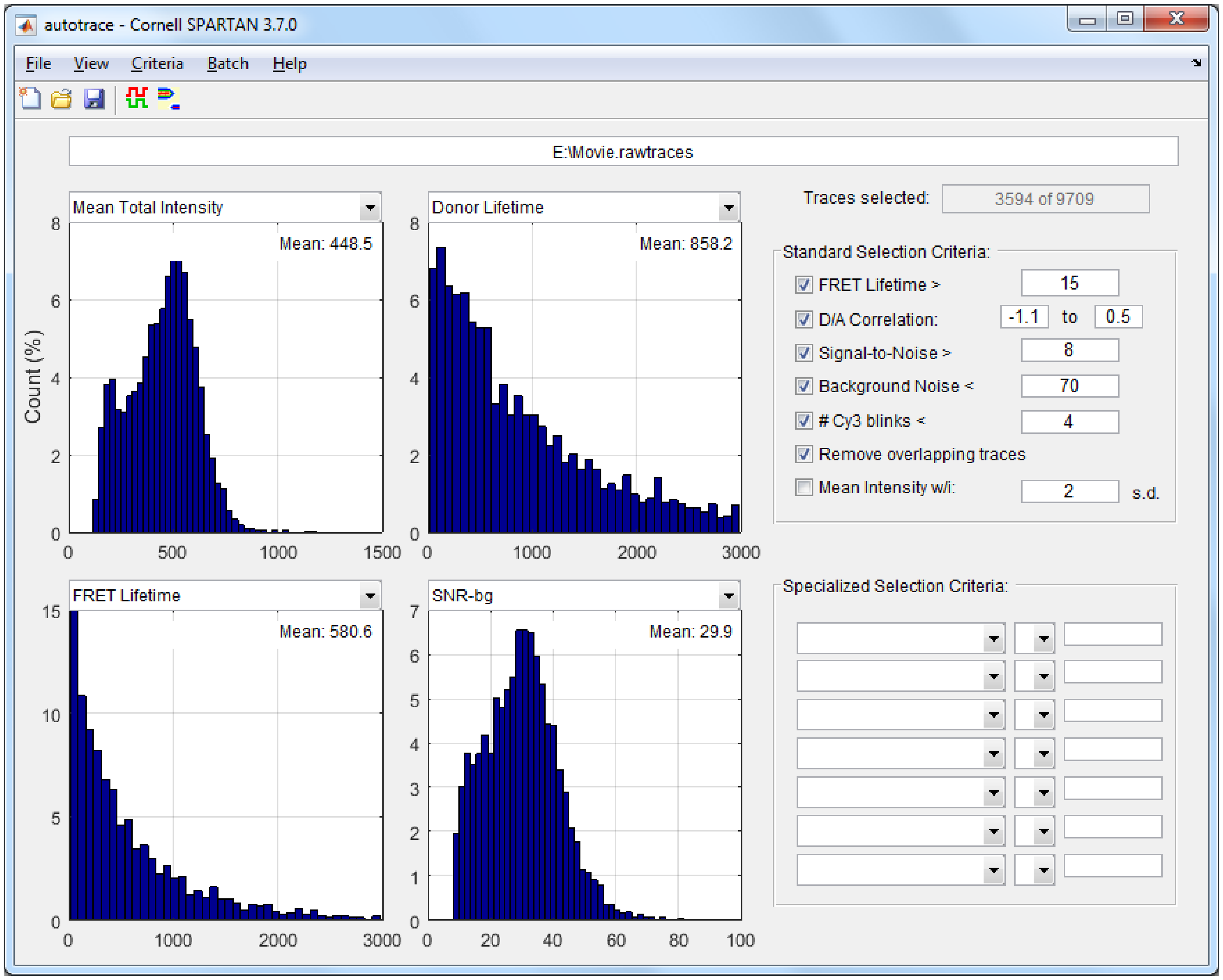
The movie analysis process can be automated. To analyze all movies in a directory using the current settings, click the “Batch→Select Directory” menu. If the “Look in subdirectories” checkbox is selected, *gettraces* will also search for movie files in all subdirectories within the selected folder. If “Skip processed movies” is selected, any movies with an associated “.rawtraces” file will be skipped. To automatically detect and process new movies as they are recorded, click the “Batch→Auto-detect new movies” menu. This last option is useful for simultaneous data acquisition and analysis. During automated acquisition, the user is advised to continuously monitor the “Deviation” score, PSF size, fraction of rejected, etc, to ensure analysis is being performed correctly.

Troubleshooting and tips

- Diagnostic fields in *gettraces* will turn red when the value is outside the expected range.
- Large PSF sizes, relative to the norm for the instrument, usually indicate poor or uneven focus. Optimal focus is very important for obtaining high quality data. If the focus cannot be improved, make sure all optics are clean, the correction collar is set to the optimal position, and the microscope slide / microfluidic imaging chamber is seated flat.
- If peaks are not detected at the edges of the field, the illumination may be low or non-uniform, or there may be optical artifacts or clipping of the imaging light path.
- If molecules are rejected disproportionately at the edge or the deviation scores are high, there may be a problem with instrumentation, such as an improper correction collar position, dirty optics, non-flat focus across the field, or optical distortions in the light path. This might also be an indication of alignment problems.
- Dim channels (Cy2 and Cy7) may show low contrast scores and warnings, even with a correct alignment (e.g., using beads). This is normal and arises from the low intensity of the channel.
- Zoom in occasionally to check the alignment, especially when deviation scores are high.

VI. Select traces according to defined criteria (*autotrace*)

SPARTAN provides a tool to summarize the behaviors of the large volume of data obtained in smFRET experiments, which can include 10,000 or more traces per movie. *autotrace* calculates various statistics that are useful for evaluating measures of data quality such as signal-to-noise ratios, photophysical behavior such as bleaching rates, and the presence of well-defined artifacts such as signal contamination from multiple distinct particles integrated into a single fluorescence trace. Selection criteria can then be defined for these statistics to extract molecules that exhibit desired behaviors such as high SNR, long lifetime before bleach, and the absence of obvious artifacts. This process is much faster and more reproducible than manual trace selection.



E:\Movie.rawtraces

Traces selected: 3594 of 9709

[✓] FRET Lifetime > 15
[✓] D/A Correlation: -1.1 to 0.5
[✓] Signal-to-Noise > 8
[✓] Background Noise < 70
[✓] # Cy3 blinks < 4
[✓] Remove overlapping traces
[ ] Mean Intensity w/i: 2 s.d.

Figure 3: Screenshot of the *autotrace* program.

1. Loading files and displaying trace statistic distributions

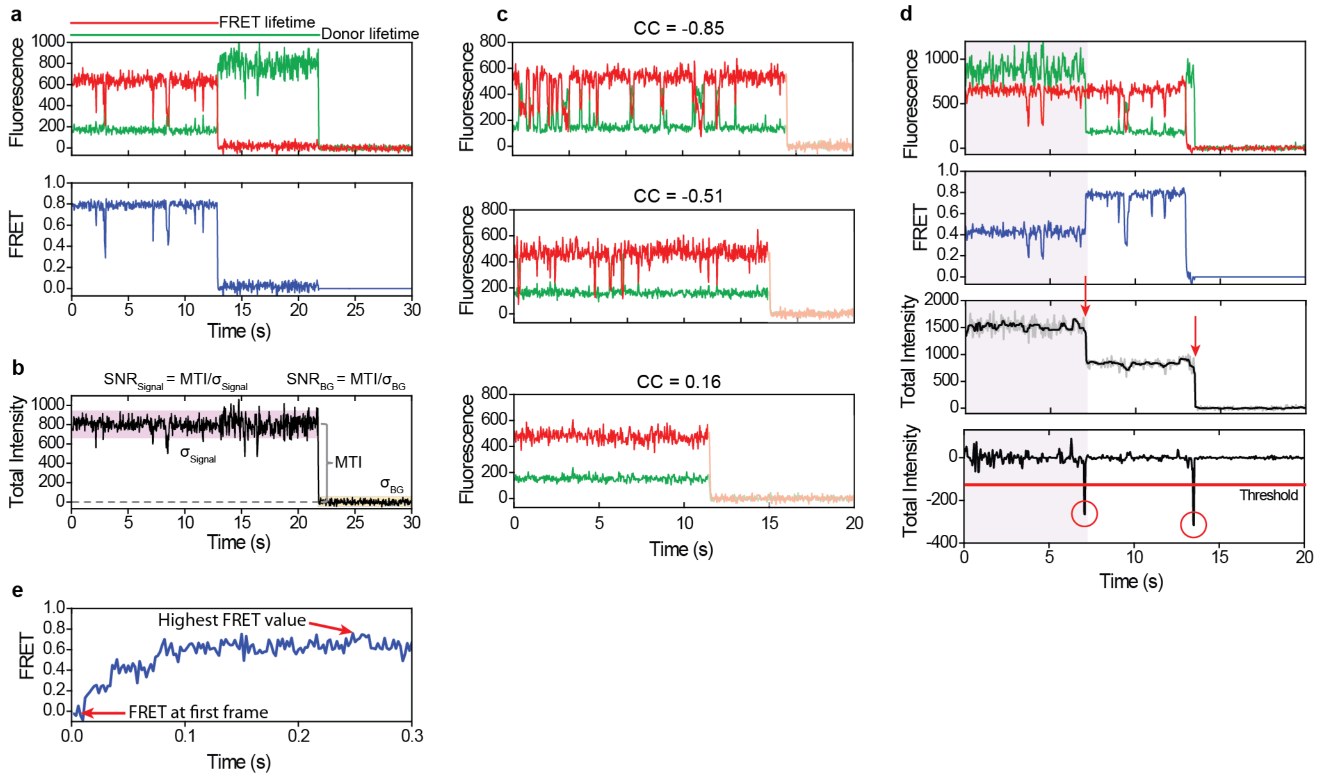


Figure 4: Visual description of commonly used trace selection statistics. (**a**) Donor (green) and acceptor (red) lifetimes before photobleaching. "FRET lifetime" includes only stretches of 5 or more frames above the noise level (0.125). Detection of donor photobleaching is described in panel d. (**b**) Calculation of two types of mean total fluorescence intensity (MTI) and signal-to-noise ratios (SNR). (**c**) Pearson's correlation coefficient of donor vs. acceptor fluorescence traces over the region prior to donor photobleaching, with several example traces that display different degrees of dynamics and therefore anti-correlation. (**d**) Detection of photobleaching steps is achieved by summing all fluorescence channels (third panel from top, gray), median-filtering this signal (black), calculating the gradient (last panel), and finding the large, negative changes in this signal that occur when the total intensity drops suddenly. (**e**) For pre-steady-state measurements, selecting traces that start in low-FRET states and achieve a particular high-FRET state are useful criteria to isolate "productive" events.

2. Choosing selection criteria

To select a subset of traces, use the selection criteria controls at the right side of the window. The most commonly used criteria are listed under "Standard Selection Criteria," with checkboxes to specify which criteria to apply. Additional selection criteria can be chosen under "Specialized Selection Criteria." Each line is a potential criterion, including the statistic name, an equivalence, and a value for the criterion (e.g., Donor lifetime > 50). Press Enter to ensure a change is registered. Any lines in which one box is empty are not applied. As criteria are added or changed, the subset of selected molecules will change accordingly, reflected in the number of selected molecules (top-right of window), and the histograms displaying the distributions of statistics. These histograms only show the values for selected molecules, not the entire dataset.

3. Saving selected traces

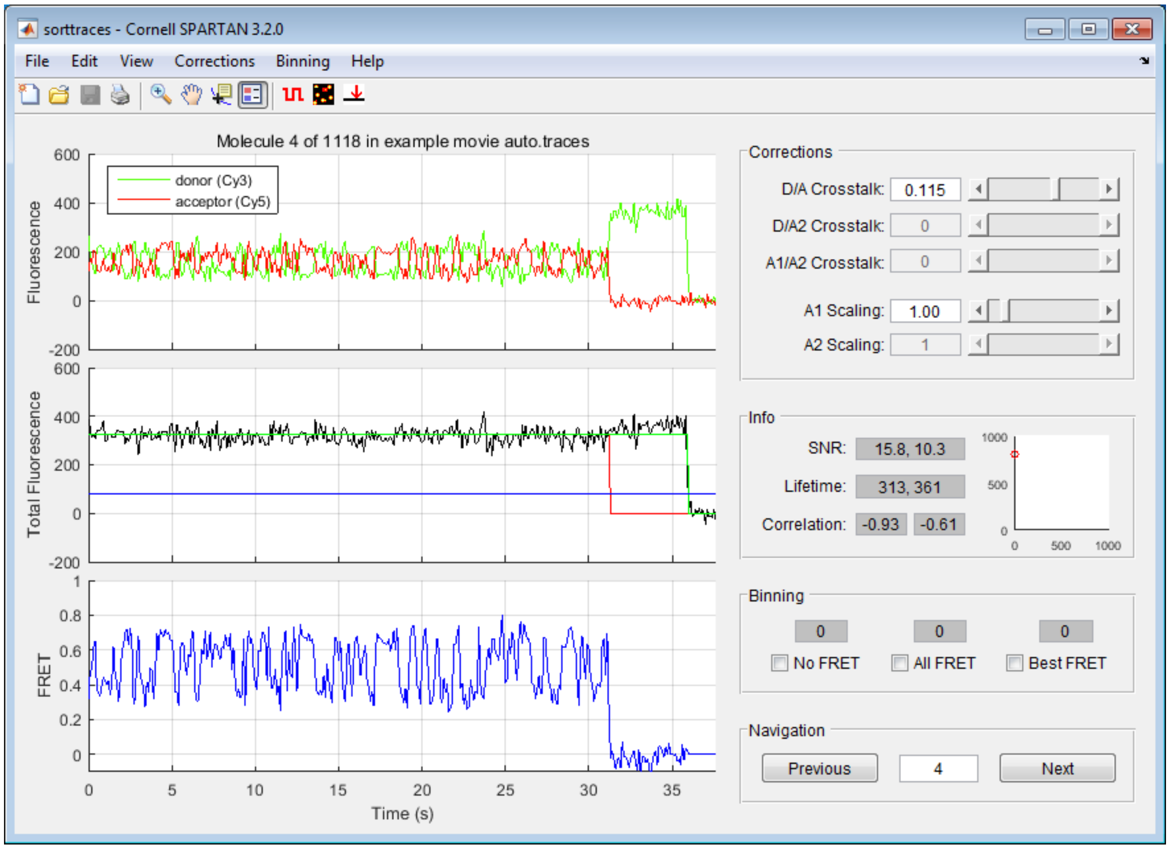
4. Batch mode

The process of loading a file and saving selected traces can be automated by selecting the "Batch→Batch Analysis" menu. For each .rawtraces file found in the selected directory and all subdirectories, *autotrace* will select traces according to the current criteria and save a corresponding "_auto.traces" file. This function is particularly useful when new selection criteria have been established and a large amount of experimental data must be reprocessed. The "Batch→Auto-detect new files" menu will automatically process new .rawtraces files as they appear in the target directory, which is useful for real time data analysis.

5. Post-processing: ensemble-level spectral corrections

Initial spectral corrections made in *gettraces* (see above) use pre-defined values, but these may need finer adjustments to correct for variability between samples. These adjustments can be made in automated ways using *crosstalkcorrect* and *gammacorrect*, or manually using *sorttraces*. *crosstalkcorrect* automatically finds the degree of donor to acceptor spectral bleed-through (crosstalk) and corrects for it. *gammacorrect* scales the acceptor channel so that the apparent brightness of the two channels is equal. Both methods estimate a single ensemble-average correction value for each file and applies it to all traces in that file. This approach is not effective in correcting for wide variation in these parameters, which may require manual adjustments in *sorttraces* (see the next section). Note that since *gammacorrect* is less stable than applying a fixed value, it may be better to use *gammacorrect* to establish an average value that can be applied to all data consistently in *gettraces*. By contrast, *crosstalkcorrect* is unlikely to increase variability in FRET values.

VII. Manually viewing, corrections, and select traces (*sorttraces*)



D/A Crosstalk: 0.115
D/A2 Crosstalk: 0
A1/A2 Crosstalk: 0
A1 Scaling: 1.00
A2 Scaling: 1

SNR: 15.8, 10.3
Lifetime: 313, 361
Correlation: -0.93 -0.61

No FRET 0
All FRET 0
Best FRET 0

Navigation: 4

Figure 5. Screenshot of the *sorttraces* program.

1. Loading and examining .traces files

Some trace statistics are also displayed for reference in the "Info" panel in the middle/right:

- **Lifetime:** donor and acceptor lifetimes before photobleaching (in that order).
- **Correlation:** Pearson's correlation coefficient of the donor and acceptor traces while the donor is alive. The right panel is the correlation coefficient for the current zoomed region of the trace.
- **SNR:** magnitude of total fluorescence intensity divided by the s.d. of background after donor photobleaching (left) and the total intensity signal before photobleaching (right).

- **Molecule location:** a red circle shows the position of the particle within the field of view.

2. Manual trace selection

Traces can be assigned to bins ("No FRET", "All FRET", and "Best FRET") using the checkboxes in the "Binning" panel. The bin names are arbitrary, but typically "No FRET" traces have major artifacts that cannot be used for further analysis, "All FRET" traces will be used for further analysis, and "Best FRET" traces are representative examples selected for publication. The number of traces in each bin is displayed above each checkbox. The bins can be renamed by right-clicking on the bin checkbox and choosing "Rename Bin." This menu has other options, such as clearing selections in a bin and moving to the next molecule within a bin.

3. Saving selections

4. Idealization

5. Making corrections to adjust for non-ideal properties of real-world data:

- **Spectral crosstalk** between fluorescence channels is corrected by subtracting a set fraction of the bluer channel from the redder channel. The "D/A crosstalk" control adjusts this value for donor to acceptor crosstalk. With the correct value, the acceptor fluorescence level after acceptor photobleaching should be zero and will not change upon donor photobleaching. The "D/A2" and "A1/A2" adjustments correct for additional pairs of channels with 3-color FRET.

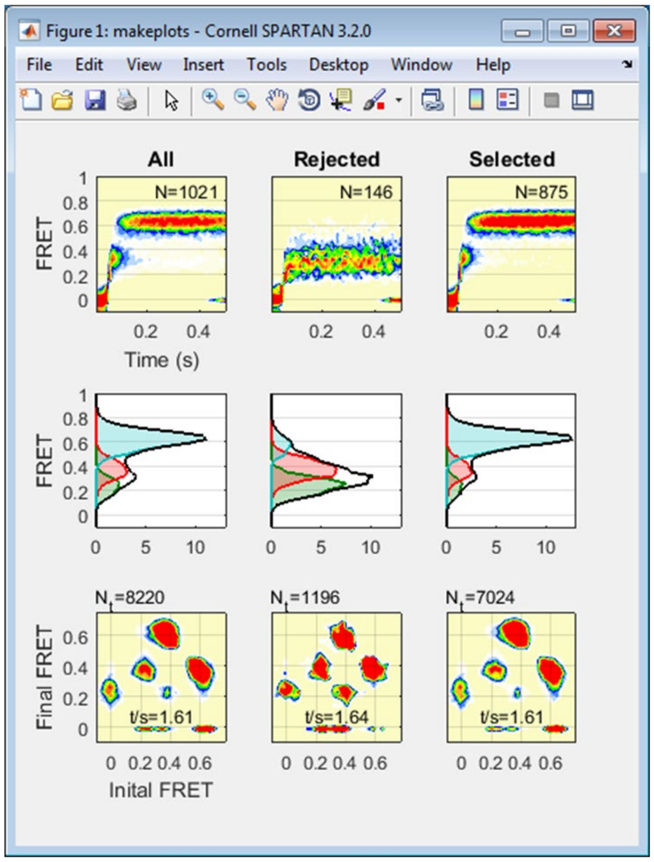
- **Uneven apparent brightness** of spectral channels is corrected by scaling the acceptor channel(s) by the factor given in the "A1 Scaling" (and "A2 Scaling") controls. These should be adjusted so that the total intensity is constant during FRET changes [3, 4].

.traces files are saved with all corrections applied, exactly as they appear. If you want to save the entire file with corrections, rather than a few selected traces, click the "Save current file as…" button. To reset the entire file to its original state click on the "Corrections→Reset All" menu.

VIII. Visualizing trace ensembles

makeplots is a program for visualizing the ensemble of single molecule behaviors using a set of standard plots. The first row contains FRET-time contour plots, which show the evolution of the system over time, along with the number of molecules in the file. In equilibrium experiments, these will be constant except for accumulation at zero FRET efficiency due to photobleaching. If no idealization data is available, the second row contains FRET histograms summed over the time span in the contour plots (use *frethistComparison* to overlay these histograms). If an idealization (.dwt) file is found with the same name as the .traces file, the second row of plots contains a set of overlaid FRET histograms, one per FRET state, along with the total histogram as a black line. The third row contains transition density plots [1], the total number of transitions (Nt), and the average transition rate (t/s).

Note that the TD plots generally include all data in the file, not just the window shown in the contour plots. This behavior and other settings can be changed from the "Edit→Settings" menu. These settings will be remembered for all future calls to *makeplots*. To reset all settings to their defaults, click on the "Edit→Reset settings" menu. Clicking the "File→Export .txt files" menu will create text files of the histogram data for plotting in external programs such as Origin.



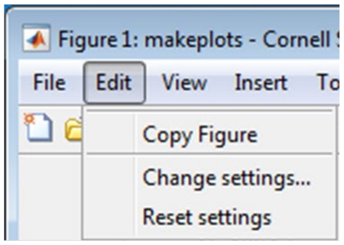
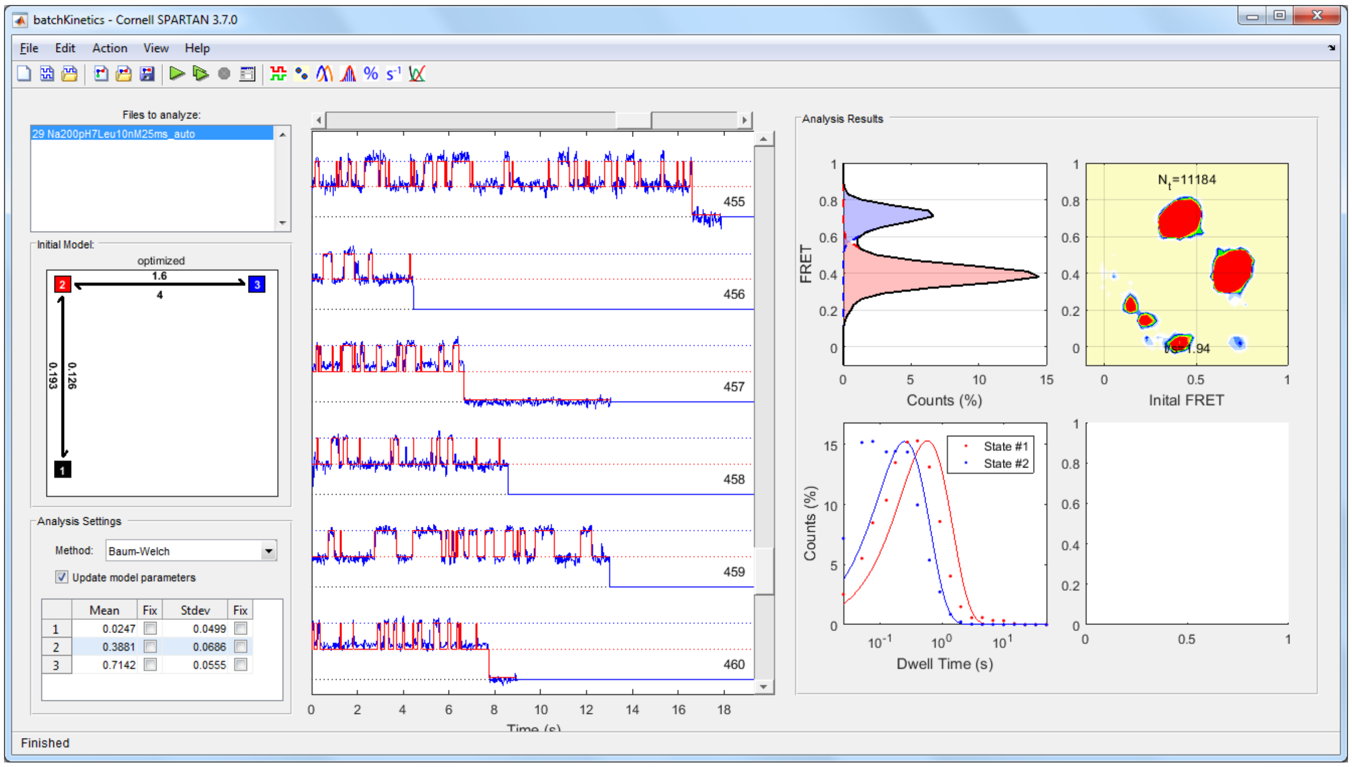


Figure 6: Ensemble plots generated with the *makeplots* program.

IX. Hidden Markov modeling (*batchKinetics*)

Interpretation of smFRET data often requires two key tasks: 1) to define a model that fully describes the behavior of the ensemble of particles (model optimization) and 2) for each particle, estimate the state of the underlying physical system at each point in time given the observed smFRET data (idealization). These tasks are generally accomplished within the framework of hidden Markov modeling [5]. HMM tools for SPARTAN are implemented in the *batchKinetics* GUI. HMM tools from other labs, which perform similar tasks in similar ways, include the following:

QuB	http://www.qub.buffalo.edu/
HaMMy	http://bio.physics.illinois.edu/HaMMy.asp
ebFRET	http://ebfret.github.io/
SMART	https://simtk.org/home/smart



Files to analyze:
29 Na200pH7Leu10nM25ms_auto

Initial Model:

Analysis Settings

Method: Baum-Welch

Update model parameters

	Mean	Fix	Stdev	Fix
1	0.0247		0.0499	
2	0.3881		0.0686	
3	0.7142		0.0555	

Finished

Figure 7: Screenshot of the *batchKinetics* program.

HMM analysis requires to main inputs: FRET-time traces and a model. Steps for loading and manipulating the data and starting model are given below:

1. **Select the smFRET data files to analyze**

2. **Loading or creating an initial model**

Clicking on any rate constant opens a dialog box that allows the rate constant to be adjusted and the parameter to be fixed (not changed during optimization). Any 'fixed' rate constants will be grayed out to indicate they will not be optimized. To change state parameters, right click on the desired state and choose "properties". Observation parameters and constraints can also be adjusted using the table in the bottom left corner of the GUI. In both cases, checking the "Fix" box will prevent the indicated parameter from being adjusted during optimization.

3. Performing analysis on a single file

SPARTAN implements a number of methods described in the literature for modeling FRET dynamics, each of which has its own advantages and disadvantages. The analysis method can be selected using the “Method” dropdown box in the “Analysis Settings” panel in the lower-left corner of the window. Additional settings such as convergence criteria can be set using the “Edit->Analysis Settings” menu.

- **SKM** (segmental k-means) [6] iteratively idealizes the FRET data using the current model and uses this idealization to re-estimate model parameters. This algorithm is fast, but not well suited to estimating kinetic parameters. It is often coupled with MIL for estimating rate constants.
- **Baum-Welch** [5] uses the forward-backward algorithm to optimize model parameters. Unlike SKM, Baum-Welch uses a fully probabilistic approach, which allows it to provide a more precise estimate of model parameters than SKM. Baum-Welch optimizes transition probabilities rather than rate constants, so it does not support kinetic constraints. Like SKM, Baum-Welch can be coupled with MIL for more accurate rate estimates.
- **MPL** (maximum point likelihood) [7] optimizes rate constants directly, which enables constraints on kinetic parameters and more accurate estimates of rate constants. Unlike SKM and Baum-Welch, which use heuristic methods to optimize model parameters, MPL uses MATLAB’s generic optimizer function *fmincon* to optimize the likelihood function directly using a gradient approach. This algorithm is slow, but very flexible.
- **MIL** (maximum interval likelihood) [8] optimizes rate constants using dwell times as the input instead of FRET traces. This dramatic reduction in data complexity enables faster and more robust optimization of even complex models. SKM or Baum-Welch are most commonly used for the initial idealization step before running MIL.
- **ebFRET** [2] uses a Bayesian approach to discover a model that best explains the observed FRET data. This method is slow, but requires no initial model – only a number of states. ebFRET is an iterative version of vbFRET [9] that uses the posterior probabilities of all traces to refine the prior probabilities in each iteration. (The number of states to use is determined from the model currently loaded in batchKinetics only determines the number of states; all other parameters are ignored).

4. Visualizing the results

and transition density plots (*makeplots*); occupancy in each state (*percentTime*); evolution of state occupancy over time (*occtime*), useful for pre-steady-state measurements; and average transition rate (*transitionsPerSecond*).

5. Simulating smFRET data

batchKinetics can also be used to simulate smFRET data according to the currently-loaded model. This feature is primarily useful to test the validity of a model derived from hidden Markov modeling of experimental data by evaluating if the features of the simulated and experimental data match (e.g., state occupancy, transition density plots, etc). It can also be useful to understand how stochasticity and specific artifacts contribute to non-ideal behavior in smFRET data, and the biases of various analysis approaches. Only basic noise sources, such as Gaussian-distributed background noise and Poisson-distributed shot noise are simulated, which together in general do not reproduce the noise of experimental smFRET data, but provide a useful approximation.

Here is a list of parameters for simulation. Target values can be estimated using *autotrace* and experimental smFRET data.

- **Integration time** (ms): simulated continuous-time dwells are binned to this time resolution.
- **Intensity** (photons): mean total (donor+acceptor) photon counts in each frame. (Non-uniform illumination can also be simulated using the "Intensity stdev" input box).
- **Signal: background noise ratio:** controls the amount of Gaussian-distributed background noise to add. (NOTE: when simulating wide-field movies, this parameter should be high (>100) since background noise is added with the background movie.)
- **Shot noise:** if checked, Poisson-distributed shot noise, which is the variance associated with photon counting statistics, will be simulated by drawing Poisson-distributed numbers at each frame, with the mean being the underlying fluorescence value.
- **Apparent Gamma:** specifies the apparent brightness of the acceptor relative to the donor, simulating the effects of unequal quantum yield and detection efficiency. A value of 0.5, for example, means that the acceptor is half as bright as the donor. Note: If this value is not unity, the mean total intensity in autotrace will only match the simulated value after gamma correction.
- **Exponential photobleaching:** if checked (default), traces will be terminated in simulated acceptor and donor photobleaching, with times drawn from exponential distributions. If not checked, fixed bleaching times are used. The time constants are specified in the "Acceptor Lifetime" and "Donor Lifetime" fields.

6. Simulating wide-field movies (optional)

To most faithfully reproduce the qualities of real data, wide-field fluorescence movies can also be simulated. This method requires that a background movie be acquired under *exactly the same conditions* as the target experiment, except that the biological sample is not immobilized on the surface.

- **Grid:** if this checkbox is selected, fluorophores will be placed on a regular grid of positions, separated by the widest possible margin to achieve the target number of fluorophores per movie. Each position is then randomly perturbed by up to 1 pixel to simulate the effect of fluorophores not perfectly aligning to pixel boundaries, as in experiments.
- **Density:** specifies the number of particles to place within the field of view.
- **Point-spread function size:** specifies the size (standard deviation, in pixels) of the symmetric Gaussian point-spread functions. Each simulated trace's intensity is distributed across a region of the field of view, centered at the fluorophore's actual location. Choose a value so that *gettraces* shows the same effective PSF size (counted in that case as the number of pixels to get 80% collected intensity) as the target experimental data.
- **Field alignment:** use these parameters to define misalignment between the two spectral channels. The values can be derived using *gettraces* and example data.

7. Output

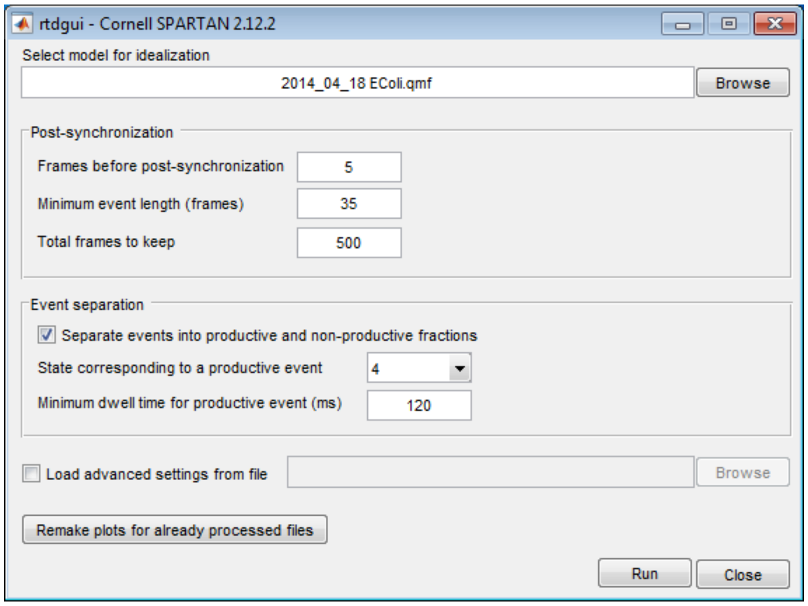
When the "Simulate…" button is clicked, a progress bar will appear that estimates the time remaining for the computation. If the rate constants in the model are much faster than the imaging rate, this computation can take minutes. Once finished, several files are generated, all with the same base name as the output filename chosen by the user:

- **___.sim.log:** documents the settings used for generating the simulated data.
- **___.sim.dwt:** simulated dwell-times, which may be much shorter than the actual imaging time resolution. Because of limitations in the .dwt file format, all times are rounded to integer milliseconds.
- **___.traces:** simulated single molecule fluorescence- and FRET traces. If simulating wide-field movies, this may not be useful.
- **___.tif:** wide-field fluorescence movie, if requested.

X. Analysis pipeline for pre-steady-state experiments (*rtdgui, rtdTool,* and *rtdPlots*)

Pre-steady-state smFRET experiments are carried out by real-time delivery (RTD) of reagent to surface-immobilized molecules. Turnover of the monitored reaction coordinate proceeds in a non-synchronized fashion with a distribution of waiting times. *rtdgui* is a tool to analyze and display the synchronized time evolution of an asynchronous non-equilibrium ensemble. It takes as its input any trace file, typically a ".rawtraces" file generated by *gettraces*, and performs the following steps in an automated fashion:

1. Select traces according to a set of criteria using *autotrace*. In addition to typical default criteria, traces that start out with FRET efficiencies greater than a low-FRET threshold are excluded to remove molecules that have already turned over at the beginning of the movie.
2. Post-synchronize FRET events to the first appearance of FRET. This point is determined by idealizing the data to a simple "on/off" two-state model using SKM.
3. Idealize the post-synchronized events using a kinetic model appropriate for the system (see *batchKinetics* documentation for details). By default, this uses a model for aa-tRNA selection monitored using tRNA-tRNA FRET, by a custom model for any system can be provided by the user.
4. (Optional) Split the post-synchronized ".traces" file into two populations termed productive and non-productive. Productive events are those that spend a given minimum dwell time in a specified state. In the case of aa-tRNA selection, this identifies successful accommodation events.
5. Generate an informative set of plots for each population (this is an extension of *makeplots*).


rtdgui - Cornell SPARTAN 2.12.2

Select model for idealization: 2014_04_18 EColi.qmf

Post-synchronization

Frames before post-synchronization: 5

Minimum event length (frames): 35

Total frames to keep: 500

Event separation

[✓] Separate events into productive and non-productive fractions

State corresponding to a productive event: 4

Minimum dwell time for productive event (ms): 120

[ ] Load advanced settings from file

Figure 9: Screenshot of the *rtdgui* dialog.

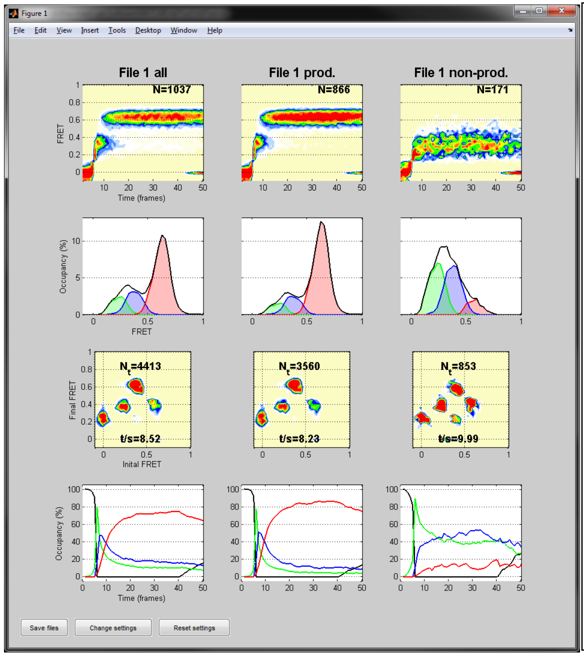
1. Defining settings and running the analysis pipeline

In the *rtdgui* dialog, select a kinetic model appropriate for your experiments. By default, the kinetic model for aa-tRNA selection in E. coli (using the tRNA-tRNA FRET signal) is selected. Click “Browse” or type a path to seect a different model. Models need to be in the “.qmf” format generated by QuB.

In the next section, “Event separation”, enable the checkbox if you want events to be classified into productive and non-productive events, based on a minimum dwell time in a specified state. You can select the “productive” state in the dropdown and type in a minimum dwell time in milliseconds in the field underneath.

Click “Run” to bring up a file selection dialog and choose any “.rawtraces” or “.traces” files that you want to process. The dialog reappears to allow multiple selections until you hit “Cancel”. At that point, the automated analysis procedure starts running. Depending on your computer and the selected files, this can take anywhere from a few seconds to several minutes. If you are running the program within MATLAB, update messages will appear in the command window.

When processing is completed, a visualization window with multiple columns of plots is launched. Each column corresponds to one output “.traces” file. If event separation is active, each input file generates three columns: “all events”, “productive events”, and “non-productive events”. The first three rows of plots are identical to the output that *makeplots* would generate (FRET-time contour plots, overlaid FRET histograms for the FRET states contained in the model, transitions density plots summarizing the FRET values immediately before and after observed state transitions). The last row shows the occupancy of each state (including the no-FRET/photobleached state) over time, using the same color scheme as the second row.



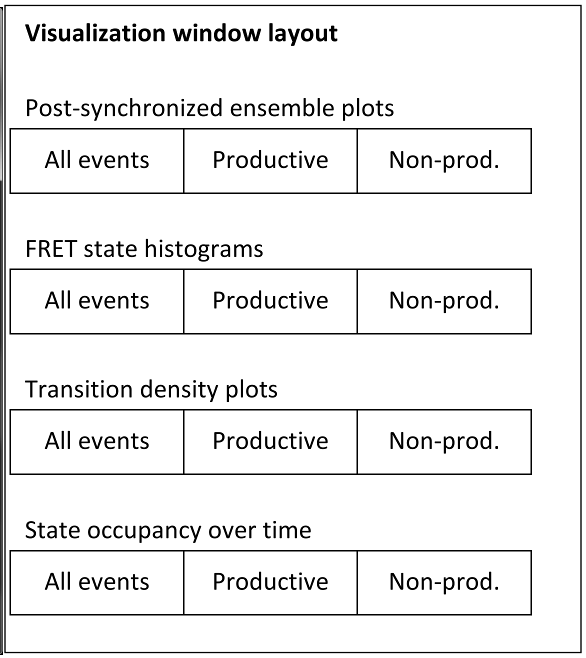


Figure 10: Post-synchronized ensemble plots generated with the *rtdgui* program.

2. Reviewing plots for previously processed data

In order to bring up the visualization window for already processed data, click "Remake plot for already processed files" in *rtdgui* and select all ".traces" files to be included in the plot window. This feature can also be called directly from the MATLAB command line by typing "*rtdPlots*".

3. Customization and advanced settings

rtdgui can optionally load advanced settings from file by enabling the corresponding checkbox and selecting an appropriate file. This allows the user to better adjust parameters for the different steps of the analysis to their specific data, or to programmatically specify which files to analyze (bypassing the file selection dialog). This functionality is currently only available in the source code, not in the compiled version of the software. The settings file is a MATLAB script (".m" format) that builds a "customOpt" structure containing the user-specified settings. For details, including allowed entries and an example, refer to the file "advanced_settings_example.m" in the rtd2 directory of the source code.

Alternatively, the core function performing the RTD analysis, *rtdTool*, can also be called directly from the MATLAB command line or in scripts. Type "help rtdTool" for more information.

XI. Customization

In some cases, users will need to change the behavior of certain functions for which there are no convenient GUI controls. *cascadeConstants.m* contains nearly all parameters in the software suite. Keeping all settings in a single location ensures only a single file is ever modified by the user, making changes more traceable. More detailed descriptions of each setting are available in *cascadeConstants.m*. The file is broken into several sections:

1. **Global:** these may apply to many functions.

2. ***Gettraces*:** default settings for each of the imaging profiles, including some settings that cannot be changed in the GUI. With some care, new profiles can be added without disrupting the other profiles in the list. Note that some settings common to several profiles are defined at the beginning of this section.

3. ***Autotrace*:** defines the default criteria and parameters used for calculating statistics in *traceStat.m*.

4. ***Makeplots*:** plotting settings such as the number of frames to display in FRET-time contour plots. Includes some settings unavailable in the GUI. Some other plotting functions such as *frethistComparison* also use these settings.

5. **Other:** miscellaneous settings. Of note, this is where you can turn off parallel processing.

Despite our best efforts, some settings may need to be modified within individual functions.

New statistics for *autotrace* can be made relatively easily by altering *traceStat.m*. Any new statistic added to the output struct array (and the statistic description struct) will be recognized seamlessly by *autotrace*.

Appendix A. Full list of MATLAB functions

The software includes a large number of MATLAB functions, including some for specialized purposes. Below is a list with a brief description of each. To find out more about each program, type "doc *function*" with the name of the function at the MATLAB command prompt. Lines in gray are internal functions that are rarely or never used by end users directly.

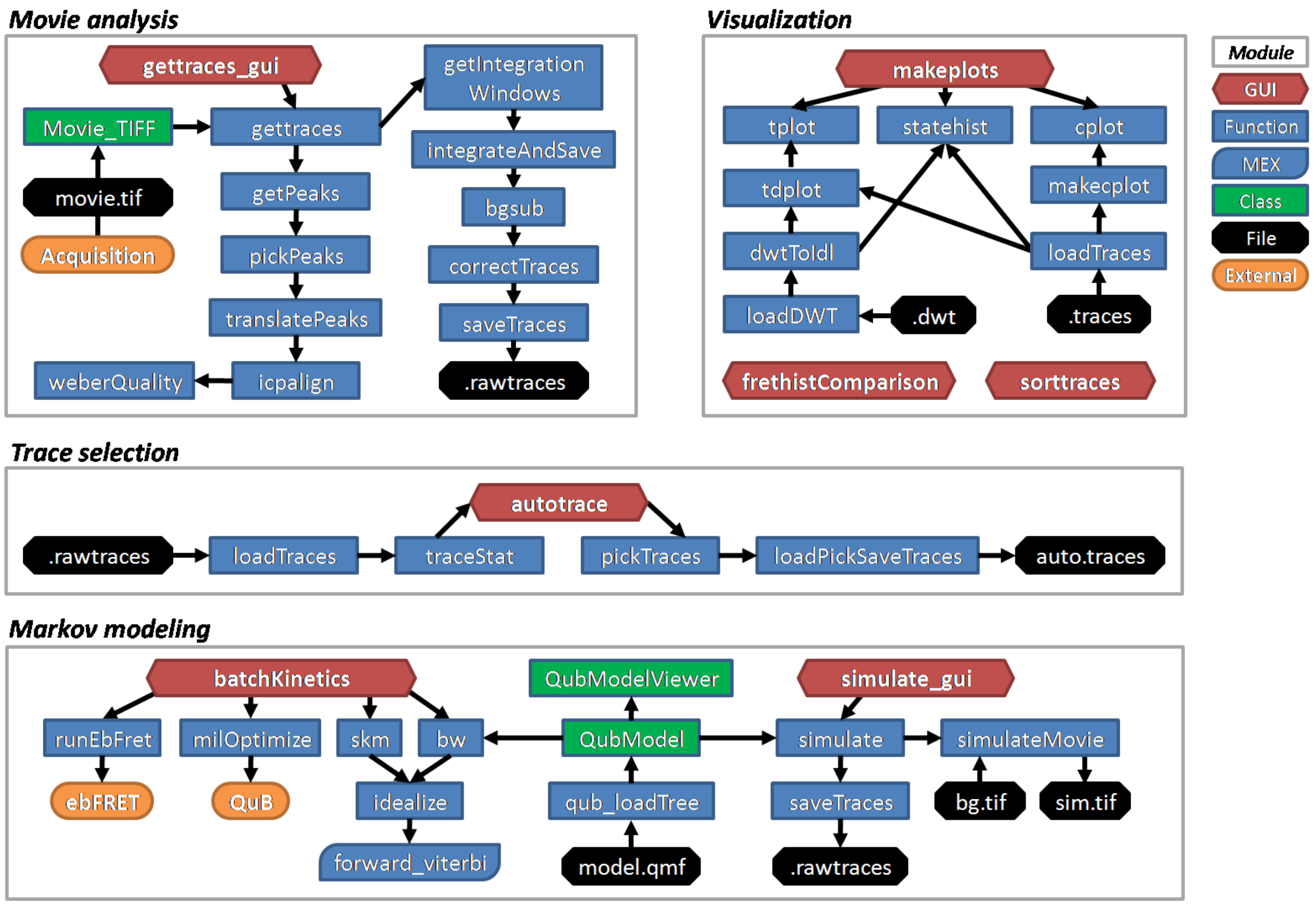


Figure 11: schematic overview of key MATLAB functions.

<u>Gettraces:</u>

gettraces_gui: user interface for loading movies and extracting fluorescence traces.
gettraces: command-line backend function used for all computation in the above.
pickPeaks: discover local intensity maxima associated with immobilized, fluorescent particles.
icpalign: iterative closest points algorithm for aligning fluorescence fields.
getCentroids: fast, approximate calculation of center of mass of each PSF.
subfield: quick way to extract a quadrant or sub-field by name (e.g., TL=top left).
translatePeaks: transform reference (donor) PSF positions to get expected acceptor PSF positions.
weberQuality: Weber contrast score for quantifying the quality of an alignment.

Wavelength_to_RGB: produce an RGB color that mimics light of a particular wavelength.

Autotrace:
autotrace: user interface for viewing trace statistic distributions and saving selected traces.
traceStat: calculates values of statistics for each trace in a dataset.
calcLifetime: find the donor photobleaching point; used by *traceStat*.
medianfilter: sliding window median filter, used by *calcLifetime*.
rleFilter: set to zero stretches of binary data that are not longer than a specified length
RLEncode: run-length encoding
pickTraces: select a subset of traces according to defined criteria.
loadPickSaveTraces: load traces file(s) and save traces picked according to defined criteria.

Trace corrections:
bgsub: subtracts fluorescence after photobleaching to zero the baseline level.
crosstalkcorrect: correct for donor-to-acceptor spectral crosstalk (one value per file)
gammacorrect: correct for unequal donor and acceptor apparent brightness (one value per file)
scaleacceptor: similar to *gammacorrect*, but a scale by a factor specified by the user.
correctTraces: internal function to apply corrections in a consistent order.

Visualization:
sorttraces: visual inspection, correction, and selection of individual traces.
forOrigin: save traces as a text file to be imported into Origin for making publication-quality figures.
makeplots: displays FRET-time contour plots, state occupancy, and transition density plots.
makecplot, cplot: calculate and display FRET-time contour plots from FRET traces.
statehist: calculate FRET histograms for each idealized state (middle row in makeplots)
tdplot, tplot: create and display transition density plots.
frethistComparison: overlays FRET histograms from multiple files for direct comparison.
avgFretTime: displays the change in average FRET value across all traces over time.
occtime: state occupancy over time, averaged over all traces.
percentTime: calculate the fraction occupancy in each FRET state from a .dwt file.
dwellhist: displays FRET state lifetimes, as exponential or Sine/Sigworth plots, from multiple files.
lifetime_exp: Fits the exponential decays to get average dwell-times, using *dwellhist*.
transitionsPerSecond: average number of transitions per total time over entire file.

HMM:
batchKinetics: GUI for idealization, model optimization, and kinetic analysis.
skm, idealize, forward_viterbi: segmental k-means and Viterbi algorithms for idealization.
forwardBackward: forward-backward algorithm for hidden Markov modeling.
milOptimize: maximum interval likelihood kinetic model optimization algorithm.
mplOptimize: maximum point likelihood model optimization algorithm.
BWoptimize: Baum-Welch model model optimization algorithm.
runEbFret: runs ebFRET (external program installed on path).
simulate: simulate smFRET data.
simulateMovie: simulate wide-field TIRF movies in smFRET experiments.

loadDWT, *saveDWT*: load and save .dwt files to disk.
dwtToIdl, idlToDwt: convert dwell-time series to an idealization (state assignments), and vice versa.

Compatibility with file formats used by other programs:
forHammy: export data in a .traces file to a format the HaMMy software can load.
hammyToDWT: convert the output of HaMMy to a .dwt file.
forvbFRET: export data in a .traces file to a format the vbFRET software can load.
vbFRET_dwt: convert the output of vbFRET to a .dwt file.
forQuB: export data in a .qub.txt file to be loaded into the QuB software.
tracesToTxt: export data in a .traces file to a .txt file.
tracesToMat: export data in a .traces file to a .mat file for importing into other software.
cy5ForQuB: convert Cy5 fluorescence data into a format that can be read by QuB.
qub_loadTree, qub_saveTree: load QuB format binary files (e.g., .qmf).

Other:
getFile, *getFiles, getFileGroups*: ask the user to select a list of files and return them as a cell array.
haranfilter: filter traces to remove noise while preserving anti-correlated transitions.
combineDatasets: combine traces from multiple .traces files.
loadTraces, saveTraces: load and save trace data to a binary format for fast access.
sizeTraces: determine the number and length of traces in a file without loading the full file.
resizeTraces: truncate all traces within a file. Useful for ensuring all files have the same length.
rebintraces: simulate lower time resolution by time binning. (somewhat outdated)
avgfret: get an average FRET value across all traces in a file.

Classes:
Movie, *Movie_STK*, *Movie_TIFF*: classes for loading movie data from disk.
MovieParser: load movie, detect molecules, and save traces (see *gettraces*).
Traces: abstract class that defines a generic collection of trace data and metadata.
TracesFret: class defining a collection 2-color fluorescence- and FRET-time traces.
TracesFret4: class defining a collection of multicolor fluorescence- and FRET-time traces.
TraceListViewer: scrollable trace viewer (used by *batchKinetics*).
QubModel: object representing a Markov model, including emission and kinetic parameters.
QubModelViewer: GUI methods for displaying a model, as in *batchKinetics*.

Pre-steady-state analysis:
rtdgui: user interface for setting parameters and selecting data for automated analysis.
rtdPlots: command-line tool for recreating plots for data analyzed by *rtdgui* or *rtdTool*.
rtdTool: command-line backend function used for all computation in the above.
postSyncTraces: iterative post-synchronization of FRET events based on two-state model.
simplePostSync: simple, threshold-based post-synchronization of FRET traces.
stateMin: classify FRET events as productive or non-productive.
stateOccupancy: calculate the occupancy of each state as a function of time.

Appendix B. File formats

The various file formats used in the analysis pipeline are listed here for reference.

1. ***TIFF stacks from wide-field fluorescence movies (.stk, .tif)***

Movies are standard TIFF format image stacks (.tif) or MetaMorph specialized TIFF files (.stk):
https://en.wikipedia.org/wiki/Tagged_Image_File_Format
ftp://ftp.meta.moleculardevices.com/support/stack/STK.doc

2. ***Binary fluorescence traces file format (.traces and .rawtraces)***

A specialized binary format is used for saving all trace data to facilitate fast loading and saving of data from disk. All of this information is available in a *Traces* object when loading a file with *loadTraces*.

```
struct TracesFile {
    uint32          zero          = 0
    char[4]         signature     = "TRCS"
    uint16          version       = 5
    uint8           dataType      = enum( 9=single )
    uint8           nChannels     = C, number of data channels (usually 3)
    uint32          nTraces       = N, number of particles (traces)
    uint32          nFrames       = M, number of data points per trace

    uint16          chNameLen     = length of the following char array
    char[*]         chNames       = channel names, delimited by char(31)

    dataType[M]     time          = acquisition time axis (in milliseconds)
    dataType[N×M]   channel1      = donor fluorescence
    dataType[N×M]   channel2      = acceptor fluorescence
    dataType[N×M]   channel3      = FRET

    MetadataField   root          = struct-type metadata field containing
                                    fileMetadata and traceMetadata children.
}
```

dataType is an *enum* (starting with zero): *char, uint8, uint16, uint32, uint64, int8, int16, int32, int64, single, double, logical, cell array, and struct*. Metadata fields have the following structure:

```
struct MetadataField {
    uint8          dataType    = enum of metadata type (see above)
    uint32         fieldSize   = length in bytes of entire metadata page

    uint8          nameLen     = Length of the text string below
    char[*]        fieldName   = tag name for this metadata field

    uint8          ndim        = Number of dimensions listed below
    uint32         dataSize    = List of dimensions of the metadata array

    dataType[…]    contents    = actual metadata content [dataSize] array
}
```

Struct-type `contents` have the following format:

```
struct StructContents {
   uint8          nFields     = Number of fields in the struct

   uint8          nameLen     = Length of the text string below
   char[*]        fieldName   = struct field name
   uint8          isPacked    = true if contents are compressed
   MetadataField  contents    = one element per array element

   …              (last four fields are repeated nFields times)
}
```

If all elements in a field are the same type, they are packed together into a single `MetadataField` element. Char types are packed into one contiguous string delimited by char(31). Numeric and logical types are packed into a matrix concatenated along a new final dimension.

Metadata fields are separated into two groups. “fileMetadata” fields apply to all traces in the file, including a list of wavelengths for each spectral channel (e.g., [532 640]). “traceMetadata” is an array with one value per trace, including unique identifier strings (“ids”) and Cartesian coordinates of PSF locations within the field-of-view for each spectral channel (e.g., “donor_x” and “donor_y”). Identifiers are typically the movie file name and trace number in the original .rawtraces file (e.g., “E:\Folder\test_movie.tif#123”).

3. ***Dwell-times text file format (.dwt)***

FRET idealization traces (state assignment at each point in time) are stored in the *.dwt* format files recognized by QuB. A *.dwt* file is a list of dwell-times grouped into segments that generally correspond to distinct traces. Dwell-time sequences represent an idealization generated using Hidden Markov Modeling algorithms (see **Section VI**). Each segment has the following format:

```
Segment: 1 Dwells: 5 Sampling(ms): 25 Start(ms): 0 ClassCount: N μ1 σ1 μ2 σ2 …
2  150
1  25
2  350
0  75
2  100
```

The first line identifies the beginning of a segment (idealized region, typically the beginning of a trace). Segments are numbered sequentially (first number). The header line includes the number of dwells that follow, the time resolution, the starting point of this segment within the linearized FRET data, and the number of distinct FRET states (classes). The end of the header includes a list of parameters used for idealization: ”μ_1 σ_1 μ_2 σ_2 … μ_N σ_N”, where μ_i and σ_i are the mean and standard deviations of FRET). A list of dwells follows, one per line, with the class number of the state and the duration of the dwell in milliseconds. Note that class numbers start with 0 in the file, but start with 1 in MATLAB (for

example using *loadDWT* and *saveDWT*). Offsets (listed under “Start(ms)”) establish a correspondence between FRET data and the idealization. Also see:
https://qub.mandelics.com/m/DWT.html

4. ***forQuB text format (.qub.txt)***

This text format is used when importing data into QuB. It is a simple list of FRET values, with all traces concatenated together. The traces should then be segmented in QuB.

5. ***QuB_Tree binary format (.qmf, etc.)***

Some information is saved in a binary format that was originally developed for QuB. This includes HMM model files (*.qmf*). The data are represented in a tree of named fields. The format is supported with the *qub_loadTree* and *qub_saveTree* functions. See the following link for further details.
https://qub.mandelics.com/m/qubdoc/

Appendix C. Compiled functions (*.mex* files)

Some functions are written in C/C++ and compiled as binary libraries supported by MATLAB (*.mex* files), generally for speed. Unlike *.m* files that are platform independent, *.mex* files must be compiled for each targeted operating system and computer architecture. These files and any supporting libraries can be found in the "binary" directory in the source code. With the exception of the *qub_loadTree* and *qub_saveTree* functions, each function also has a MATLAB (.m file) fallback function with the same name, but without the 'x' at the end.

1) *qub_loadTree* and *qub_saveTree*: load and save files in the "qubtree" format (e.g., *.qmf* files). The "treestruct.cpp" file contains functions for interfacing between MATLAB and the QuB source code. The necessary QuB source code files for compiling the qubtree library are available in the qubtree subdirectory. https://qub.mandelics.com/sources.html

2) *forward_viterbix*: Viterbi algorithm [5] for finding the optimal state sequence (idealization) given a model. This function is used by the *idealize* and *skm* functions. Source code for this and the remaining functions can be found in the "HMM" folder.

3) *gillespiex*: Gillespie stochastic simulation algorithm used by the *simulate* function.

4) *forwardBackwardx*: compiled implementation of the forward-backward algorithm used to calculate probabilities for Baum-Welch and MPL model optimization algorithms.

References

1. McKinney, S.A., C. Joo, and T. Ha, *Analysis of single-molecule FRET trajectories using hidden Markov modeling.* Biophys J, 2006. **91**(5): p. 1941-51.

2. van de Meent, J.W., et al., *Empirical Bayes methods enable advanced population-level analyses of single-molecule FRET experiments.* Biophys J, 2014. **106**(6): p. 1327-37.

3. Roy, R., S. Hohng, and T. Ha, *A practical guide to single-molecule FRET.* Nat Methods, 2008. **5**(6): p. 507-16.

4. McCann, J.J., et al., *Optimizing methods to recover absolute FRET efficiency from immobilized single molecules.* Biophys J, 2010. **99**(3): p. 961-70.

5. Rabiner, L.R., *A Tutorial on Hidden Markov-Models and Selected Applications in Speech Recognition.* Proceedings of the Ieee, 1989. **77**(2): p. 257-286.

6. Qin, F., *Restoration of single-channel currents using the segmental k-means method based on hidden Markov modeling.* Biophys J, 2004. **86**(3): p. 1488-501.

7. Qin, F., A. Auerbach, and F. Sachs, *A direct optimization approach to hidden Markov modeling for single channel kinetics.* Biophys J, 2000. **79**(4): p. 1915-27.

8. Qin, F., A. Auerbach, and F. Sachs, *Estimating single-channel kinetic parameters from idealized patch-clamp data containing missed events.* Biophysical Journal, 1996. **70**(1): p. 264-280.

9. Bronson, J.E., et al., *Learning rates and states from biophysical time series: a Bayesian approach to model selection and single-molecule FRET data.* Biophys J, 2009. **97**(12): p. 3196-205.