User Manual
BlurLab v0.9 – Fluorescence Microscopy Simulator

Contents

1. Introduction

BlurLab is an easy to use platform for generating simulated fluorescence microscopy data for use in mechanistic model visualization, image comparison, and hypothesis testing. This user manual will explain the program's features and guide you through a detailed example using the software.

This software is released free of charge under a GNU General Public License – you are free to distribute and modify this software and its source code as needed, please see the 'BlurLab_License.txt' file for details.

In publications please cite usage of this software as:

BlurLab v0.9 -- Fluorescence Microscopy Simulator, © Tristan Ursell, Stanford University, 2011.

Current and future releases, manuals, documentation and discussion boards are hosted through SimTK.org (https://simtk.org/home/blurlab), the software framework from the NIH Center for Biomedical Computing.

If you make substantive improvements to BlurLab, please forward them to Tristan Ursell (natsirt@stanford.edu) for incorporation into general software releases.

2. Installation

To start the installation process, go to the SimTk.org (simtk.org/home/blurlab) website to download the installation package, called 'BlurLab_pkg.exe'. The software can be used in two different modes: either through Matlab (recommended), or as a standalone program that does not require Matlab.

2.1 Computing Resources

Image processing can be a computationally intensive operation. By default (although it can be changed), BlurLab keeps all of the simulated output images in RAM, hence we recommend using a computer with at least 1 gigabyte of free RAM and a 2.0 gigahertz or faster processor. In Windows systems, the top of the program window displays the total amount of available RAM ('Total Memory') and largest single block for array RAM ('Array Memory').

2.2 Using BlurLab with Matlab

If you are using Matlab 2009a or higher on any computing platform with a graphical user interface (i.e. not a command line) and the Image Processing Toolbox, it is recommended that you follow these steps.

1. Launch the installer, and when asked to ‘Choose a Setup Language’, click ‘cancel’. This prevents the MCR_Installer from running – BlurLab will be running directly from MatLab sothere is no reason to install the C++ wrapper library.

2. The installer will put all of the necessary files into its current folder. Three directories should appear in the installation folder: (i) ‘BlurLab_m-files’, (ii) ‘documentation’, and (iii) ‘m-file_models’. Keep the m-files in these folders together – if they are separated, BlurLab may not work correctly.

3. Use the ‘…’ key to navigate through Matlab’s ‘Current Folder’ until you reach the ‘BlurLab_m-files’ directory on your computer. Then in the Matlab command line type: ‘addpath(pwd)’, this simply tells Matlab where to look for the files that are needed.

4. Finally, in the Matlab command line, type ‘BlurLab_3D’, this should launch the program through Matlab. A window similar to Figure 1 should appear.

2.3 Using the BlurLab executable

If you do not have Matlab or simply do not wish to use Matlab with BlurLab, you can run BlurLab as a standalone executable with the freely distributed C++ wrapper library, included with this installation.

1. Launch the installer, and when asked to ‘Choose a Setup Language’, click ‘OK’. This will run the MCR_Installer to put a C++ wrapper library on your computer which will allow BlurLab to run.

2. The installer will put all of the necessary files into its current folder. Three directories should appear in the installation folder: (i) ‘BlurLab_m-files’, (ii) ‘documentation’, and (iii) ‘m-file_models’. Keep the m-files in these folders together – if they are separated, BlurLab may not work correctly.

3. Once the installation has finished, there should be a new program called ‘BlurLab.exe’. Double click this icon to launch the program. A window similar to Figure 1 should appear.

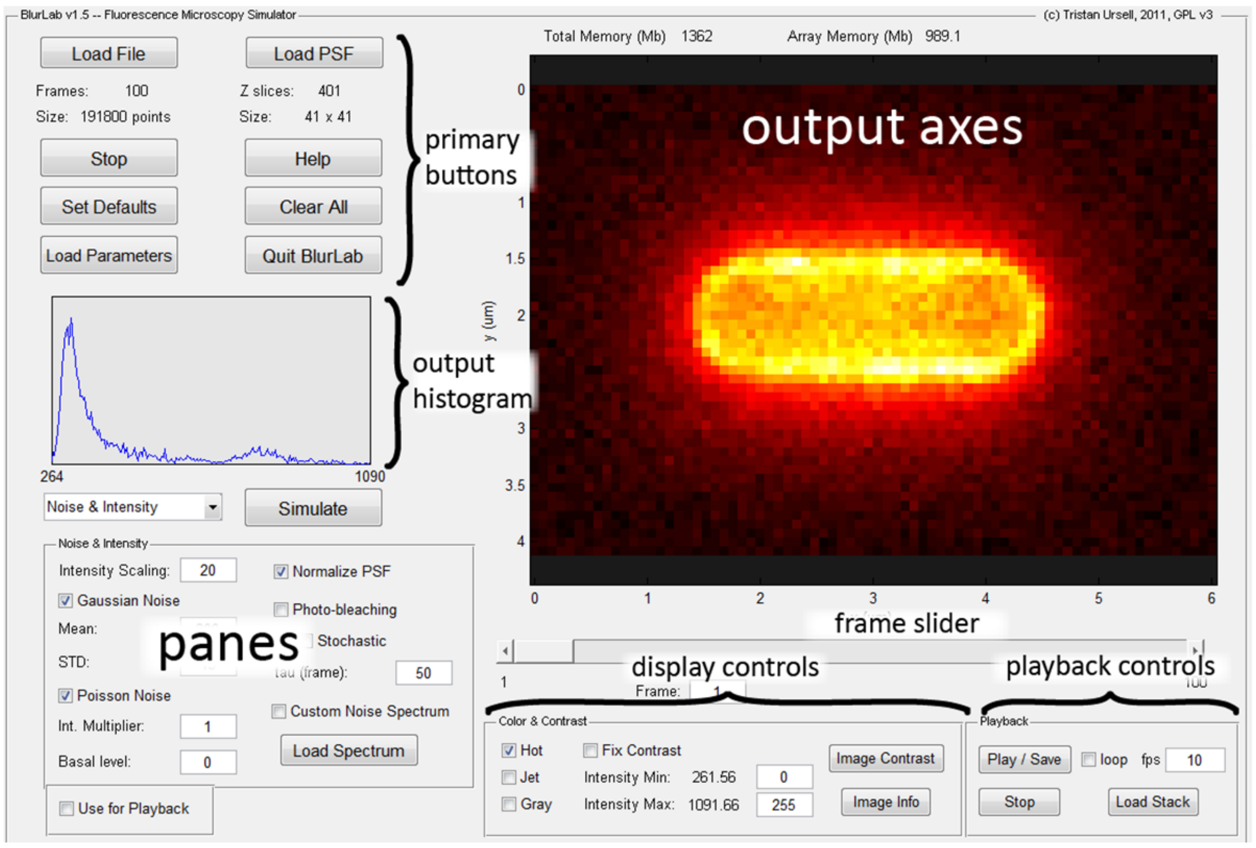
2.4 Optional Components

Additionally, the WCIF ImageJ image processing software* is a useful additional program both for creating the desired point spread functions (PSF) and viewing and manipulating the resulting stacked TIF image files made by BlurLab.

*(http://www.uhnres.utoronto.ca/facilities/wcif/imagej/)

*(http://www.optinav.com/Diffraction-PSF-3D.htm)

By default (although it can be changed), BlurLab keeps all of the simulated output images in RAM, hence we recommend using a computer with at least 1 gigabyte of free RAM and a 2.0 gigahertz or faster processor. At the top of the program window, the total amount of available RAM ('Total Memory') and largest single block for array RAM ('Array Memory') are displayed.



BlurLab v1.5 -- Fluorescence Microscopy Simulator

(c) Tristan Ursell, 2011, GPL v3

Total Memory (Mb) 1362

Array Memory (Mb) 989.1

Frames: 100

Size: 191800 points

Z slices: 401

Size: 41 x 41

Noise & Intensity

Intensity Scaling: 20

☑ Normalize PSF

☑ Gaussian Noise

☐ Photo-bleaching

Mean:

Stochastic

STD:

tau (frame): 50

☑ Poisson Noise

☐ Custom Noise Spectrum

Int. Multiplier: 1

Basal level: 0

☐ Use for Playback

Frame: 1

Color & Contrast

☑ Hot

☐ Jet

☐ Gray

☐ Fix Contrast

Intensity Min: 261.56 0

Intensity Max: 1091.66 255

Playback

☐ loop

fps 10

Figure 1: Program screen-shot with labels showing main button and control groups.

3. Primary Buttons

Upon launching the program, you should see nine primary buttons on the left hand side of the program window: Load File, Load PSF, Simulate, Help, Stop, Clear All, Load Parameters, Set Defaults, and Quit BlurLab (under the output histogram). We will address each button in detail, starting from the simplest button. Please see Figure 1 for a program screen shot showing all nine buttons.

- 'Quit BlurLab' does just that, all currently loaded information will be cleared from memory and the program will close. If one is running the program in MatLab, this will close BlurLab, but will not quit MatLab.

- ‘Clear All’ does the same thing as ‘Quit BlurLab’, except that it automatically re-launches the program, clearing the memory and resetting the program.
- ‘Help’ launches this PDF manual in Adobe Acrobat. Additionally, typing ‘help BlurLab_3D’ into the command line of MatLab, or examining the beginning comments of the ‘BlurLab_3D.m’ file gives a list of all the files that should be included with the installation.
- ‘Set Defaults’ will take the current values of all input and check boxes and save them into a text file ‘BlurLab_Prefs.txt’. Those settings will then automatically be loaded the next time the program starts. If that file is deleted or becomes corrupted, the program will use internal default settings for all the parameters. When a new set of defaults is saved the old ‘BlurLab_Prefs.txt’ file will be over-written. Additionally, a particular configuration of the program can be saved by hitting ‘Set Defaults’ and choosing ‘Not as Default’. This creates a text file with the proper formatting, but with a name of your choosing, to be used with the ‘Load Parameters’ primary button.
- ‘Load Parameters’ loads into the current GUI the text input and check box settings from a properly formatted text file of your choice. Often, after an input file has been simulated and written to disk (see Section 3.1), a parameter file will be created with the same formatting as a preference file from ‘Set Defaults’. Thus if one wants to know or reproduce precisely under what conditions a particular output TIF file was created, simply click the ‘Load Parameters’ button, and select the corresponding parameter text file.
- ‘Stop’ will halt an active simulation and return control to the user interface. All of the images simulated up to that point remain accessible to the user interface.

Keep in mind that in general, across the entire program, button presses are additive – e.g. if ‘Simulate’ is pressed twice in a row, the program will simulate the input data under the current conditions twice in a row.

3.1 Loading Input Files

Before any simulation can take place, two different kinds of files must be loaded: a PSF image stack and an input data file. The example at the end of this tutorial will cover this material.

3.1.1 PSF Files

The ‘Load PSF’ button will bring up a window that allows one to load a stacked TIF file of the point-spread-function (PSF) for the current simulation.   Click on ‘Load PSF’ and navigate into the ‘PSFs’ folder, and select any one of the available PSF image stacks. The program will ask that the file be calibrated – for now, accept the default values and directly below the ‘Load PSF’ button the size and number of images in the PSF stack should be displayed. This indicates that the program has accepted the PSF file as input.

The installation package comes with three properly formatted wide-field PSFs, with the following description:

	PSF_1	PSF_2	PSF_3
effective magnification	20X	100X	100X
numerical aperture	0.7	1.4	1.4
index of refraction	1.515	1.515	1.515
wavelength (nm)	532	532	532
Z spacing (nm / slice)	40	40	10
X-Y spacing (nm / pixel)	400	80	80
Z extent (um)	-10 to 10	-10 to 10	-2 to 2

Table 1: Optical and pixel properties of the point spread functions (PSFs) included with BlurLab.

When creating one's own PSF image stack, there are a number of things to keep in mind. Generally, the program accepts PSFs for wide-field or confocal fluorescence microscopy (or any point emission source). The program expects this file to have certain symmetry properties; each image in the PSF stack should be square, such that the center of the PSF is in the X-Y center of the image. If the image is not square, the program will not accept the image file. The Z-center of the PSF should be in the central frame of the image stack. If the center of the PSF is not in the center of the image stack, the program will still run, but the X, Y, and Z position of the focal plane will be translated or unevenly illuminated and hence inaccurate. Other PSFs can be created using the 'Diffraction PSF 3D' plug-in for NIH ImageJ software (freely available).

Generally, it is a good idea to over-sample the PSF – the X-Y spacing of the pixels should be smaller than the wavelength of light by less than or approximately equal to a factor of four, and the Z-spacing of the PSF should be about the same. If one is trying to simulate the output for a specific camera pixel resolution, it is easiest to match the PSF resolution to the desired pixel resolution. Alternatively, one can use bi-linear scaling to change the X-Y pixel resolution of the PSF to match that of the detector, as implemented under 'Scale' in the 'Image' drop down menu of ImageJ. Once the PSF file is loaded, information will appear below the 'Load PSF' button indicating how many Z-slices are in the image stack, and the size of each X-Y image.

3.1.2 Input Files

BlurLab accepts three kinds of input data files. The preferred file format for BlurLab is a text file organized by columns. The program can accept either a 5 column or a 6 column format.

The first three columns are the X, Y and Z positions of the emitters, respectively, in either nanometers or microns (the program will query which units are appropriate). The fourth column is the intensity of the emitters at those positions – all intensities must be positive. The fifth column is the frame number; all the emitters with the same frame number will appear together in the same image, thus the frame number is akin to time in a normal image stack collected from a microscope (although BlurLab has no explicit notion of time). The sixth column assigns a unique label to each emitter across all frames, such that the label identifies the same emitter in different frames.

Thus each row of the input text file specifies the three dimensional location, intensity, and frame of each and every emitter. The sixth column of the input text file can be neglected, however stochastic photobleaching and FRAP will not be available – the labels are required for these processes.

	X position	Y position	Z position	Intensity	Frame	Label
Line 1	x_1	y_1	z_1	1	1	1
Line 2	x_2	y_2	z_2	1	1	2
Line 3	x_3	y_3	z_3	1	1	3
…	…	…	…	…	…	…

Table 2: Table showing the basic structure of a correctly formatted six column input file. In this case, the file shows the first three emitters on the first frame, for use in single molecule analysis (i.e. all intensities set to 1).

There are a number of ways in which the text input file can be improperly formatted. If the input text file does not have 5 or 6 columns, the program will give a warning message and not accept the file as input. The frame numbers must start from 1 and increase sequentially by whole numbers. If the program detects that the input file has missing or non-consecutive frames, a warning will appear, and the input file will not load. Likewise, it is a good practice, although not required, to label unique emitters sequentially starting from 1. To model single molecules the intensities should all be equal; whereas larger groups of diffraction-limited emitters may have varying intensity values, and in general the intensities and positions are free to change from frame to frame.

To use either stochastic photobleaching and / or FRAP, the input file must include a sixth column specifying a label for the emitters in each frame. The labels should be sequential starting from 1. For instance, if the first simulated frame has 5 emitters, labeled 1 to 5, and the second frame introduces 3 new emitters, they should be labeled, 6, 7 and 8. Not all labels need be in each frame. Without this label, the program would not know which emitters to keep dark once they have been bleached.

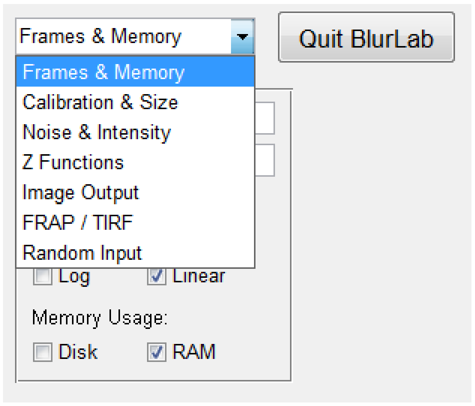
While stochastic photobleaching and FRAP each have a simple user interface control within BlurLab, advanced users may want to simply input a data file that turns emitters on or off at the desired frames in the desired locations by changing the intensity directly in the input file – this allows for the highest level of control, and hence does not require a label column.

The simplest, but least precise type of input data are stacked TIF image files. The program interprets each non-zero pixel as the position of an emitter with the X-Y position corresponding to the pixel location, and the pixel spatial calibration set by the 'Image Pixels' input in the 'Calibration & Size' panel (see Section 4). When a TIF file is selected the user is asked to calibrate how many images in the stack comprise a single image frame, or in other words, how many Z slices are in each time frame of the input TIF file. The program will also query the user to calibrate the Z slice spacing. Once these parameters are set, the program will read in the TIF image stack, *create* a corresponding 5 column 'unlabeled' input text file (see below), and then load that input text file. The newly created text file appears in the same directory as the original TIF file. The output image size is calculated from parameters set in the 'Calibration & Size' pane.

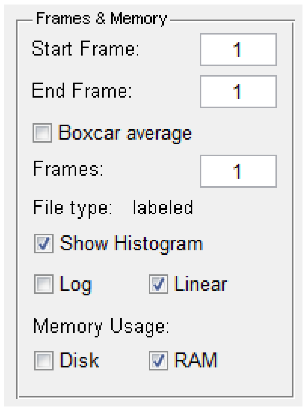
After the input file has loaded, data will appear below the 'Load File' button indicating how many frames are in the input file and how many emitters will be simulated from those frames in total.

4. Panes

To the left of the 'Quit BlurLab' button is a drop-down menu, as shown below, from which one can select the different user control panes of the program. In this section, we will explore each of these panes.



4.1 Frames & Memory



Frames & Memory

Start Frame: 1

End Frame: 1

☐ Boxcar average

Frames: 1

File type: labeled

☑ Show Histogram

☐ Log ☑ Linear

Memory Usage:

☐ Disk ☑ RAM

Each input file loaded into BlurLab has some number of frames in which the emitters will appear. The 'Start Frame' is automatically set to 1 (initially) and the 'End Frame' is automatically set to the number of frames detected in the input file. However, one may not want to process the entire input file and hence one can select a subset of frames to process. When the input file is loaded, the program will automatically detect if the file has labeled emitters – if it does, the phrase 'labeled' will appear to the right of 'File type', else 'unlabeled' will appear. The 'Show Histogram' checkbox will toggle the histogram display on and off just above the 'Quit BlurLab' button. The X axis of the histogram automatically adjusts to the current image intensities. The Y axis can be displayed in either 'Log' or 'Linear' scaling.

One can simulate a dynamic process, where motion blur is important over the time scale of image exposure, by simulating the process at a higher temporal resolution and then using Boxcar averaging to down-sample in a manner that is dynamically accurate. Selecting 'Boxcar average' will perform this averaging over the number of frames specified in the 'Frames' field. For instance, if one wants to simulate an image with an exposure time Δt, and the input file has frames spaced by a time dt, such that $\Delta t/dt = n$, then the 'Frames' field should be set to n. When 'Boxcar average' is selected, noise is only applied to the Boxcar averaged image since this is the image that would effectively reach the detector in the real physical system.

Finally, the simulated images the program generates can either be kept in RAM or written to a temporary file on the hard disk called 'BlurLab_temp.tif'. The 'RAM' memory usage is usually faster than 'Disk', however, depending on the size of the input file and the number of frames to be processed, your computer may not be able to allocate enough memory to simulate in RAM. RAM allocation can be an issue on 32-bit systems, but is almost never an issue on 64-bit systems. If RAM allocation becomes an issue, the program will give a warning and automatically switch to 'Disk' mode. When the program starts or quits, it will look for any lingering temporary image files in the home (BlurLab_3D.m or BlurLab.exe) directory and erase them. Do not erase temporary image files while the program is running as this may crash the program.

4.2 Calibration & Size

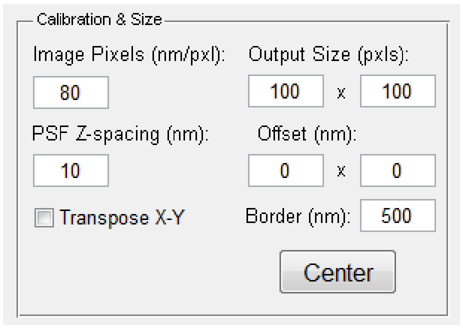
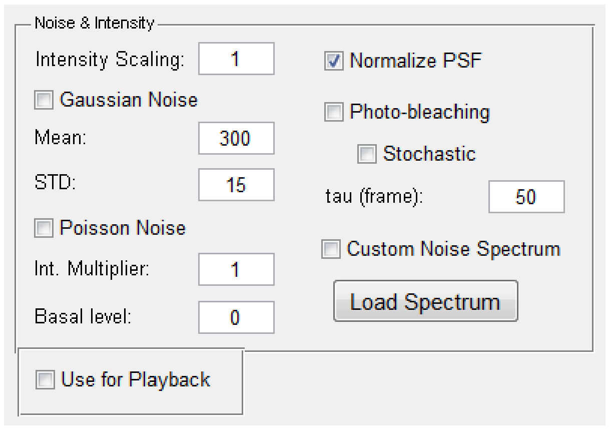


Image Pixels (nm/pxl): 80
Output Size (pxls): 100 x 100
PSF Z-spacing (nm): 10
Offset (nm): 0 x 0
Transpose X-Y
Border (nm): 500

This pane is used to set the scaling, orientation, and size of the output images. Upon loading a PSF image stack, the program will prompt the user for the proper 'Image Pixels' and 'PSF Z-spacing' scaling. The 'Image Pixels' scaling sets the X-Y scaling for both the input PSF file and the output images. In a real microscope image this would be set by the pixel size of the detector divided by the magnification of the optical system. The 'PSF Z-spacing' sets the distance between consecutive images in the PSF Z-stack.

When an input file is loaded, the program examines the extent of emitter positions in X and Y across all frames and centers this cloud of emitters in the output image plane with a border specified by 'Border'. Hitting 'Center' will automatically calculate the proper 'Output Size' and 'Offset' to center the emitters in the output image, but those options may be changed as desired. The 'Output Size' can be modified, but take care not to make the image smaller than the X-Y extent of emitter positions, as this will result in a program error. Likewise, the 'Offset' can be used to translate the emitters in the X-Y image plane. The program will generate an error message if the positions of any of the emitters, in either X or Y, become negative or lie outside of the extent of the output image. The 'Transpose Image X-Y' checkbox will transpose (i.e. rotate plus reflect) the output image – there is no need to adjust the 'Output Size' or 'Offset' when transposing, the program will internally make the necessary adjustments.

4.3 Noise & Intensity


Noise & Intensity

Intensity Scaling: 1

Gaussian Noise [ ]

Mean: 300

STD: 15

Poisson Noise [ ]

Int. Multiplier: 1

Basal level: 0

Normalize PSF [✓]

Photo-bleaching [ ]

Stochastic [ ]

tau (frame): 50

Custom Noise Spectrum [ ]

Use for Playback [ ]

This pane allows one to introduce noise, photobleaching and intensity scaling into the output images. As such, this is the most complicated control pane.

We recommend leaving ‘Normalize PSF’ checked. This option scales the input PSF so that its maximum value (which should be at the exact center of the PSF image stack) is equal to one. Thus, if this box is checked, regardless of the scale of the input PSF file, if an emitter has an intensity of one in the input file, the corresponding Airy disc after convolution with the normalized PSF will have a maximum of one. In the unlikely case that you are using an input file and PSF file that have a specific relationship in to each other in intensity space, you may want to uncheck this.

The ‘Intensity Scaling’ parameter simply multiplies the image intensities by a constant, positive factor. If, for instance, one wants to adjust the intensities of all the emitters to better match some desired noise spectrum, one can do so by adjusting this parameter. If the ‘Intensity Scaling’ is set to any other value than 1, it will blink at the end of the simulation to alert the user that any subsequent playback may reapply the scaling factor, leading to spurious results.

Selecting ‘Gaussian Noise’ adds random Gaussian noise (e.g. thermal or read noise) at each pixel with the specified ‘Mean’ and ‘STD’. This noise will be different for every frame and for each instance of simulation. These parameters can be easily measured from one’s detector by choosing the relevant camera settings and exposure time, and collecting images of the background noise.

When ‘Poisson Noise’ is selected, the intensities at each pixel of the output image, after convolution, photobleaching, and / or Boxcar averaging, but before Gaussian noise addition, are now generated by a Poisson random variable whose seed is a function of the original intensity from the convolution and the settings in this pane. If the ‘Int. Multiplier’ is IM, the ‘Basal Level’ is BL, and the intensity at the current pixel after intensity scaling, convolution, photobleaching, and Boxcar averaging is $I(x, y)$, then the Poisson parameter for that pixel is given by $\lambda(x, y) = IM \times I(x, y) + BL$. It is important to realize that with 'Poisson Noise' checked, the intensity at each pixel *is* the Poisson random variable with seed $\lambda(x, y)$. The 'Basal Level' is equivalent to setting a background level of fluorescence in the sample. Keep in mind that changing 'Intensity Scaling' will have an effect on the Poisson parameter as it will augment the intensities of the emitters before convolution with the PSF.

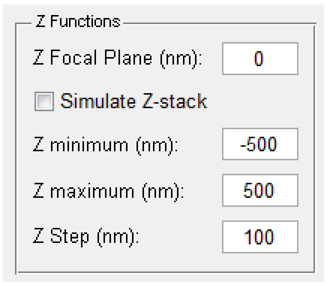
Thus, for instance, if one knows that background noise in dark conditions is 300 counts with a standard deviations of 15 counts, and that the object being simulated should at its brightest point be registering approximately 1500 counts, one needs to adjust the 'Intensity Scaling' to brighten the image to the appropriate level given these noise estimates. Then, depending on your camera, that number of counts corresponds to the arrival of a certain number of photons per exposure – often there is approximately a one-to-one correspondence between pixel counts (and hence electrons on the CCD) and the number of photons that arrive during the exposure, thus leaving the Poisson 'Int. Multiplier' set to one is often appropriate. For electron multiplied cameras that have a separate gain setting, the 'Int. Multiplier' is usually, simply the gain itself (i.e. the gain is multiplicative constant on the photon-to-electron conversion before the detector chip is read).

Checking 'Photo-bleaching' without 'Stochastic' simply multiplies the intensities in each frame by an exponential weighting, $\mathrm{e}^{-i/\tau}$, where i is the frame number starting from 'Start Frame' in the *Frames & Memory* pane. This mode of photobleaching is appropriate when the emitters in the input file are not meant to represent single molecules, but larger fluorescent emitters with many fluorophores. If the emitters are meant to represent single molecules then a labeled input file should be used, and the 'Stochastic' option should be checked. This will stochastically bleach individual emitters using an exponential time distribution with the time constant 'tau', where 'tau' is not measured in time, but in frame number. If one tries to use stochastic photobleaching without a labeled input file, the program will proceed, but will automatically uncheck 'Stochastic'.

While many thermal or read noise spectra can be simulated as Gaussian distributions, a specific camera may have a slightly different noise distribution. Likewise, it may be of interest to simulate images with non-standard noise spectra. The 'Custom Noise Spectrum' option allows for these possibilities. Before checking this box, use the 'Load Spectrum' button to load an input noise file. The program accepts two different kinds of input noise files. The first file type is a TIF image stack – created, for instance, by simply taking dark frames at the appropriate exposure and digitization rate from an actual camera, so as to simulate the scenario of interest from an exact camera – the more frames one collects, the more accurate the reproduction of the noise spectrum of that device will be. Once the TIF noise image stack is loaded, the program will generate a noise spectrum from the input images and use that exact spectrum to generate random noise in the simulated images. When the TIF stack is loaded, the program will automatically write an output text-based noise file with the name 'text_noise_TIFname_date.txt' and with the following internal format. In the same fashion, the program will also accept an input noise text file with two columns; the first column being the intensity, the second column being the frequency or probability density of those intensities. The program will automatically normalize the input noise distribution. When 'Custom Noise Spectrum' is selected the program will automatically deselect 'Gaussian Noise', and *vice versa*.

All of the settings in the *Noise & Intensity* pane will be used when the 'Simulate' button is pressed, however, when the simulated frames are played back (using the 'Play / Save' button in the *Playback* pane – see Section 7), these settings are *not* applied unless the 'Use for Playback' box is selected. This prevents one from simulating an input file with some specified noise settings and then unintentionally applying those same noise characteristics to the output images a second time if 'Play / Save' is pressed. This also means that one can simulate an input file without any noise, and then apply different noise spectra to the image during playback without having the run the simulation over again. One can apply any of the types of noise discussed above to the playback, except stochastic photobleaching.

4.4 Z Functions


Z Functions

Z Focal Plane (nm): 0

☐ Simulate Z-stack

Z minimum (nm): -500

Z maximum (nm): 500

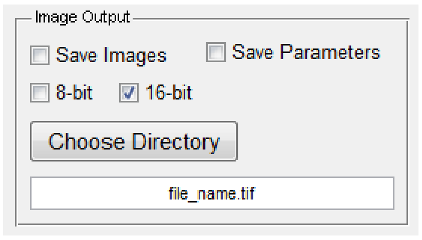
Z Step (nm): 100

This pane allows the user to adjust how the image is viewed in the Z direction. When the input PSF file is loaded, the 'PSF Z-spacing' and number of images in the PSF stack set the extent in Z over which the program can accurately simulate the convolution of a point emitter. For most cases, at a given focal plane, light coming from emitters multiple wavelengths away from the focal plane is very dim and diffuse, and hence does not contribute meaningful light to the image. Thus it is a good idea to set the size of the PSF to be multiple wavelengths above and below the focal plane. For instance, if one is simulating using 532 nm light with a Z-slice spacing of 40 nm, it would be appropriate for this PSF stack to have at least ~130 Z-slices.

The program aligns the center of the PSF with each emitter – effectively creating a list of where to place the PSF in Z (and X and Y) for each emitter in each frame. When 'Simulate Z-stack' is not checked, the 'Z Focal Plane' option sets the absolute position of the focal plane, with absolute position of the emitters set by the Z column of the input file. For instance, if one is simulating emitters on a sphere, and X=Y=Z=0 is the sphere's center, then setting 'Z Focal Plane' to '0' puts the focal plane at the mid-plane of the sphere. The program will ignore emitters that are out of range of the PSF, but generally, this is not a problem if the PSF stack extends multiple wavelengths from the PSF center. For instance, let us assume that the sphere of emitters in question is 1 um in radius, and that the loaded PSF stack has a total Z-height of 2 microns, that is, one micron above and one micron below the PSF center. If the 'Z Focal Plane' is set to 1000 nm, 1 micron above the sphere's center, all the emitters below the Z=0 plane will be ignored because they do not fall within range of the loaded PSF at the current focal plane. Again, this will not significantly impact the results as those emitters would not contribute much structured light to the output image, but it will affect the amount of diffuse light that appears in the output image. However, all of this can be avoided by using a PSF whose Z height is matched to the full span of emitter Z positions.

The same principles apply to the 'Simulate Z-stack' option. Checking 'Simulate Z-stack' disables the 'Z Focal Plane' option, and the program will then simulate a unique focal plane from 'Z minimum' to 'Z maximum' with the number of distinct slices set by 'Z Slices', for each frame of the input data. In this case the output data, and resulting output TIF file, is organized first by frame and then by Z slice number. For instance, simulating 2 frames with 3 focal planes in each frame, the output frame order would be (1,1), (1,2), (1,3), (2,1), (2,2), (2,3) where the first number is the frame and the second number is the focal plane number.

4.5 Image Output



The program can output two types of files. The first and most important is an output TIF file of the simulation of interest. The second file type is a text based meta-file that contains all of the settings used to create its sibling TIF file.

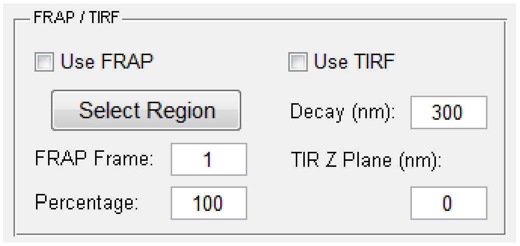
When an input file is loaded, a name is automatically generated for output files with format 'output_date_input-file-name' of either TIF or text extension for each of the respective output file types. The user can modify this name as desired. If the program detects that a file with the same name exists in the current directory, it will append '_n#' to the file name, where # is set by which files are currently in the directory.

Checking 'Save Images' will automatically check 'Save Parameters'. When 'Save Parameters' is checked, the program creates a text file with the aforementioned name structure that contains the values of all program parameters, in the same format as a file created when the 'Set Defaults' button is pressed. Thus, for each simulated image stack created by the program, 'Save Parameters' records the precise parameters used to create it. Pressing 'Load Parameters' allows one to load a previously generated set of parameters. If you do not wish to create a unique parameter file for each output image stack, simply uncheck 'Save Parameters'; this may be useful if, for instance, many distinct input files are being processed with the exact same set of parameters.

Finally, the output images can be saved, keeping the intensity of the output images in mind, in either an 8-bit or 16-bit format. Recall that 8-bit is positive integer values from 0 to 255; 16-bit is positive integer values from 0 to 65535. Any values above the bit range will be truncated to the maximum bit value (saturated), and that information will be lost. Note that most modern cameras do not go above 14 or 16 bit intensity values.

The 'Choose Directory' button allows one to choose where the output files will be created. If no explicit choice is made then the files will appear in the directory where the input file is located.

4.6 FRAP & TIRF

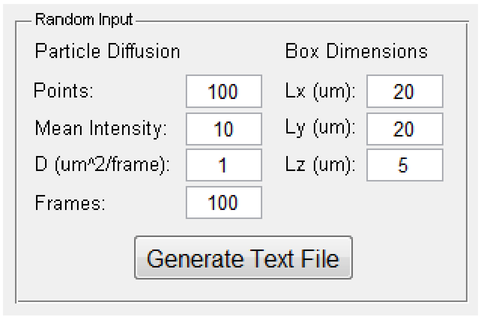


Decay (nm): 300
FRAP Frame: 1
TIR Z Plane (nm): 0
Percentage: 100

This pane allows the user to apply rectangular FRAP or TIRF imaging to the simulation. To engage FRAP, load a *labeled* input file, and use the *Frames & Memory* pane to simulate a single frame that shows the object and scene you wish to FRAP. With that image on the output axes, switch to the *FRAP & TIRF* pane from the drop down menu and push 'Select Region'. The program will then give the opportunity to draw a rectangular box around any region of the image. The program will show the selected region and verify that you are satisfied with the selection. Only after selecting a region will the 'Use FRAP' checkbox be available. The 'FRAP Frame' setting dictates at which absolute frame number the FRAP will occur. The 'FRAP Frame' must be equal to or greater than the 'Start Frame' in the *Frames & Memory* pane – if it is not, the program will automatically set the 'FRAP frame' as the first simulated frame. The 'Percentage' value sets the percentage of emitters in the selected region that will be stochastically bleached. This is akin to adjusting the FRAP laser power or FRAP time duration during a real experiment.

The ‘Use TIRF’ setting applies an exponential weighting function to the PSF in the Z direction with a decay length set by ‘Decay’. In real experiments, the Z plane from which the evanescent wave front that causes TIRF illumination emanates is fixed by the position of the glass-aqueous interface, and hence does not depend on the focal plane, *per se*. Similarly, the program needs to know from which Z plane the evanescent field emanates. When an input file is loaded and the ‘Use TIRF’ box is checked, the program automatically sets the ‘TIR Z Plane’ to be the minimum Z emitter value detected in the input text file, that is, the bottom of the simulated object. This value can be set lower (i.e. that evanescent field can emanate from a point farther below the sample), but any value above the minimum Z emitter value in the input text file will not be accepted by the program.

4.7 Random Input


Random Input

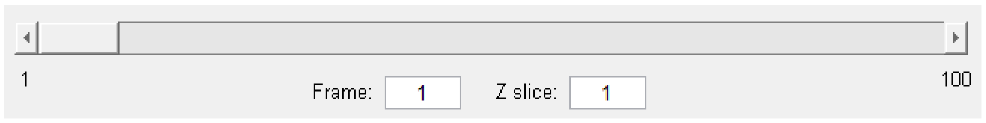
Particle Diffusion		Box Dimensions	
Points:	100	Lx (um):	20
Mean Intensity:	10	Ly (um):	20
D (um^2/frame):	1	Lz (um):	5
Frames:	100		

This feature is a relatively simple test-bed for noise and other program features. In this mode the program still requires a PSF input file, but does not require an input data file. The program will generate a labeled text file with particles of distinct, randomly chosen intensities when the ‘Generate Text File’ button is pressed. The text file has the name structure ‘BlurLab_rand_output_date_time_parameters.txt’. After being created, this file must be loaded as a regular input file. The number of particles is set by ‘Points’, and their Poisson distributed intensities are set by ‘Mean Intensity’. The number of frames to be simulated and the effective diffusion coefficient of the emitters are set by ‘Frames’ and ‘D’, respectively. The particles are confined to diffuse inside a box whose dimensions are set by ‘Lx’, ‘Ly’ and ‘Lz’.

5. Display Controls

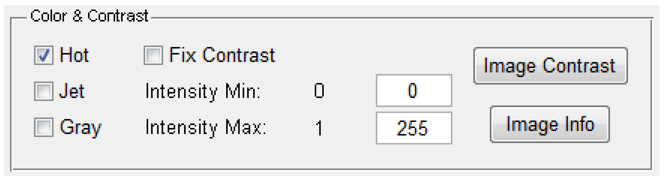
To the right of the **Primary Buttons** and the **Panes** are the output display axes where all simulated images and play back occurs. These axes are always displayed in microns, with the X direction increasing to the right, and the Y direction increasing down the image – this is the standard orientation of the axes for image manipulation.

5.1 Frame Slider



When the user sets the frames to be simulated in the *Frames & Memory* pane, or when a file is loaded for playback, the slider bar will be set to the appropriate bounds. Before an input file is loaded, moving the slider bar will result in a warning. After simulation, dragging the slider bar moves through the simulated image stack. You can also type the desired frame number in the 'Frame' field, to skip to the frame of interest. If the 'Simulate Z-stack' box is selected, a second 'Z slice' input box will appear to the right of the 'Frame' box. One can skip to a desired frame and Z slice by adjusting these inputs. Additionally, using the right and left arrow keys will move sequentially through the current stack in memory.

5.2 Color & Contrast



The program natively uses three different color maps for images shown in the output display axes. Each pixel stores an intensity value and depending on the color map, that intensity value can be mapped to an RGB color representation. The three color maps in use are:

'Hot' – modeled after a black-body-like spectrum

'Jet' – a standard rainbow-styled color map

'Gray' – the standard grayscale color map

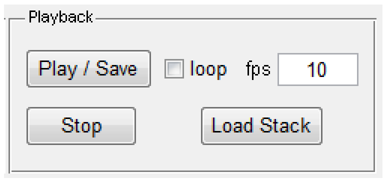
Selecting one of these will automatically deselect the others.

The program keeps track of the minimum and maximum intensity values in each image of the simulated stack, and over the stack as a whole. As frames are simulated or played back, the intensity minimum and maximum of the current image will be shown in the 'Intensity Min' and 'Intensity Max' text output, respectively. When 'Fix Contrast' is selected, the input boxes to the right of these text outputs allow one to set the contrast of the image. Upon clicking 'Fix Contrast', the program will automatically set the 'Intensity Min' text input to the minimum value in the image stack, and the 'Intensity Max' text input to the maximum value in the image stack. One is free to change these values, but clicking 'Fix Contrast' again will reset the minimum and maximum text input values to the stack extrema.

The 'Image Contrast' button can be used with any image in the stack, and brings up an interactive contrast window showing the histogram of pixel intensities in the current output display axes. From there, one can adjust the brightness and contrast as desired, and use the values determined in this mode to set the appropriate 'Fix Contrast' minimum and maximum intensities. One can also use the 'Eliminate Outliers' option in the image contrast window to trim the histogram by a fixed percentage. While one can use this window to find suitable values for contrasting, the contrast settings in no way change the data stored in or saved from the image stack.

The 'Image Info' function can be used to get detailed information on a local pixel region. When the 'Image Info' button is pressed, a new window will open showing pixel intensities numerically mapped onto a zoomed-in version of the output display axes. Dragging with the mouse pans over the image in the output display axes. Zoom-in and zoom-out functions are located on the upper left of the window, and the current address and intensity of the pixel under the mouse is shown in the lower left of the window. One can also print the current axes, in a variety of formats, from this window.

6. Loading and Playing Stacks



After an input file has been simulated, it remains in memory (either RAM or a temporary disk file) until the program is closed. This allows one to continue manipulating the simulated data, for instance by adding noise or adjusting contrast. Pressing 'Play / Save' in the *Playback* pane displays the stack images sequentially, including any z-slices, approximately at the speed set by the frames-per-second ('fps'). The speed of playback depends on the speed of the computer and if additional noise is being applied during playback. The maximum frame rate is usually about 10 frames per second. Hitting 'Stop' at any point during the playback will stop playback and return control to the user interface. Hitting 'Play / Save' again, will start playing from the current frame on the slider. Selecting 'loop' will loop the playback until either 'Stop' is pressed or the 'loop' box is deselected.

An advantageous way to use the playback feature is to simulate an input file, with, for instance, 'Poisson Noise', and then use the 'Gaussian Noise' feature to test out different background noise levels on the same set of simulated images. Once a suitable set of noise characteristics have been found, one can check 'Save Images' in the *Image Output* pane and hit 'Play / Save' again to save the playback with the current settings in a TIF file with the given output name. The program will ask to make sure you want to save the playback, for instance, in case you left this boxed checked after saving a previous simulation.

Pushing the 'Load Stack' button will clear the current image stack out of memory and load the selected TIF file. To this file one can add Gaussian noise or Poisson noise, apply photobleaching, or adjust contrast, and then re-save the images, without loading an input or PSF file. None of the other features (such as TIRF, FRAP, or image resizing) will work when simply loading a TIF file through 'Load Stack'.

7. Detailed Example: Diffusing Protein on a Spherical Surface

In this section we will work through a short but detailed example, explaining how to create different kinds of output images from an input text file. We will use a relatively simple example of proteins diffusing on the surface of a small sphere. You can modify this example by changing the relevant parameters in 'BlurLab_sphere_example.m' (see section 9), and creating a new input text file. We will go through the image creation process using BlurLab in a step-by-step fashion.

First, let's ensure you have all the files needed to run this example. In the directory where you installed BlurLab, using the 'BlurLab_pkg.exe' installer, there should be a folder called 'PSFs' and in that folder should be a file that starts with 'PSF_2…' (see Table 1 for a description of this PSF). This is the PSF for a 100X objective with a numerical aperture of 1.4, a continuous index of refraction of 1.515, using an emission wavelength of 532 nanometers, with a z-spacing of 40 nanometers, and the detector has a pixel calibration in X-Y of 80 nanometers.

There should also be a folder called 'm-file_models' which contains an input text file called 'example_spherical_diffusion_R-2_rho-200_dx-0.283.txt'. This file simulates diffusion on a spherical surface with a radius of 2 microns, an average density of 200 particles per micron squared, and an average Gaussian jump distance of 283 nanometers between frames (to simulate diffusion). Note that this jump distance is about half the wavelength of light being used, thus the temporal resolution of this input file is high enough to perform motion blur through Boxcar averaging. Though the program has no explicit notion of time, it will be useful to know that the time between frames in this example is 20 milliseconds and this file simulates a total of 10 seconds of diffusion on a sphere. Each particle has the same intensity and hence this file is appropriate for simulating single molecules. Given these facts, a quick back of the envelope calculation indicates that this file should have around 5 million lines of input – which is quite a bit, this is a detailed simulation.

We will explore how this object appears through the given PSF under many distinct conditions, starting from the simplest conditions of wide-field fluorescence up to the most complicated conditions of FRAP. We recommend that you shut down all other programs before you start using BlurLab as some of these processes are computationally intensive and the input file has, literally, millions of emitters of which the program must keep track.

Let us begin …

1. Launch BlurLab either from the executable or, preferably, through Matlab as described in section 2. Load the PSF file described above with the 'Load PSF' button. In the dialog box that pops up, in the pixel calibration field enter '80' and in the Z-spacing field enter '40'. Then press 'OK'. Below the 'Load PSF' button, it should show that there are 501 Z slices each with a size of 100 x 100 pixels. Thus this PSF is 8(x) x 8(y) x 20(z) microns in size.

2. Load the input text file described above with the 'Load File' button. Select 'microns' in the pop-up window 'Text Input Units'. Since we are simulating full molecular detail, this file is quite large (~250Mb), and depending on your computer may take a minute or two to load – please be patient. The 'Loading text file …' pop-up window will automatically disappear when the file has finished loading. Below the 'Load File' button, the program should show that it found 500 frames, and a total of 5,098,500 lines in the input file, where each line corresponds to the position of a unique point emitter found within the frames of the simulation.

3. Let us assume that at first, we just want to see different views of this object in a static fashion, as if it were a fixed sample.

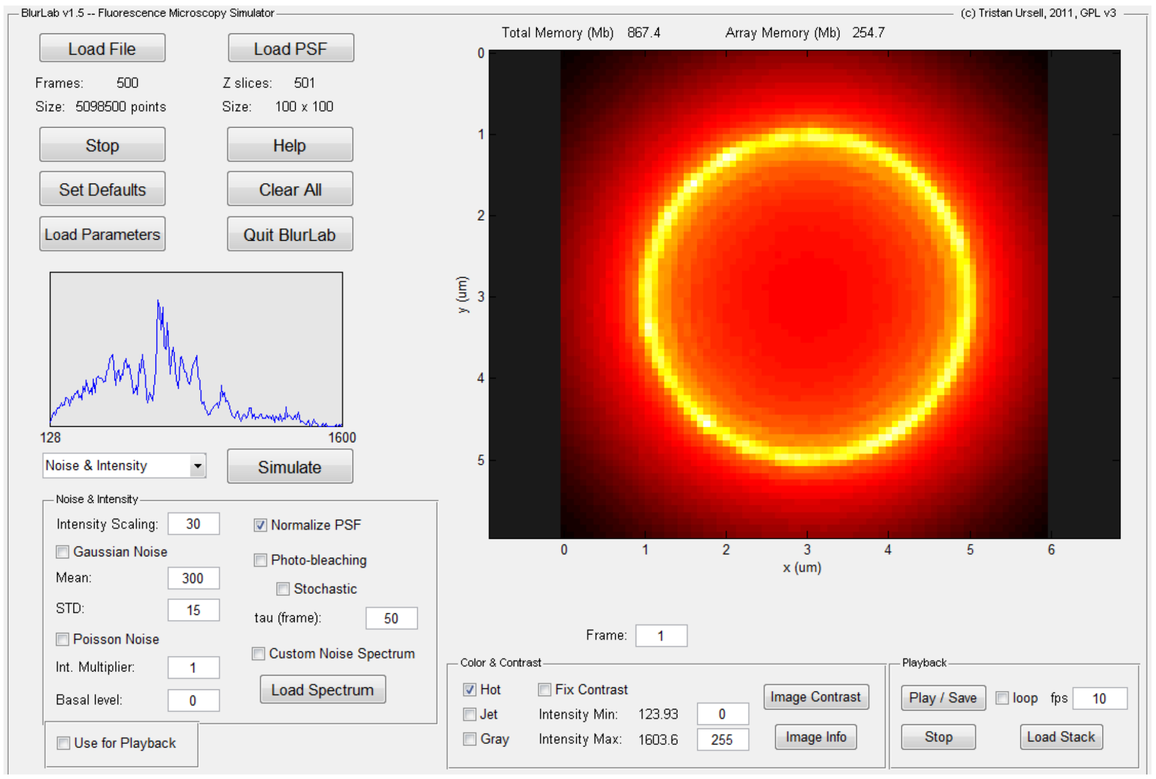
   -- Go to the 'Frames & Memory' pane and enter '1' in the 'Start Frame' field and '1' in the 'End Frame' field.

   When you do this, the display slider under the output axes will disappear as there is only one frame being simulated, and hence no need for a slider.

4. -- Now go to the ‘Calibration & Size’ pane.
   -- The ‘Image Pixels’ and ‘PSF Z-spacing’ should already be set to ‘80’ and ’40’, respectively, from step 1.
   -- Enter ‘1000’ into the ‘Border’ field and hit the ‘Center’ button.

   The output sizes should now read ‘75’ and ‘75’, and the offsets should both read ‘3000’ – in this case the numbers are the same because we are simulating a sphere, which is symmetrical.

5. Now switch to the ‘Noise & Intensity’ pane and enter ‘30’ into the ‘Intensity Scaling’ field. This will set the output intensity to an appropriate range for a typical fluorescent object. Hit the ‘Simulate’ button below ‘Load File’. The program will load the appropriate frame and align the PSF information to the emitters in the input file. The output should appear on the right hand side of the program window, and the program window should look very similar to:



BlurLab v1.5 -- Fluorescence Microscopy Simulator
(c) Tristan Ursell, 2011, GPL v3

Total Memory (Mb) 867.4
Array Memory (Mb) 254.7

Frames: 500
Size: 5098500 points
Z slices: 501
Size: 100 x 100

Noise & Intensity

Noise & Intensity
Intensity Scaling: 30
[✓] Normalize PSF
[ ] Gaussian Noise
Mean: 300
STD: 15
[ ] Photo-bleaching
[ ] Stochastic
tau (frame): 50
[ ] Poisson Noise
Int. Multiplier: 1
Basal level: 0
[ ] Custom Noise Spectrum
[ ] Use for Playback

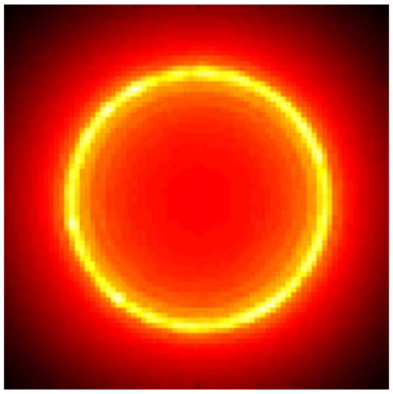
Frame: 1

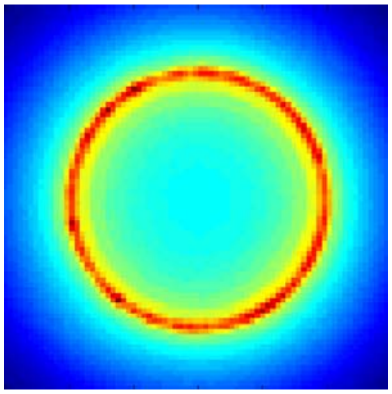
Color & Contrast
[✓] Hot
[ ] Jet
[ ] Gray
[ ] Fix Contrast
Intensity Min: 123.93 0
Intensity Max: 1603.6 255

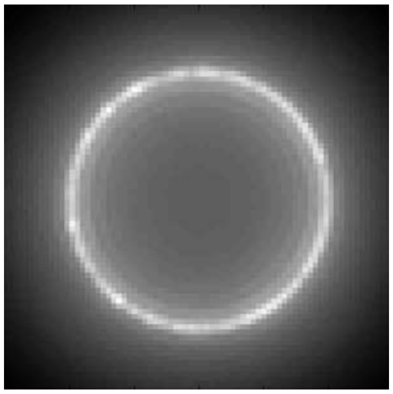
Playback
[ ] loop fps 10

This is what the sphere would look like at its midplane with a very short time exposure (20 ms) if there were no detector or Poisson (shot) noise in the image.

6. Now let us get an idea of what the display controls do. Cycle through the different colormaps to get an idea of what each one looks like:

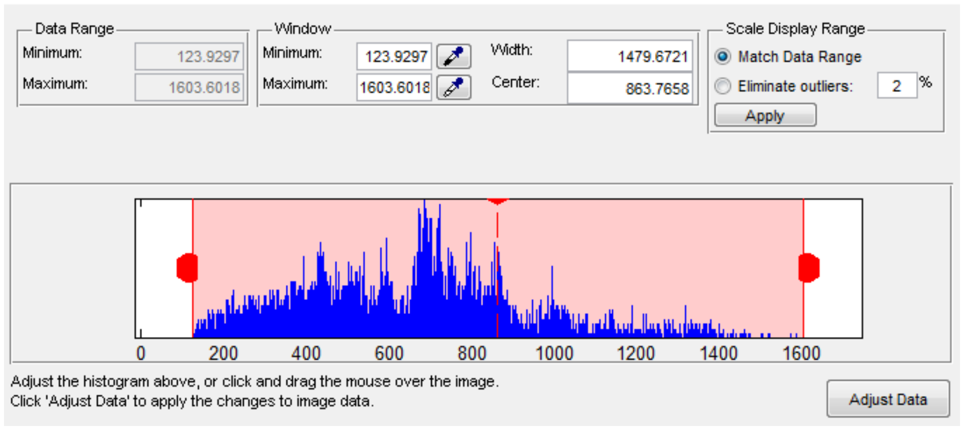

(Hot)


(Jet)


(Gray)

The 'ringing', which is most apparent in the Gray image, is due to the way that intensities from a perfect, noiseless Airy disc (the function that underlies the PSF) add together on a symmetric object like a sphere.

Click on the 'Image Contrast' button, a window like the one shown below should appear. Using the solid red sliders, adjust the size of the red shaded window that lies over the histogram of pixel intensities. This will adjust the black and white points of the image, and thus adjust the contrast. When you are finished, close this window, and click on the '1' in the 'Frame' field below the output axes and hit the 'enter' key – this should reset the image to its original contrast settings.


Data Range

Minimum:	123.9297
Maximum:	1603.6018

Window

Minimum:	123.9297	Width:	1479.6721
Maximum:	1603.6018	Center:	863.7658

Scale Display Range

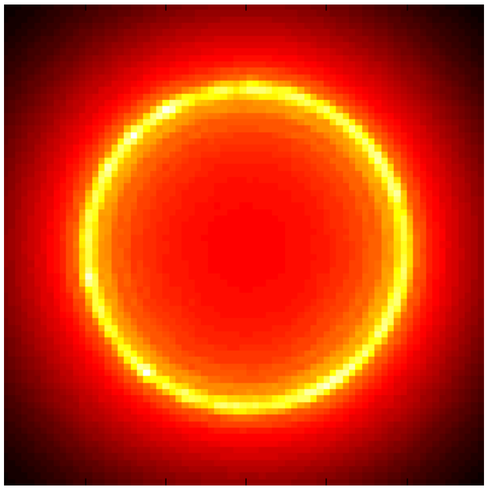
[✓ Match Data Range]

Eliminate outliers: 2 %

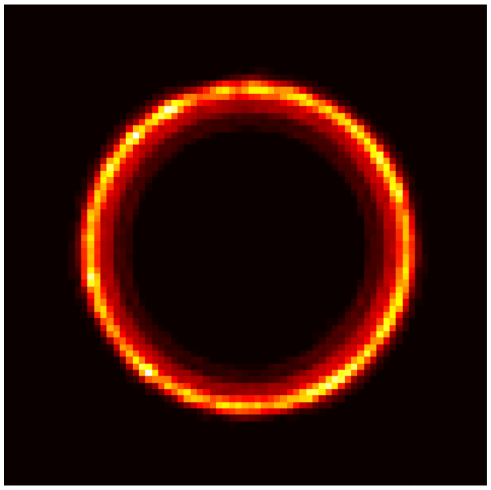
Adjust the histogram above, or click and drag the mouse over the image.
Click 'Adjust Data' to apply the changes to image data.

-- Now click on the 'Fix Contrast' box and set the 'Intensity Min' field to '800' and the 'Intensity Max' field to '1600'.

The image should now look like there is less out-of-focus light around the sphere (simply because that light is dimmer). For comparison here is the before and after images:

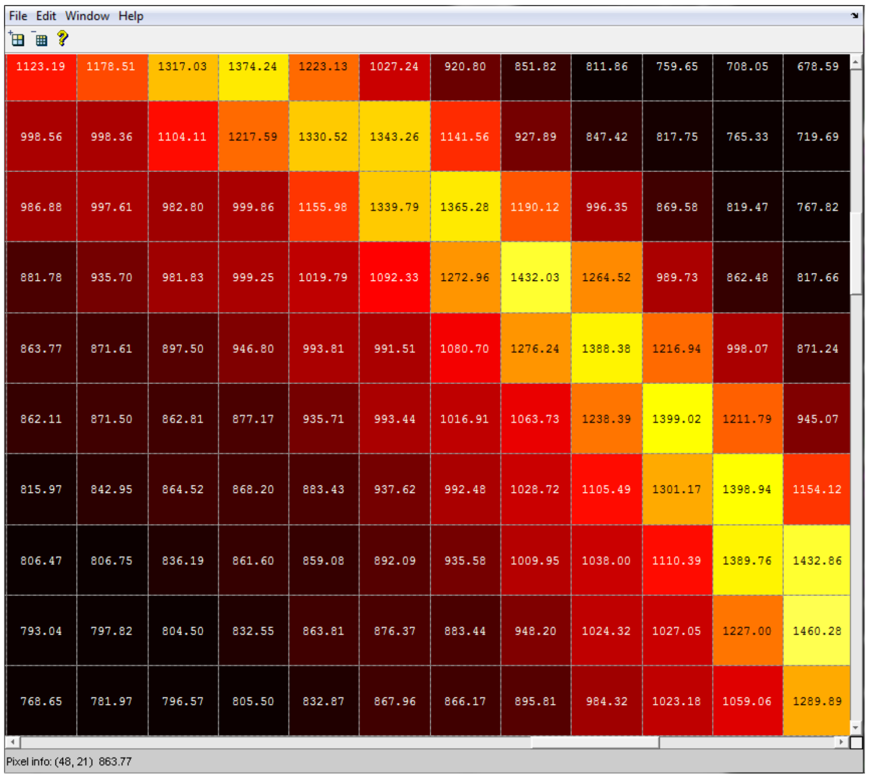


contrast set 0 to 1604



contrast set 800 to 1600

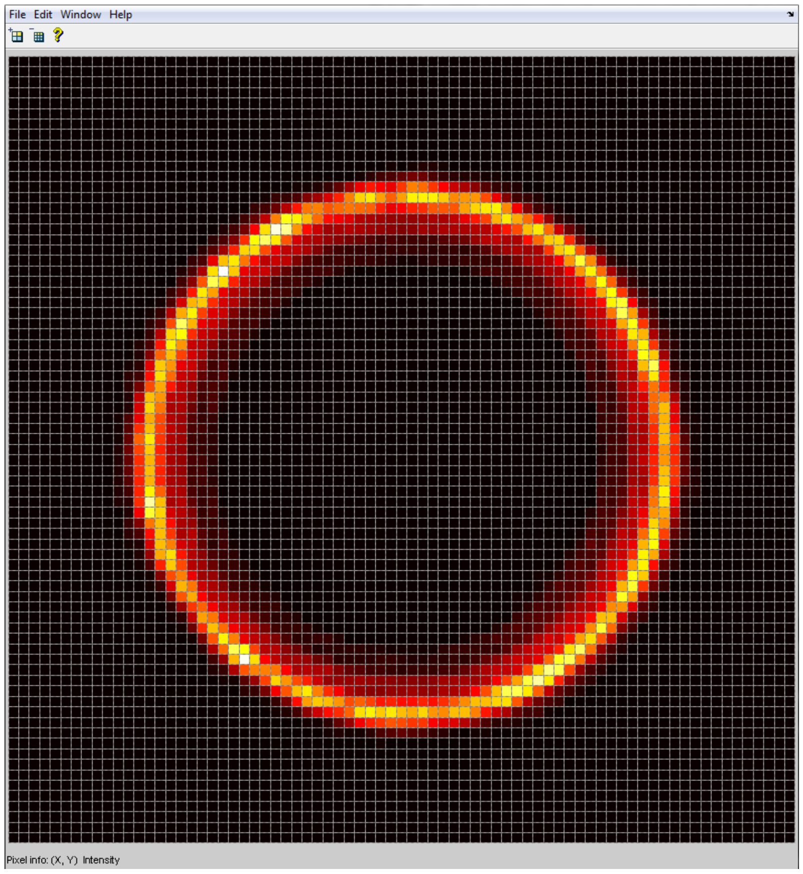
Finally, click the 'Image Info' button. Expand the bounds of this window and pan around the image with the mouse until you see a region of variation in intensity, maybe something like this:



1123.19	1178.51	1317.03	1374.24	1223.13	1027.24	920.80	851.82	811.86	759.65	708.05	678.59
998.56	998.36	1104.11	1217.59	1330.52	1343.26	1141.56	927.89	847.42	817.75	765.33	719.69
986.88	997.61	982.80	999.86	1155.98	1339.79	1365.28	1190.12	996.35	869.58	819.47	767.82
881.78	935.70	981.83	999.25	1019.79	1092.33	1272.96	1432.03	1264.52	989.73	862.48	817.66
863.77	871.61	897.50	946.80	993.81	991.51	1080.70	1276.24	1388.38	1216.94	998.07	871.24
862.11	871.50	862.81	877.17	935.71	993.44	1016.91	1063.73	1238.39	1399.02	1211.79	945.07
815.97	842.95	864.52	868.20	883.43	937.62	992.48	1028.72	1105.49	1301.17	1398.94	1154.12
806.47	806.75	836.19	861.60	859.08	892.09	935.58	1009.95	1038.00	1110.39	1389.76	1432.86
793.04	797.82	804.50	832.55	863.81	876.37	883.44	948.20	1024.32	1027.05	1227.00	1460.28
768.65	781.97	796.57	805.50	832.87	867.96	866.17	895.81	984.32	1023.18	1059.06	1289.89

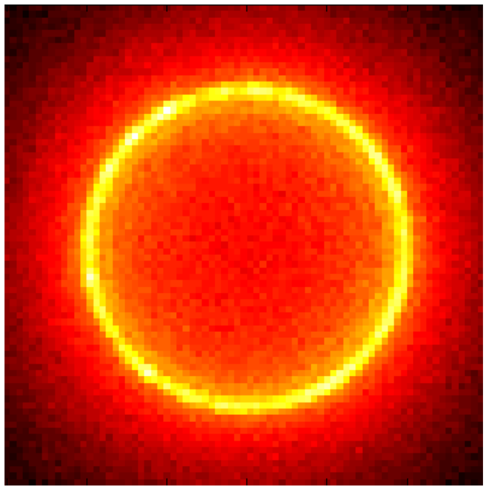
Pixel info: (48, 21) 863.77

Note that as you slide your mouse over the image, the lower left corner displays the pixel address and intensity. In this zoomed-in view, each pixel is labeled with its intensity value. The program computes the image in double precision floating point numbers, and hence, the values have decimals associated with them. It is not until the image is saved as either an 8 or 16 bit TIF that these values are forced to integer values. In the upper left part of this window, there are two icons for zooming and zooming out. Zoom out until you can see the whole image:

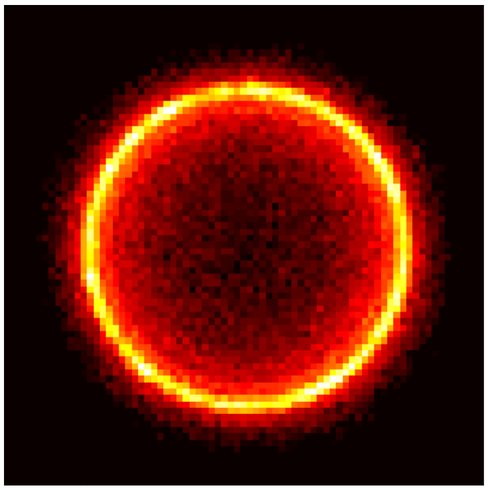


From the 'File' drop-down menu, select 'Print to Figure'. From the resulting window a great many options are available to you. The simplest and possibly most useful of which is to simply hit the save icon ( ), and choose an output file format (TIF, EPS, BMP, JPG, etc) and name for this image. The intensity scale bar icon ( ) applies a nice color bar to the image. The same set of display manipulations apply to any image created in BlurLab, hence we leave it to the user to explore these options in the other contexts of this example and beyond.

7. -- In the ‘Noise & Intensity’ pane, set the ‘Intensity Scaling’ to ‘1’.
   -- Select ‘Gaussian Noise’ and set the ‘Mean’ to ‘100’ and the ‘STD’ to ‘30’.
   -- Deselect ‘Fix Contrast’, and then select the ‘Use for Playback’ box.
   -- Push the ‘Play / Save’ button in the ‘Playback’ pane.
   The image should now look as it did before the contrast adjustment, but now with random Gaussian noise:

   Now select the ‘Fix Contrast’ box, set the ‘Intensity Min’ field to ‘700’, the ‘Intensity Max’ to ‘1600’, and press the ‘Play / Save’ button to get:

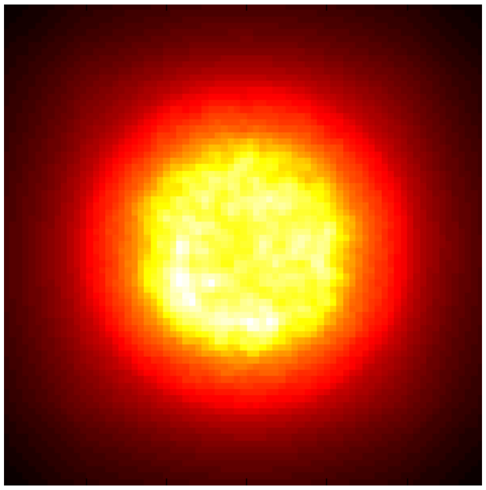
   

   Notice that if you press ‘Play / Save’ repeatedly, a new noise spectrum is added to the image each time you press the button.

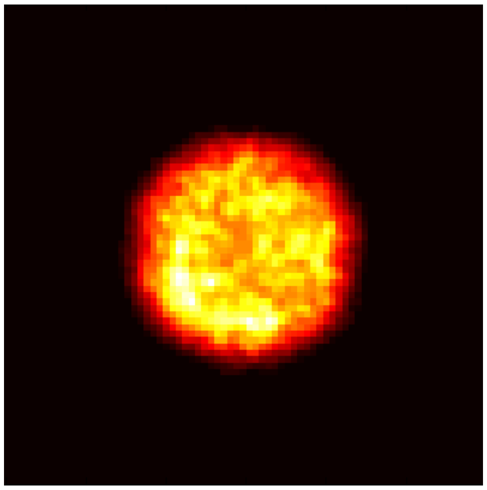
8. -- Now uncheck the ‘Gaussian Noise’, ‘Use for Playback’, and ‘Fix Contrast’ checkboxes.
   -- Set the ‘Intensity Scaling’ back to ‘30’.
   -- Move to the ‘Z Functions’ pane, and move the focal plane from the sphere’s midplane

at Z = 0, to the sphere's surface by setting the 'Z Focal Plane' field to '-1800'.

We can take advantage of the finite depth-of-field of the PSF by centering the PSF not quite at the bottom surface of the sphere, but about $\lambda/2$ above the sphere's bottom surface – this will put more of the sphere in focus in the same image. Hit the 'Simulate' button, this should result in an image that looks like:



Select 'Fix Contrast' and set 'Intensity Min' to '600', to give the higher contrast image:



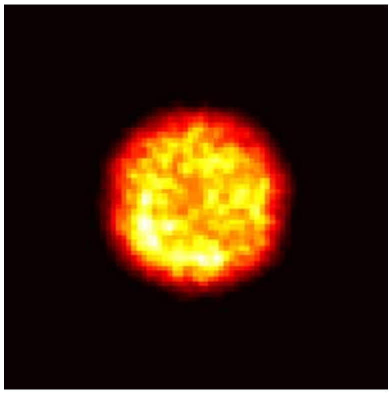
The variations in intensity on the sphere's surface are due to the fact that we are viewing a very quick snap-shot of protein positions, and they are dilute enough on the surface that they create variations in intensity due to fluctuations in surface density. To create an image with a longer exposure, while keeping the same contrast settings, go to the 'Frames & Memory' pane.

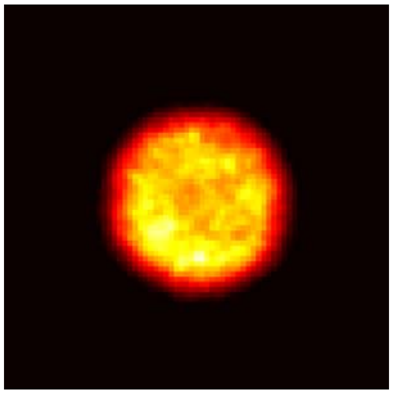
-- Select ‘Boxcar average’, in the ‘Frames’ field enter ‘5’, and in ‘End Frame’ enter ‘5’, and hit ‘Simulate’.

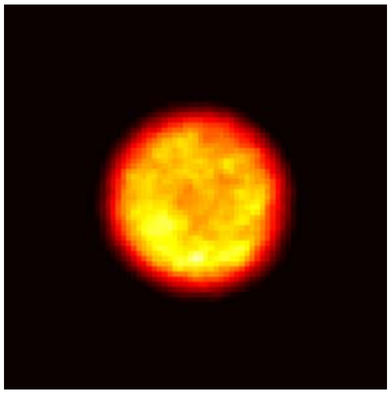
A new indicator will appear below the output axes that indicates which frame is being computed for the Boxcar average.

-- When this has finished, change the ‘End Frame’ and ‘Frames’ settings to ‘10’ and hit ‘Simulate’ again.

You should now have seen three images of the sphere’s bottom with increasing degrees of motion blur due to the increasing effective exposure times, as shown below:


20 ms exposure

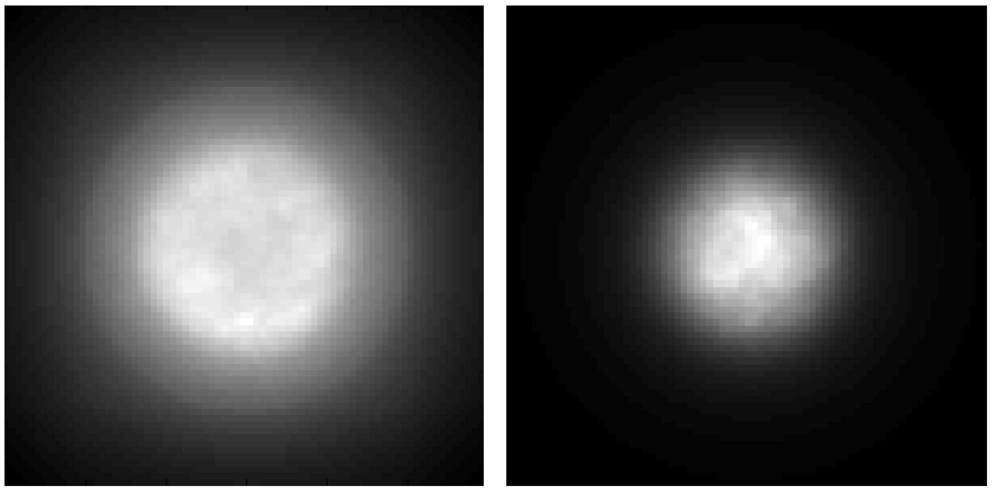

100 ms exposure


200 ms exposure

9. Let us now compare how the bottom of the sphere looks in TIRF with the how it looks in wide-field fluorescence.

   -- Deselect ‘Fix Contrast’ in the ‘Color & Contrast’ pane, and go to the ‘FRAP & TIRF’ pane.
   -- Select ‘Use TIRF’ and set the ‘Decay’ constant to ‘200’.
   -- When you click ‘Use TIRF’ that should automatically set the ‘TIR Z Plane’ to the lowest Z height detected in the input file which is -2000 nm.
   -- Then hit ‘Simulate’.

   Comparing the wide-field and TIRF images reveals the improvement in background when using TIRF:



wide-field fluorescence TIRF

10. -- Now deselect ‘Use TIRF’ and go to the ‘Frames & Memory’ pane.
    -- In the ‘End Frame’ field enter ‘5’ and in the ‘Frames’ field below ‘Boxcar average’ enter ‘5’ – this will simulate a 100 ms exposure.
    -- Go to the ‘Z Functions’ pane.
    -- Select ‘Simulate Z-stack’ and set the ‘Z minimum’ to ‘-2000’, the ‘Z maximum’ to ‘2000’, and set ‘Z slices’ to ‘10’.

    This will take 10 equally spaced Z-sections through the sphere, averaging each for a 100ms exposure. Press ‘Simulate’ and you should see ten frames sequentially appear on the display axes. It may take a few minutes to simulate this Z stack since in total 50 frames must be simulated to create the Z stack.

    Once the simulation has finished:
    -- Go to the ‘Noise & Intensity’ frame.
    -- Set the ‘Intensity Scaling’ to ‘1’ and select ‘Poisson Noise’.

    We will assume that the counts from the simulated image represent electrons on the CCD and that those electrons represent the arrival of photons (without read noise). Thus the intensity of the image at each pixel can be used as the Poisson parameter, and hence we will leave the ‘Int. Multiplier’ field equal to ‘1’.

    -- Select ‘Use for Playback’.
    -- Go to the ‘Image Output’ pane and select ‘Save Images’ and this will automatically select ‘Save Parameters’.

-- Ensure that ’16-bit’ is selected.

The output name should already have a form of:

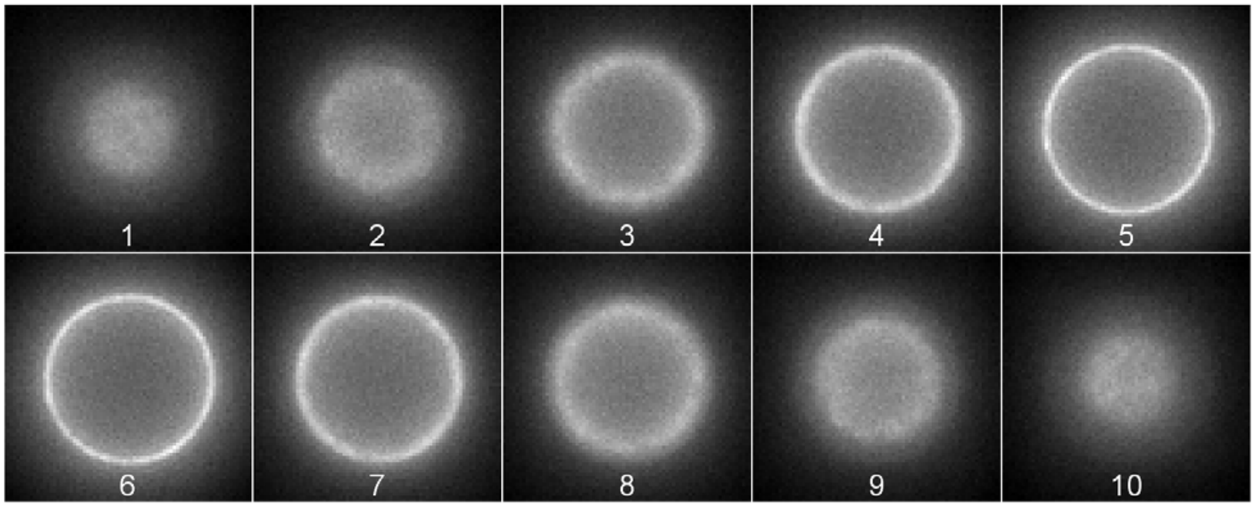
‘output_date_example_spherical_diffusion_R-2_rho-200_dx-0.283.tif’

and the parameter text that will be created will have a name of the form:

‘output_date_example_spherical_diffusion_R-2_rho-200_dx-0.283.txt’.

-- Finally, press ‘Play / Save’ in the ‘Playback’ pane, and press ‘Yes’ when asked if you want to save the playback.

When put into a montage (which was done with ImageJ), the output TIF stack should look something like:



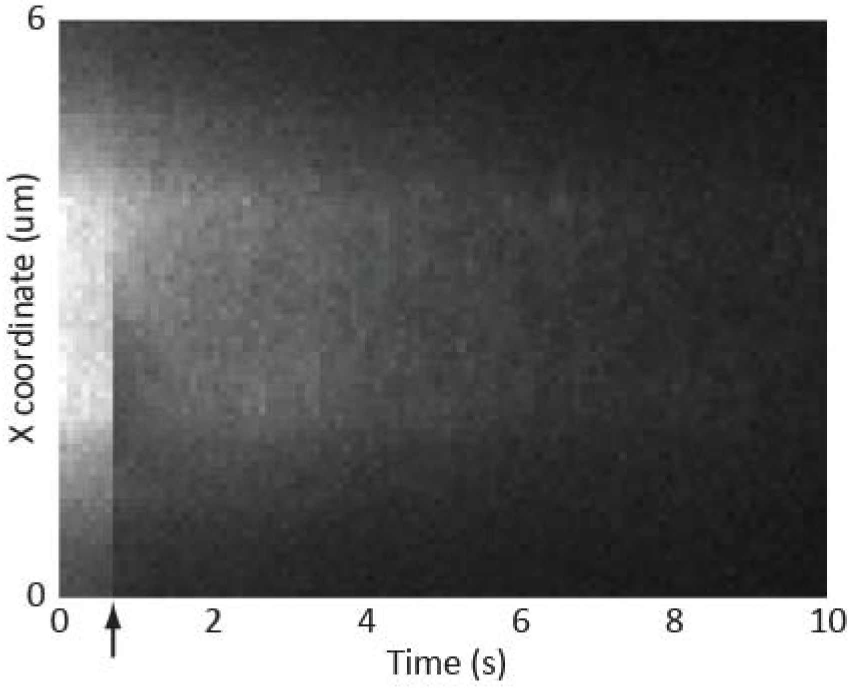
One can take these images and construct maximum intensity or 3D projections from the Z-stack. For instance, in ImageJ one can perform these manipulations using the menu options: Image > Stacks > ZProject or 3D Project.

11. Finally, we will go to the bottom of the sphere, FRAP one half of the sphere, and measure the recovery, with all of the most realistic processes possible.

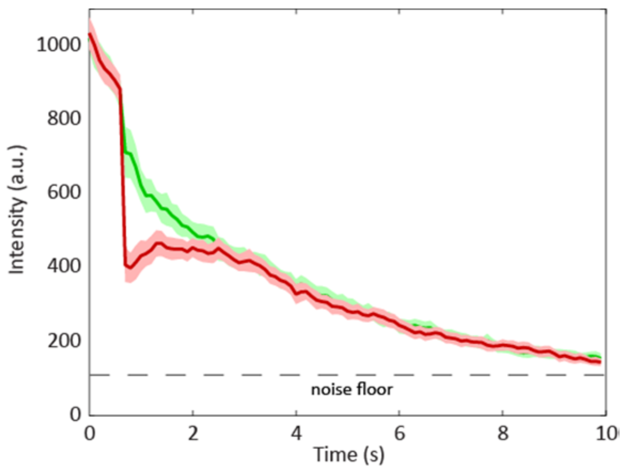
    -- You should still be in the ‘Z Functions’ pane. Deselect ‘Simulate Z-stack’ and set the ‘Z Focal Plane’ to ‘-1800’.
    -- Go to the ‘Noise & Intensity’ pane.
    -- Set the ‘Intensity Scaling’ to ‘30’.

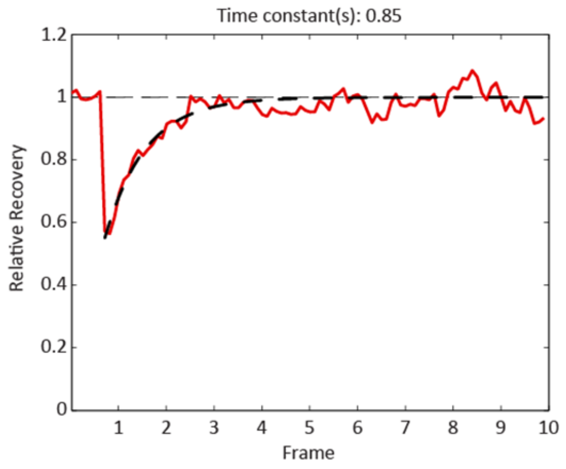
-- Select ‘Gaussian Noise’ and set the ‘Mean’ to ‘100’ and the ‘STD’ to ‘7’.
-- Select ‘Poisson Noise’ and set the ‘Int. Multiplier’ to ‘1’.
-- Select ‘Photo-bleaching’ and ‘Stochastic’, set the photo-bleaching time constant ‘tau’ to ‘200’, which in this case is an actual time constant of four seconds.
-- Go to the ‘Frames & Memory’ pane and set the ‘End Frame’ to ‘500’.
-- In this pane, ‘Boxcar average’ should already be selected with ‘Frames’ set to ‘5’.
-- In the ‘Image Ouput’ pane select ‘Save Images’ and ‘Save Parameters’.
-- In the ‘FRAP & TIRF’ pane, make sure ‘Use TIRF’ is not selected.
-- Press the ‘Select Region’ button and notice that the mouse is now a pair of cross-hairs. In the output image axes, select approximately half of the sphere. You will be asked if you are satisfied with the selected region, if so, select ‘Yes’, and this will automatically check ‘Use FRAP’.
-- Set the ‘FRAP Frame’ to ‘10’ and set the ‘Percentage’ to ‘100’.
-- Press ‘Simulate’ and go get a coffee – this will take a few minutes.

A kymograph across the center of the output image stack should look something like this (the arrow indicates when photobleaching occurred):



If you are running Matlab, you can go into the ‘m-file_models’ and launch the ‘frap_processing.m’ script which will allows you to perform the following analysis on the TIF image stack that BlurLab produced by FRAPing the sphere. The first plot below shows the mean intensity of pixels in the FRAPed region in red and of the non-FRAPed region in green. Notice the clear recovery of the red line and the clear decrease in the green line as active emitters move from the non-FRAPed region into the FRAPed region. On the longer time scale both regions decrease in intensity due to photo-bleaching, and the equalization of fluorophores on the sphere's surface causes the green and red lines to approach each other. The second plot is the ratio of the red and green lines, which shows the full recovery of the FRAPed region relative to the non-FRAPed region, fit with an exponential recovery curve.





We hope you have found this example useful. Please report any bugs to natsirt@stanford.edu.

8. Helpful m-files

Many of the m-files included with the installation are part of the BlurLab program itself, however other external m-files are included to help you simulate different scenarios, create input files easily, or analyze output.

BlurLab_text.m

This m-file is designed to create properly formatted input text files for BlurLab. Within Matlab, the function *BlurLab_text* can be called when one wants to create an output file. Once a mechanistic model has been simulated, and the X, Y, and Z positions, intensities, frames numbers, and labels of all of the point emitters are known for the model in question, calling

```
>>BlurLab_text(X,Y,Z,I,F,L,'file_name.txt')
```

will create a text file 'file_name.txt' properly formatted for BlurLab. Note that X,Y,Z,I,F, and L must all be column vectors.

BlurLab_sphere_example.m

This is the m-file that was used to create the input file for the example in the last section, which simulated diffusion of particles on a sphere. Open the m-file in Matlab and you can adjust:

M = the number of iterations in the simulation (i.e. number of time steps)

R = the radius of the sphere in microns

rho = the density of fluorescent particles in units of number per square micron

D = the diffusion coefficient in units of square microns per second

dt = the time step of the simulation in seconds

For a fixed density, more iterations or a larger sphere will generate a larger input file. Higher temporal resolution can be achieved by choosing smaller values of '*dt*'.

Once the m-file is executed a plot will appear showing the simulated diffusion – while interesting, this can be slow and hence one may want to turn off this graphical output. To do so, comment out lines 73-75. When the simulation has finished, Matlab will prompt if a text file should be written. Typing 'yes' will create the input text file, with the name 'date_spherical_diffusion_R#_rho#_dx#.txt'.

frap_processing.m

This file will analyze different regions of a TIF stack to determine the relative intensities in sub-regions of the image – it is therefore suitable for analyzing FRAP data. Running this m-file will bring up a GUI in which the user chooses the file to be analyzed. The first frame of the chosen image stack will be displayed and from this image, a FRAPed and a non-FRAPed region must be chosen by the user. The user will also be asked to assign a time scale to the frames of the stack, but if no time scale is entered, a time scale of one second between frames is assumed.

The output of this m-file is two plots. The first plot shows the absolute mean intensity in the selected FRAPed region in red and the absolute mean intensity of the selected non-FRAPed region in green, both as a function of time. The standard deviations of the intensities in these regions are shown as dashed line above and below the mean. The second plot is the ratio of the curves in the first plot, which is a measure of the relative recovery rate. This plot has the advantages that it is not affected by photo-bleaching and will always return to one as long as there is full relative recovery between the two regions.

The program will also ask the user whether to fit an exponential curve to the relative recovery plot and will output the fitted time constant of recovery.

helix_maker.m

This file allows one to create three dimensional helices with stochastic labeling along the length of the helix. The helix's radius, length, ellipticity, labeling density, and rotation can all be adjusted. There is no explicit notion of time in this simulation, however multiple frames are simulated that allow the helix to rotate at a constant rate. Files created by this simulator have the name structure 'helix_input_file_date.txt'. The parameters within the file are:

N = number of frames to be simulated

L = length of the helix projected along its central axis in nanometers

Ry = helix radius in nanometers along the first orthogonal direction

Rz = helix radius in nanometers along the second orthogonal direction

ptch = pitch of the helix in turns per nanometer

lam = stochastic labeling density along the length of the helix in fluorophores per nanometer

phi = the initial helix rotation, before the active helix rotation between frames

sig = sets the Gaussian 3D jitter in the points forming the helix in nanometers

dtheta = incremental helical rotation between each frame of the simulation

Setting *Ry=Rz* creates a circular helix.

line_maker.m

This file allows one to create lines in three-space with stochastic labeling along the length of the line. The line's length, rotation, labeling density, translational speed and rotation can all be adjusted. There is no explicit notion of time in this simulation, however multiple frames are simulated that allow the line to translate at a constant rate. Files created by this simulator have the name structure 'line_input_file_date_L#_lam#.txt'. The parameters within the file are:

N = number of frames to be simulated

L = length of the line in nanometers

lam = stochastic labeling density along the length of the line in fluorophores per nanometer

m = slope of the line in the X-Y plane (first primary rotation)

b = offset of the line in the X-Y plane along the Y direction in nanometers

phi = rotation of the line about the X-Z axis (second primary rotation)

sig = sets the Gaussian 3D jitter in the points forming the line in nanometers

spd = incremental linear translation between each frame of the simulation in nanometers

Setting *m, b,* and *phi* all to zero puts the line along the X-axis flat in the X-Y plane.

9. Advanced Concepts

BlurLab simulates the 3D light field produced by the 3D positions of point emitters. It performs this function best when certain conditions are met. Thinner samples will be simulated more accurately than thicker samples. This is due to two distinct effects. First, BlurLab currently does not have the capability to impose non-uniform illumination fields, hence each emitter is fluorescing as if it is being exposed to the same intensity if incident excitatory light. The depth-of-field of the magnification system is set by the numerical aperture, and dictates the illumination depth over which excitation can be reasonably approximated as constant. A plot of this function is shown in Figure 1. Second, depending how the PSF image stack is setup, light emitted from far out of the focal plane may be ignored by the program. Thus it is important to choose carefully what PSF one uses for a sample with a given spatial size, to ensure that the size of the object is not significantly larger than the depth of the PSF.

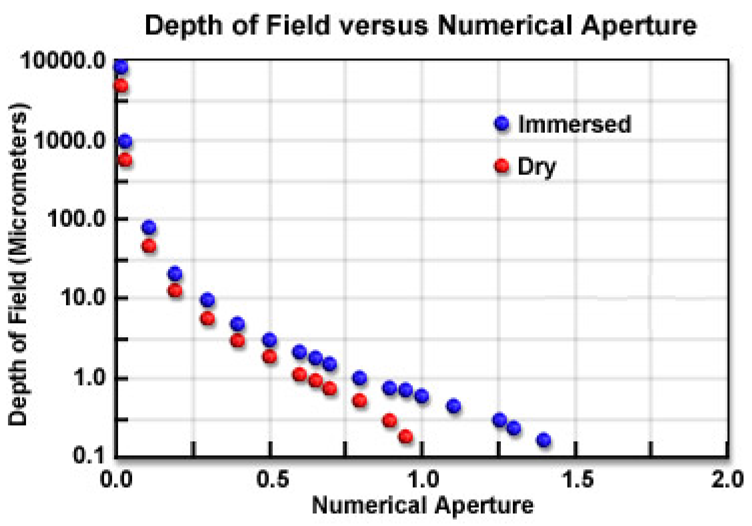


Figure 1: Plot of depth of field versus numerical aperture of a focusing system. Graph courtesy of Nikon's MicroscopyU (http://www.microscopyu.com/articles/formulas/formulasfielddepth.html)

BlurLab is particularly well-suited to processes where fluorescing objects are not moving significantly during the exposure time of an image, or in other words, situations where motion blur is insignificant. Although, BlurLab can, at higher computational cost, compute motion blurred images.

Simulating motion blur is fairly straightforward, if one has a notion of how fast particles are moving. Choose a temporal sampling rate that is high enough so that the width of the point spread function is larger than the expected inter-frame spacing between a single particle due to motion at the higher temporal resolution. Then Boxcar average over the higher temporal resolution frames such that they add up to one exposure time. The Boxcar averaging scheme for an image I_j, which is temporally up-sampled by a factor n is given by

$$I_j = \frac{1}{n} \sum_{i=1}^{n} I_{n\cdot(j-1)+i}$$

where n is the number of higher resolution frames blurred to form single exposure image.