An open source LABVIEW platform for simulating image series of fluorescent microtubules in gliding assays

Lawrence J. Herskowitz (lherskow@unm.edu), Steven J. Koch (sjkoch@unm.edu)
University of New Mexico, Center for High Technology Materials and
Department of Physics and Astronomy

Introduction

Kinesin family proteins are motor proteins that are able to use chemical energy to translocate on microtubules. They are essential to many cellular processes such as cell reproduction and axonal transport (1; 2; 3). A technique that has proven very valuable to studying kinesin's physical properties is the gliding motility assay (GMA) (4). In the GMA, the kinesin tail is fixed to a slide with the motor domains exposed to the solution. A solution of microtubule with buffer and ATP are then added to the flow cell. This allows the microtubules to be propelled by the kinesin motor domains as depicted in Figure 1. Often the microtubules are labeled with dye molecules that allow them to be seen through fluorescence microscopy (5). GMAs enable the study of many kinetic properties of wild type and mutant kinesins, such as gliding speed, microtubule polarity, and force induction in the kinetic process (6; 7; 8; 9).

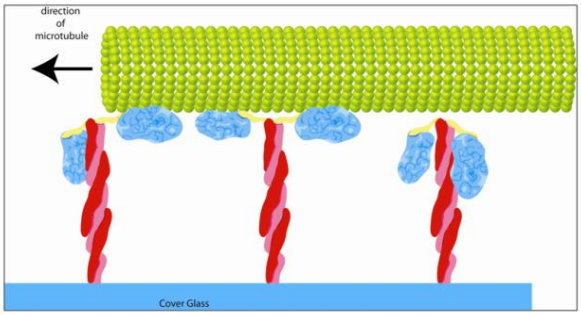

Figure 1: This is an artist rendition of a gliding motility assay. The kinesin are attached to the cover glass so that the motor domains are exposed to the flow cell. When ATP is in the solution, the kinesin will walk along the microtubule which then will be propelled forward.

To extract information from these experiments, it is often imperative to track the moving microtubules through a series of images, obtaining the position versus time of the microtubules. However, manual tracking means going through hundreds of images and recording the position of the microtubule by hand. This is a tedious and exhausting process, though it certainly has proven useful (10). The data are also potentially more prone to selection bias and systematic and random errors than automated tracking. Because of this several groups have developed microtubule tracking software (11; 12) (23). To test the effectiveness of these tracking programs, many laboratories will track a stationary object. This however is not ideal since it does not match experimental conditions well. To characterize the systematic and random errors in the tracking algorithms, it is best to test them on a moving object. One way this can be achieved is by moving a fixed microtubule with a piezoelectric stage. However, even in that case it is difficult to completely eliminate drift and other effects. A simulated image series can eliminate those problems and is often ideal for testing tracking algorithms (13). For this purpose we developed an application that can generate microtubule images that mimic those captured from a typical gliding motility assay. A simulated image is shown in Figure 2A next to an actual gliding motility assay image shown in Figure 2B.

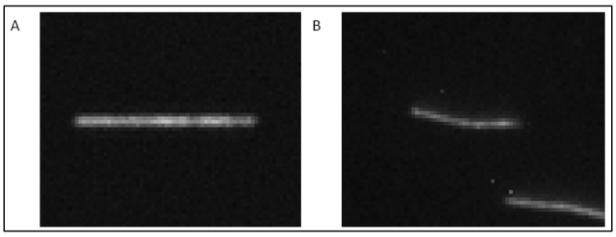

Figure 2: Image A was created using the software written in this paper. Image B is an image of a microtubule from a gliding motility assay. The settings used to create this image can be found at http://kochlab.org/files/Simulating%20Images%20Paper named imitating andys tube.ini

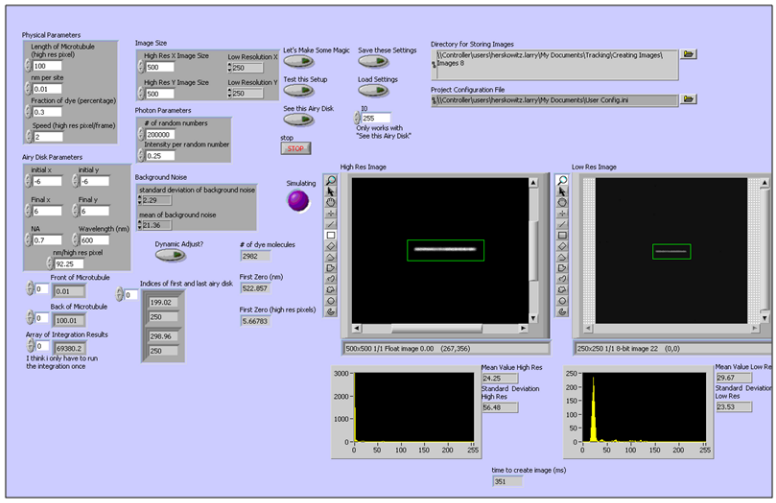

Figure 3: This is an image of the front panel of this software. The top left corner contains the main user inputs that are described in this paper. To their right are the buttons to create a series of images (top left), test this set up to create just one image (second from top on left), below that is the option to see just one airy disc from the "Airy Disk Parameters," after that is the stop button to stop the program. The right column of buttons from top to bottom are to save the settings on the screen and the load settings. The save button saves the settings as an .ini file in the path to the right named "Project Configuration File." The load button will load the settings from that .ini file. Above "Project Configuration File" is the directory location to save the series of images. The two images on this page are the high resolution image (left) and the low resolution image with background noise (left). Below that is the histogram of pixels inside the ROI drawn on the corresponding image. A video tutorial created with camstudio for this software can be found at https://sourceforge.net/project/admin/explorer.php?group_id=312060.

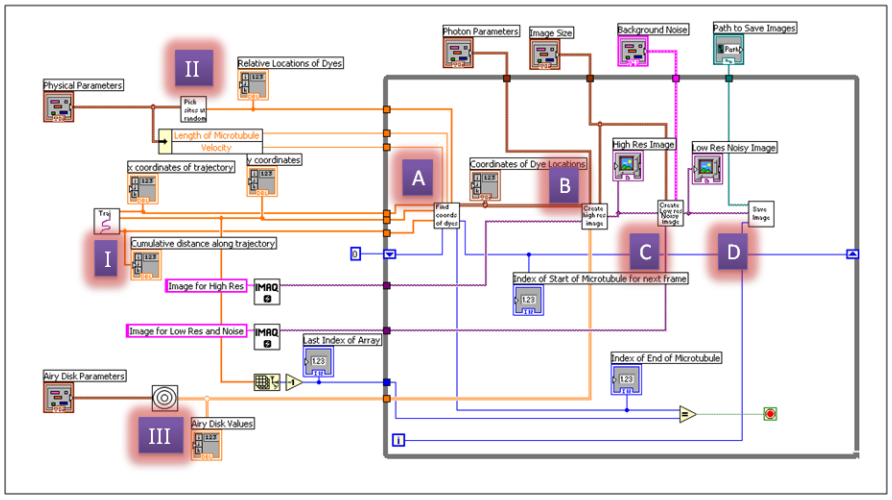

Figure 4: Simplified diagram of the image series simulation algorithm. See text for a clear description of the representation of this figure

The application was written in National Instruments LABVIEW 7.1. The user interface can be seen in the Figure 3 with user inputs on the left and top and outputs on the right and bottom.

This paper is demarcated into two sections. *Section 1* describes the process implemented to produce the simulated image, the controls that the user can change, and the method for storing images and settings. *Section 2* provides instructions on how to obtain the open source code, the compiled application, and a video tutorial.

SECTION 1

Algorithm Overview

Figure 4 shows a simplified diagram of the overall algorithm. This paper will go into detail of how each step of the algorithm works. There is a main while loop seen in the diagram (see Reference (14) for a LABVIEW video tutorial). In each iteration a frame is constructed and saved. Before the while loop is entered, three initialization tasks are completed shown as subVIs I, II, III in the figure. (I) First the user must create the trajectory the microtubule will follow in the image series that will be created. This is done in the trajectory sub.vi. (II) Next using the settings in the "Physical Parameters" control the program randomly sets the locations of the dye molecules along the microtubule length. (III) The third task is to create a prototype Airy Disk from the "Airy Disk Parameters." Each one of these subVIs is completed only once for each set of images, prior to execution of the while loop that generates each of the individual images.

In the while loop, first the absolute coordinates of the dye molecules are set in the subVI labeled "A" in Figure 4. This VI, "Find Coord of Dyes.vi" has inputs of the microtubule trajectory, the index of the start of the microtubule, speed, and length of the microtubule. SubVI A has three outputs: "Coordinates of Dye Locations" (bundle of two double precision numbers), the "Index of Start of Microtubule for next frame" (int32) and the "Index of End of Microtubule" (int32). The dye location coordinates determine the locations of the centers of the airy disks in the simulated images. These are absolute coordinates (relative to the top left of the image) instead of the relative coordinates with respect to the microtubule end that the "Pick sites at random" subVI outputs. The second output, "Index of Start of Microtubule for next frame," goes into a shift register and replaces the previous iteration's index for the start of the microtubule. The distance the microtubule moves between frames is calculated from the speed inside of this subVI. This is used to find the index along the trajectory the microtubule will start in the next frame. The index of the end of the microtubule is calculated using the length of the microtubule value similar to calculating the index for the next starting index.

After subVI A finishes, it passes parameters to subVI B, "Create high res image" including the coordinates of the dye locations, a pointer to an image location and the user inputs "Photon Parameters" and "Image Size." Inside subVI B are the probability distribution function and the Monte Carlo algorithm to randomly select the photon locations.

Next is subVI C, "Create low res noisy image," with the inputs of "Image Size," "Background Noise" and the "High Res Image." SubVI C resamples the high resolution image to create a lower resolution image

that matches experimental resolution, dictated by the user input. If selected, Gaussian noise is added to the background. The output of subVI C is the final image.

SubVI D saves the image in a directory the user has chosen, and named after the frame number which is the iteration number, such as 1.png. When the end of the microtubule index has reached the end of the trajectory array, the while loop stops.

This section will be broken up into four segments to provide details of the above subVIs: Microtubule Construction, Resampling and Noise Addition, Trajectory, and Saving. In each of these segments the necessary user input is highlighted as well as a simplified version of the code.

Microtubule Construction

User Input:

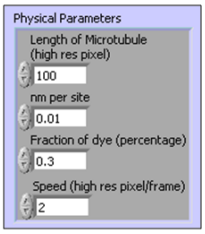

Figure 5: Physical Parameters user input: sets the length of the microtubule, minimum spacing of dye molecules (pixels per site), probability (out of 1) of dye molecules attaching to each location, and the speed of the microtubule.

This section will go into detail of how a single image of a microtubule is constructed. The coordinates for the dyes need to be randomly selected along the length of the microtubule. The user can set the microtubule length via the “Length of Microtubule” control. This control is in units of high resolution pixels. This algorithm produces two images; a high resolution image and a lower resolution image. The final image is the low resolution image. This allows for more accurate numerical estimation of the airy disk probability density function which is implemented in the Monte Carlo algorithm. The “high res pix per site” control dictates how many fluorescent dyes are theoretically possible to be found in each pixel. For a typical fluorescent tubulin preparation (15), the molecules are attached to surface lysine residues on the alpha/beta tubulin dimer. There is one dimer per 8 nm on a single protofilament. For a 13-protofilament microtubule, there are thus approximately 1.5 dimers per nanometer. There also could be multiple dye molecules on a single dimer, which would further reduce the number of high resolution pixels per dye site. This control will change depending on the users desired nm/pixel ratio.

The first two controls thus dictate the microtubule length and the maximum number of dyes that can be attached. In the example shown in the figure, there can be a maximum of 10,000 dyes. However in practice, fluorescently labeled tubulin is mixed with unlabeled tubulin resulting in partially labeled microtubules after polymerization. To mimic this, the program uses a Monte Carlo method to determine if a given site will be labeled dependent on the fraction of dye control. A uniform random number from 0 to 1 is selected for each site. If that random number is lower than the user-set labeling fraction, the site is deemed as labeled. If not the site is left as unlabeled. This code can be seen in Figure 6. There is an indicator on the front panel seen to the right of the dynamic adjust control in Figure 3 which specifies the number of dye molecules on the microtubule.

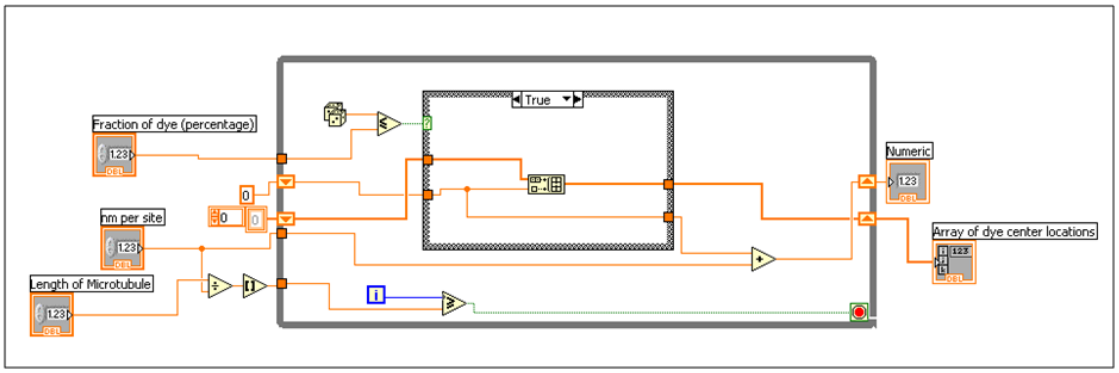

Figure 6: Code for determining location of dye molecules. A random number is generated for each possible site. If the random number is lower than probability value input into Fraction then the site is added to the array of dye centers.

If the site is selected to be labeled, an airy disk is centered at that location. An airy disk represents the fluorescent dye in the image plane since the dye emits light that is captured by the circular aperture of the objective (16). Mathematically an airy disk is described with the following equation:

$$I(r) = I_0\left(\frac{2J_1(kr)}{kr}\right)^2$$

where I_0 is the maximum intensity of the airy disk center, $J_1(kr)$ is the Bessel function of the first kind in circular coordinates, r is the radius in high res pixels, and k is what we have named the prefactor. It can be calculated by

$$k = \frac{2\pi NA}{\lambda}C$$

NA is the numerical aperture of the objective, and λ is the wavelength in nanometers. C is the conversion factor from high resolution pixels to nanometers because we represent r in pixels. The first zero for the airy disk can be calculated in nanometers using

$$r_0 = \frac{.61\lambda}{NA}$$

where r_0 is the distance to the first zero (17).

Since the microtubule is narrow compared to the airy disk radius, we treat it as a one-dimensional object. So each airy disk location is recorded as a distance from the end of the microtubule, and these relative distances along the contour remain fixed while the microtubule moves or curves during the simulation.

This method of labeling dyes randomly along the microtubule is supported by prior experimental work. A microtubule in a solution with a low concentration of fluorescent dyes looks speckled. Waterman-Storer and Salmon studied this phenomenon and showed that this is caused by a non-uniform distribution of the fluorescent dyes along the microtubule proto-filament in a purely stochastic process (18).

User Input:

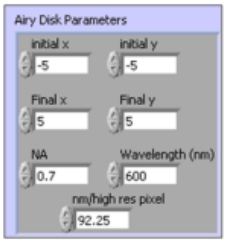

Figure 7: Airy Disk Parameters. Airy disk is centered at 0,0. Initial x,y controls set the left, top of the bounding box. Final x,y controls set the right, bottom. Limiting the box size increases computation speed. NA sets the numerical aperture value. Wavelength represents the wavelength of the emitted light in nanometers, and the final input is the conversion factor that transforms nm to high res pixels. These final three inputs are used to calculate the prefactor of the airy disk as seen in equation X.

Since the airy disk intensity falls quickly, it is not necessary to calculate the airy disk over large values. The numerical aperture and wavelength determines how quickly the airy disk reaches zero. A higher numerical aperture or shorter wavelength means fewer pixels need to be calculated to get an accurate representation than a higher numerical aperture and longer wavelength. This is illustrated in Figure 8. It is obvious that the airy disk in 8A with a radius of 5.67 high res pixels (522.86 nm) needs fewer pixels than 8B whose radius is 19.84 high res pixels (1,830 nm) to approximate the dye molecule. In the user input image above, a square of -6 to 6 for both x and y values was chosen. This means that a 12 pixels x 12 pixel box centered at the airy disk's center was used for each airy disk in the simulation which has the same radius as Figure 8A. This is done to ensure a shorter processing time for the software. The radius of the airy disk is calculated in both nanometers and high res pixels. This can be seen on the front panel in Figure 3 below the number of dye molecules indicator.

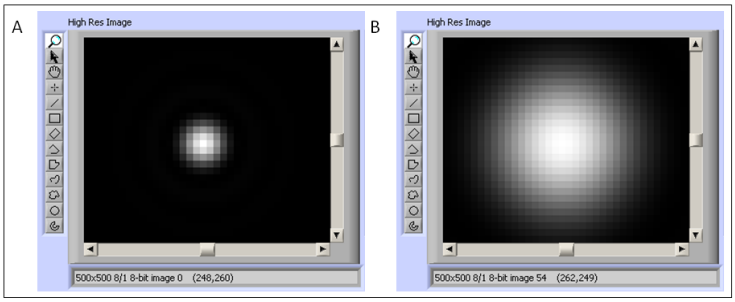

Figure 8: Airy disks with differing characteristic sizes. (A) size of 522.86 nm (5.67 high res pixels) (B) size of 1830 nm (19.84 high res pixels). A smaller characteristic size requires a smaller bounding box (described in Fig. xx).

User Input:

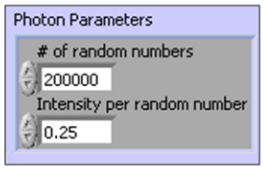

Figure 9: Photon Parameters. "# of random numbers" determines how many virtual photons will arrive in the image. "Intensity per random number" determines how much the pixel value will be incremented by one photon.

Generating the total probability density function for photon arrival in the image plane

After all the coordinates of all airy disks have been determined, the functions are added together. This is shown in the equation below.

$$I_T = \sum_{i=0}^{M} I(|r_{pixel} - r_{shift}|)$$

Here I_T is the total intensity of all *M* airy disks added together. $I(r)$ is the intensity profile of a single airy disk. Its center is shifted inside the image by r_{shift}. I_T is then normalized by numerical integration to produce the probability density function (PDF) for photon arrival. Each pixel coordinate now represents the probability of a given photon landing there.

Generating images based on PDF for photon arrival

Photon locations are randomly selected using a Monte Carlo method and the total photon PDF. To minimize the number of random numbers needed for each photon arrival, all values in the two dimensional PDF is flattened into a one-dimensional cumulative probability array. This is seen in Figure 10. Because the PDF was normalized, the last element in the cumulative probability array is 1. Each element in the cumulative array has a corresponding pixel coordinate. For a given random number, the index of the cumulative array element closest to that value (without exceeding it) is found by a binary search algorithm. The value of pixel corresponding to that element is increased by an amount set by "intensity per random number." This method requires only one random number for each photon (19). This code is shown in Figure 11.

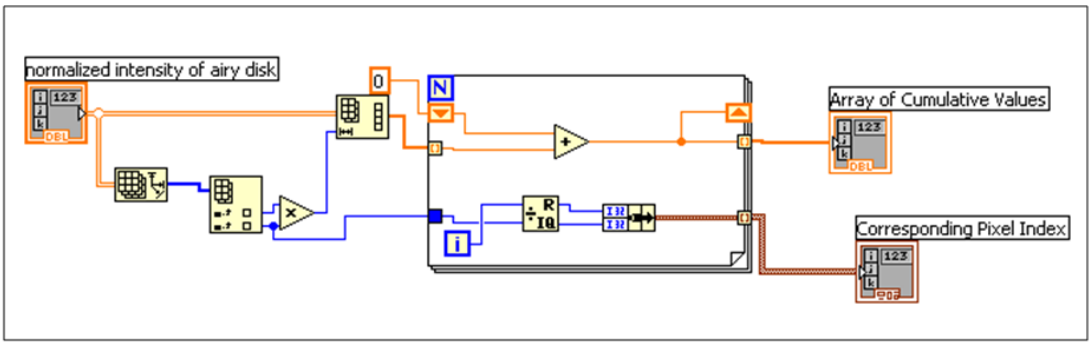

Figure 10: Code to create cumulative array used in Monte Carlo algorithm for determining photon location. The normalized intensity of the overlapping airy disks enters this subVI as a 2-d array and is flattened into a 1-d array. The newly formed 1-d array is indexed by the loop where each element is added together. The "quotient and remainder" VI calculates the corresponding pixel coordinates which are bundled together and formed into a 1-d array. Each element in the pixel index array corresponds to an element in the cumulative probability array.

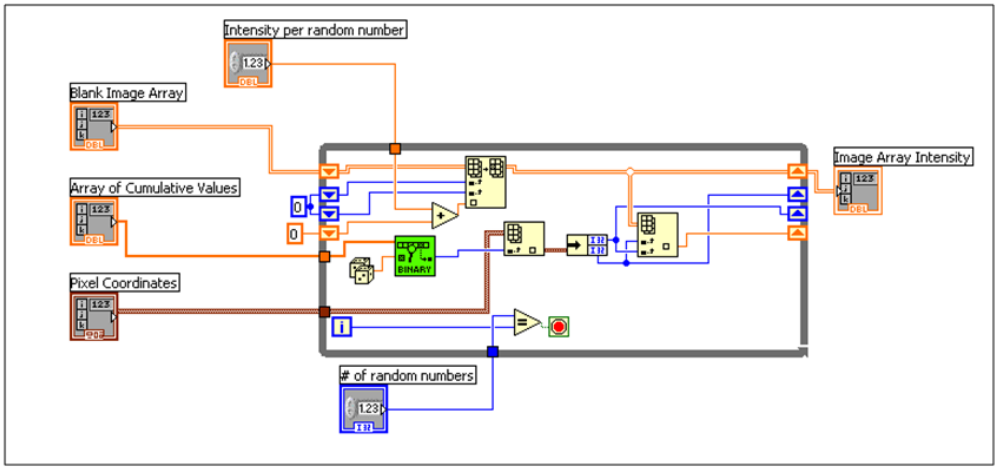

Figure 11: Code for simulating photon arrival in high resolution image. The dice generates a uniformly distributed random number from 0 to 1 and then the green VI (22) uses a binary search algorithm to find the highest index in the cumulative array whose value is lower than the random number. This index is used to find the corresponding pixel in the Image Array, which is incremented by the amount "Intensity per random number." The intensity of the corresponding pixel is then increased by the user set amount of "intensity per random number." This while loop continues until it has reached the number of photons specified by the user.

The user chooses how many random numbers, or photons, used for each image. Figure 12 represents what an image from this step in the process looks like. The image produced is a floating point image to allow for any number of photons per pixel. The image is next cast to an 8-bit image, with a maximum pixel value of 255. The user is given the option of dynamically adjusting the image so that the maximum pixel value in the floating point image set to 255. If the user chooses not to do this, the image can mimic either under- or over-exposure. The latter case will show saturated pixels.

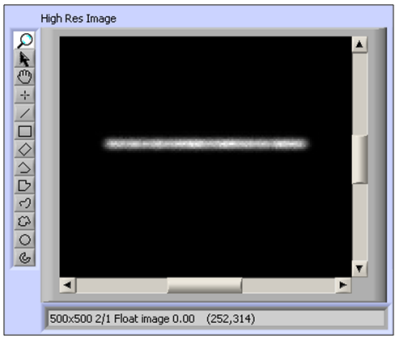

Figure 12: Examples of simulated high-resolution microtubule images. Dye-labeling probability of .3.

Resampling and Noise Addition

User Input:

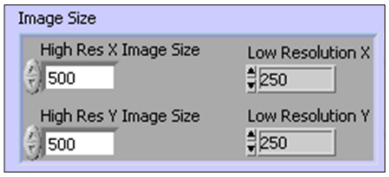

Figure 13: Paremeters for image sizes. On the left, the width and height (in pixels) of the high-resolution image is set. On the right is low-resolution image.

After float image is cast into an 8 bit image, the image is then resampled into a lower resolution image to match experimental resolution. The user inputs the x and y resolution of the higher resolution image in "High Res X Image Size" and "High Res Y Image Size." The user can then control how much the lower resolution image will be in "X Resolution" and "Y Resolution." This resampling into a lower resolution image is done to allow for a more precise numerical integration to create the probability density function.

User Input:

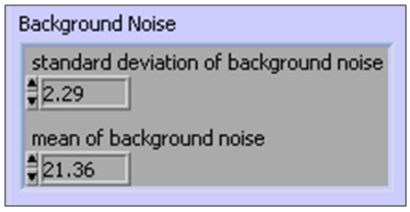

Figure 14: Background noise parameters. The mean and standard deviation (in 8 bit pixel intensity) for Gaussian probability distribution are specified.

Finally, background noise can be added to the image depending on user settings. We add background noise to the 8-bit resampled image. The user chooses the mean value and standard deviation of the

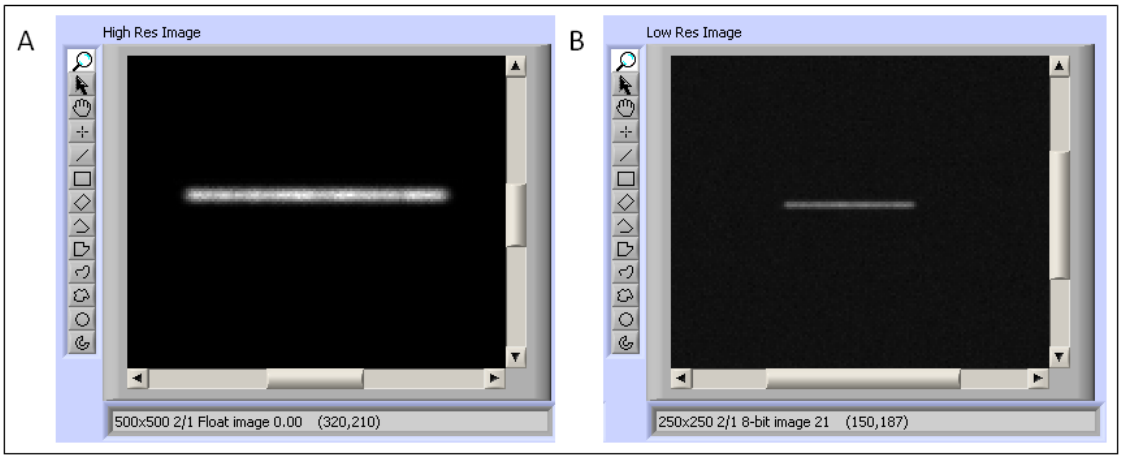

Figure 15: Example of image resampling and background noise addition. (A) High-resolution image (500x500) without background noise., floating point (B) Image in (A) after resampling (250x250), casting to 8-bit, and addition of background noise. See text for details of simulation parameters.

background noise Random noise is added to every pixel. The resulting image with its higher resolution counterpart can be seen in Figure 15.

Because there are many parameters it is useful to view test images before creating the entire sequence of images. This option is available to the user along with the additional ability to see what a single airy disk looks like given the current relevant user settings. This allows the user to minimize the box surrounding the airy disk and thus speed up subsequent simulations. Pressing “Test this set up” produces a microtubule while “see this Airy Disk” shows what a single airy disk looks like. The maximum intensity of the airy disk can be controlled with the “I0.” This only works with “See this Airy Disk” since the sum of all the airy disks is normalized when constructing the microtubule image. These buttons can be seen in Figure 3 below the “Let’s Make Some Magic” button.

Trajectory

The ultimate goal of this software is to produce a series of images that mimic microtubule motion in gliding motility assays. This software contains a subVI that allows the user to create a trajectory that the microtubule will follow throughout the frames. The trajectory can be composed of four different base trajectories; a horizontal line, a vertical line, a sloped line, and a circle. The user can manipulate these simpler paths to create elementary or more complicated trajectories as seen in the front panel image of this subVI in Figure 16.

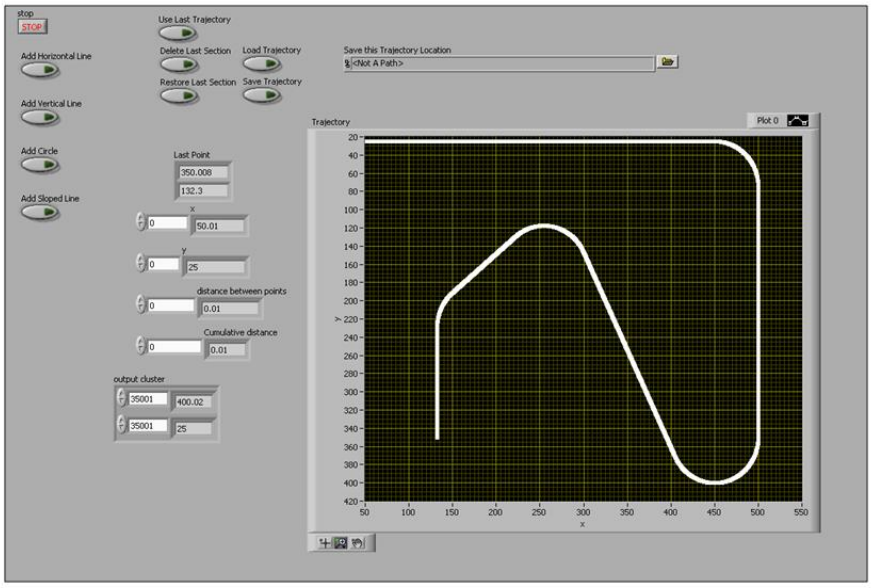

Figure 16: Front panel of the trajectory subVI. This VI allows the user to design a trajectory the microtubule will follow. The user can add horizontal lines, vertical lines, circles, and sloped line to the trajectory. The user can also load previous trajectories and edit them. Trajectories can be saved as files. The current trajectory is shown to the user in the graph seen on the right. See text for further explanation.

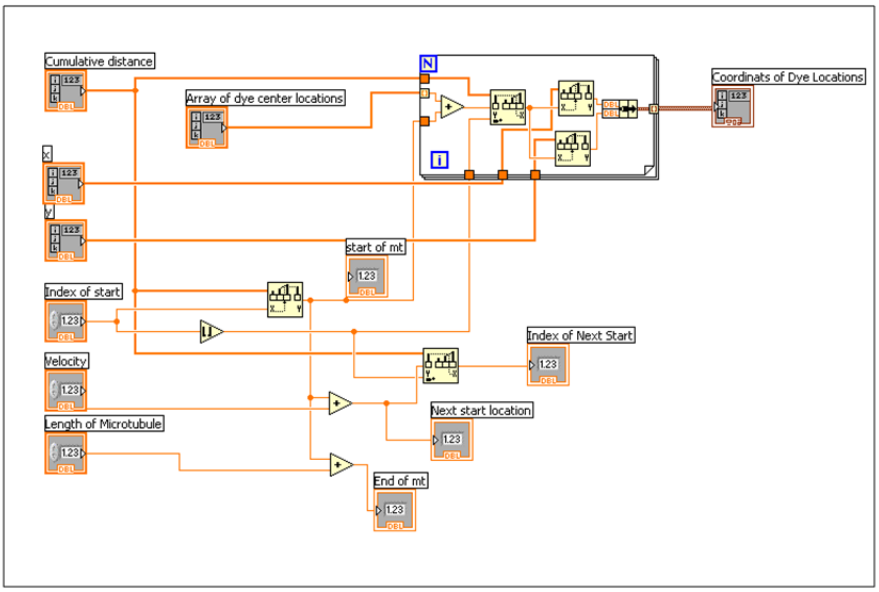

Figure 17: Calculation of dye molecule locations. This code calculates the coordinates of the dye locations, the start of the microtubule in the next frame, that point's index along the trajectory, and the end of the microtubule in this frame. The coordinates of the dye locations are calculated by finding a fractional index along the trajectory that corresponds to each dye's distance from the start of the microtubule. This is done by mapping the trajectory onto a straight line using the cumulative distance array (*top right*). The start of the microtubule and the index to start the microtubule in the next frame are calculated using the specified speed value. The end of the microtubule is found by adding "length of microtubule" to the start value. The program stops creating images when the end of the microtubule reaches the end of the trajectory.

The simplest paths are the horizontal and the vertical line. For the horizontal line, the user inputs the starting point coordinate (x,y) and the ending x coordinate. The vertical line is very similar in that it needs the user to input the starting point coordinate (x,y), but it needs the ending y coordinate instead of x. The sloped line needs both the x and y coordinates for the starting and ending points. It will output the slope of the line for reference. For a circular arc, the user specifies the origin, radius, and angles to start and end the arc. These angles can vary from -2π to 2π.

To make creating more complicated trajectories easier, each one of these paths have an option to automatically link to the existing path. For the horizontal and vertical line this only means that the starting point is set automatically to be the last point of the semi-completed path. For the sloped line, the starting point is set just like the horizontal line, but the slope is also set to the slope of the incoming path. It won't set the slope automatically if the incoming slope is horizontal or vertical. To best attach a circle to the trajectory, the program can set the starting point on the circle so the incoming trajectory has the same slope as the tangential line to the circle. This ensures a smooth transition into the circle. Since there are two points on the circle with the same slope, the user can choose which point to use.

This trajectory subVI also allows the user to load a trajectory previously made and edit that path. The user can also save the completed trajectory as a .dat file in a chosen directory. The main program will automatically save the trajectory in the same directory as the images.

Speed

User Input:

See Figure 5

The microtubule will have a fixed speed (see "Updates" section at end of manuscript), defined by the user in units of high res pixel/frame. In the case shown in Figure 5 the user has chosen a speed of 2 high res pixels/frame. The amount of resampling will determine the speed in the final, lower resolution images. With the settings shown earlier in Figure 5, the tracked speed is 1 low res pixels/frame.

In each image frame, the microtubule dye molecules coordinates will be moved along the trajectory a distance specified by the speed. Since the airy disks are labeled by their distance from the start of the microtubule and not from a location on the image, their relative distances remain unchanged despite curves along the trajectory. The program calculates the distances between adjacent points on the trajectory. Using this array of distances it is able to set the front of the microtubule and quickly search for points that are certain distances away from the start. The code is shown in Figure 17.

Finally, when satisfied with the settings and trajectory, the user presses the "Let's Make Some Magic" button which can be seen in Figure 3 to the right of Image Size parameters. The program runs and saves images until it reaches the end of the trajectory. Currently the user cannot stop the simulations early. Pressing the stop button will stop the entire program but will only be responsive after all the images have been created.

Saving images and user settings

User Input:

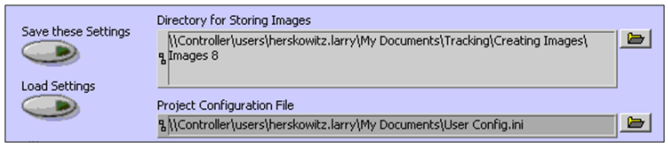

Figure 18: Settings and file information. "File Path" specifies the directory in which the images and other information will be saved. "Save Settings" or "Load Settings" will save / load the current settings to / from the file specified by "Project Configuration File."

This software saves the images in a directory that is specified in "Directory for Storing Images." The images are .pngs and are named after the frame number. The parameters used to create the microtubule as well as the trajectory are saved. The parameters are saved in an .ini file while the trajectory is saved as a .dat file.

This speed of this software is dependent on the user parameters. For example a higher number of photons will slow down the algorithm proportionally. A larger box around the center of the airy disk will also slow down the simulation. With the settings shown in this paper, the software creates an image in about 350 milliseconds on our Intel Core 2 Duo CPU running Windows XP.

Section 2

How to obtain code and a video tutorial

This code is available from sourceforge at https://sourceforge.net/projects/simulatingimage/files/. The VIs used to create this program are available for download in the Simulating Images folder. LabVIEW and the Vision Development Module are required to view and edit the source code. An executable version of this program is also available in Executable Folder found in the Simulating Images directory. A $420 "NI Vision Development Module Run-Time License" is required to run the executable. http://sine.ni.com/nips/cds/view/p/lang/en/nid/207700

A video tutorial created by CamStudio is also available in the Tutorial Folder in the same directory. The .ini file that saved the settings used to create the images found in this paper and the trajectory shown above are located in Settings for Paper folder inside this directory.

Prior Attempts

This report describes our second attempt to create images of microtubules. The first attempt followed more closely to the steps taken in the Cheezum paper (13). We found the stochastic method we report here more satisfying as far as mimicking our experimental data. However the other process, based on convolutions of a line or rectangle with an airy disk worked well. We do not describe those methods here, but out work can be seen in LJH's open notebook, dates 11/12/2009 through 11/18/2009.

Updates

Since writing this pre-print, we have added features to this software. For example, the ability to pause or switch speeds. Also, the ability for speed to vary according to Poisson stepping. The updates may be reflected in the code and thus some figures may be slightly outdated.

Conclusion

This software can create a series of images of a microtubule moving along a user specified trajectory. This can be used to test tracking software designed for gliding motility assays or other microtubule assays. It is possible to adapt this software to create images of other polymeric protein structures such as f-actin and some cytoskeletal proteins (20; 21). However it is not equipped to handle these yet. The code is freely are available at SourceForge under an MIT license for reuse and adaptation.

1. Goldstein L, Yang Z. Microtubule-based transport systems in neurons: the roles of kinesins and dyneins [Internet]. Annual review of neuroscience. 2000 ;23(1):39–71.Available from: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.neuro.23.1.39

2. Vale RD, Fletterick RJ. The design plan of kinesin motors. [Internet]. Annual review of cell and developmental biology. 1997 ;13745-77.Available from: http://www.ncbi.nlm.nih.gov/pubmed/9442886

3. Wittmann T, Hyman a, Desai a. The spindle: a dynamic assembly of microtubules and motors. [Internet]. Nature cell biology. 2001 ;3(1):E28-34.Available from: http://www.ncbi.nlm.nih.gov/pubmed/11146647

4. Vale RD, Reese TS, Sheetz MP. Identification of a novel force-generating protein, kinesin, involved in microtubule-based motility. [Internet]. Cell. 1985 ;42(1):39-50.Available from: http://www.ncbi.nlm.nih.gov/pubmed/3926325

5. Greene L, Henikoff S. Kinesin Home Page [Internet]. Available from: http://www.cellbio.duke.edu/kinesin/

6. Clemmens J, Hess H, Lipscomb R, Hanein Y, Böhringer KF, Matzke CM, et al. Mechanisms of Microtubule Guiding on Microfabricated Kinesin-Coated Surfaces: Chemical and Topographic Surface Patterns [Internet]. Langmuir. 2003 ;19(26):10967-10974.Available from: http://pubs.acs.org/doi/abs/10.1021/la035519y

7. Dennis JR, Howard J, Vogel V. Molecular shuttles : directed motion of microtubules along nanoscale kinesin tracks. Nanotechnology. 1999 ;232

8. Hess H, Clemmens J, Matzke C, Bachand G, Bunker B, Vogel V. Ratchet patterns sort molecular shuttles [Internet]. Applied Physics A: Materials Science & Processing. 2002 ;75(2):309-313.Available from: http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s003390201339

9. Hess H, Howard J, Vogel V. A Piconewton Forcemeter Assembled from Microtubules and Kinesins [Internet]. Nano Letters. 2002 ;2(10):1113-1115.Available from: http://pubs.acs.org/doi/abs/10.1021/nl025724i

10. Howard J. The movement of kinesin along microtubules [Internet]. Annual review of physiology. 1996 ;58(1):703–729.Available from: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.ph.58.030196.003415

11. Sturrman N. MTrack2 [Internet]. 2009 ;Available from: http://valelab.ucsf.edu/~nico/IJplugins/MTrack2.html

12. Chisena EN, Wall RA, Macosko JC, Holzwarth G. Speckled microtubules improve tracking in motor-protein gliding assays. [Internet]. Physical biology. 2007 ;4(1):10-5.Available from: http://www.ncbi.nlm.nih.gov/pubmed/17406081

13. Cheezum MK, Walker WF, Guilford WH. Quantitative comparison of algorithms for tracking single fluorescent particles. [Internet]. Biophysical journal. 2001 ;81(4):2378-88.Available from: http://www.ncbi.nlm.nih.gov/pubmed/11566807

14. Labview Tutorial 1 [Internet]. 2010 ;Available from: http://www.youtube.com/watch?v=Em5R_RM8E08

15. About Tubulin [Internet]. 2010 ;Available from: http://www.cytoskeleton.com/products/tubulins/abouttub.html

16. Wolf E. The diffraction theory of aberrations [Internet]. Reports on progress in physics. 1951 ;14(1):95–120.Available from: http://www.iop.org/EJ/abstract/0034-4885/14/1/304

17. Davidson MW. Numerical Aperture and Image Resolution [Internet]. 2010 ;Available from: http://www.microscopyu.com/tutorials/java/imageformation/airyna/

18. Watermanstorer C, Salmon E. How Microtubules Get Fluorescent Speckles [Internet]. Biophysical Journal. 1998 ;75(4):2059-2069.Available from: http://linkinghub.elsevier.com/retrieve/pii/S0006349598776489

19. Voter A. Introduction to the kinetic Monte Carlo method [Internet]. In: Radiation Effects in Solids. Citeseer; 2007. p. 1568–2609.Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.3560&rep=rep1&type=pdf

20. Danuser G, Waterman-Storer C. Quantitative fluorescent speckle microscopy: where it came from and where it is going [Internet]. Journal of Microscopy. 2003 ;211(October 2002):191–207.Available from: http://www3.interscience.wiley.com/journal/118870697/abstract

21. Vallotton P, Ponti A, Salmon ED, Waterman-storer CM, Danuser G. Recovery, visualization, and analysis of actin and tubulin polymer flow in live cells: a fluorescent speckle microscopy study. Biophys. J. 2003 ;851289-1306.

22. Binary Search 1D Array Function for Labview [Internet]. Available from: http://zone.ni.com/devzone/cda/epd/p/id/220.

23. Herskowitz, L., Maloney, A., Koch, S. *Open Source Microtubule Tracking Algorithm*. In preperation