Tutorial for MitoQuant

1. Brief introduction

MitoQuant toolkit contains *MiTracker* and *MPA*, along with other functions such as image importing and exporting. The *MiTracker* is designed to locate mitochondria in a 4-D image sequence (*xyz-t*) and link their coordinates to 3-D trajectories, which consists of four major functions. The *ParticleEnhancement* function performs image enhancement for later segmentation step. the *MitoDetect* function identifies mitochondrial particles from background. The *DetectionLink* function and the *TrackRectify* function generates and rectifies movement trajectories respectively.

MPA is designed to extract motion pattern information from the mitochondrial trajectories that *MT* generated. It has one major processing function and four result-output functions to perform the transient speed analyses of mitochondrial movement. The *MitoKymograph* function plots the 3D-kymopraph of moving mitochondria, and the *MitoTrajectories* function plots 3D trajectories of mitochondrial movement. The *StatisticsExport* function generates the statistical results based on the transient model of mitochondrial movement. *Speedmap* is used to plot the map of transient speed and sustained speed.

Figure 1. Functional components of MitoQuant

Type "`help`" and "`function name`" to check the usage of the functions of *MitoQuant*. For instance,

```
>> help tif2mat
tif2mat(filename,framinterval) converts TIFF image to
4-D matrix for MiTracker.
    INPUTS
    filename : filename of the TIFF image which
               contains mitochondrial moving image
               data.
    frameinterval: the frame interval in second (s).

    OUTPUTS
    I        : a structure which contains image data
               and other information, including
               following fields:
               data -- the image data, a 4-D matrix.
               info -- the file description
               xResolution -- the resolution in x
               direction. (pixel/um)
               yResolution -- the resolution in y
               direction. (pixel/um)
               framesNum -- number of temporal frames.
               layersNum -- number of z-layer.
```

2. Installation

Please download the MitoQuant package first. Unzip the package, where you will find the MitoQuant toolkit, and this tutorial. For demonstration purpose, a sample image is provided at http://ese.nju.edu.cn/yogo/sampleimage.zip. To install the MitoQuant, simply use `path` to find MATLAB's toolbox directory in your system and copy MitoQuant directory there (copy and paste with the file browser, such as Finder in Mac OS and Windows Explorer). For instance, in Mac OS, the directory usually is located at:

```
>> path
/Applications/MATLAB_R2014a.app/toolbox/
```

Before using MitoQuant, you should add the path of MitoQuant to MATLAB's default path. In command window, use `addpath` command to add the MitoQuant to MATLAB default path. Then `savepath` it, as shown below.

```
>> addpath('/Applications/MATLAB_R2014a.app/toolbox/MitoQuant')
>> savepath
```

To test if the installation is successful, type "`help tif2mat`" in command window. Once you see the help message, this confirms the installation is done.

3. Preparing the image sequences.

MitoQuant only accepts 4-D image sequences in Tag Image File Format (TIFF), which should be exported by ImageJ. To prepare the TIFF file, you should open your images with ImageJ and *OME Bio-Formats* plugins. In our experiment, the Leica SP8 saved the images as LIF files (extension is .lif). A single LIF file may contain several image serials. If so, choose the one you want to analyze and export the TIFF file.

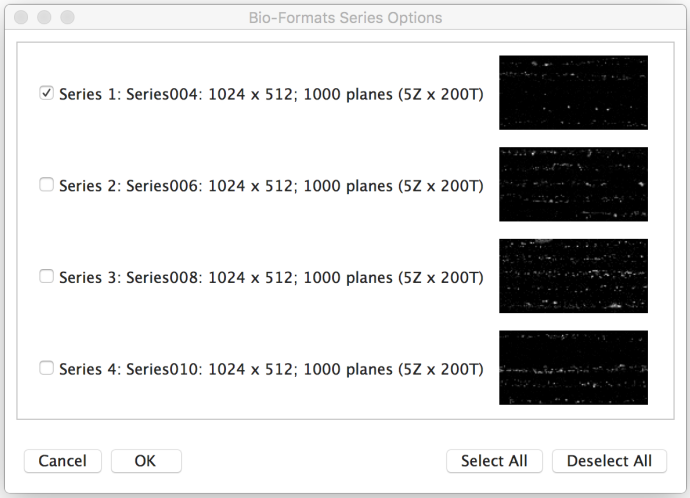

Figure 2. Leica LIF file contains multi image series. Choose the one you want to analyze.

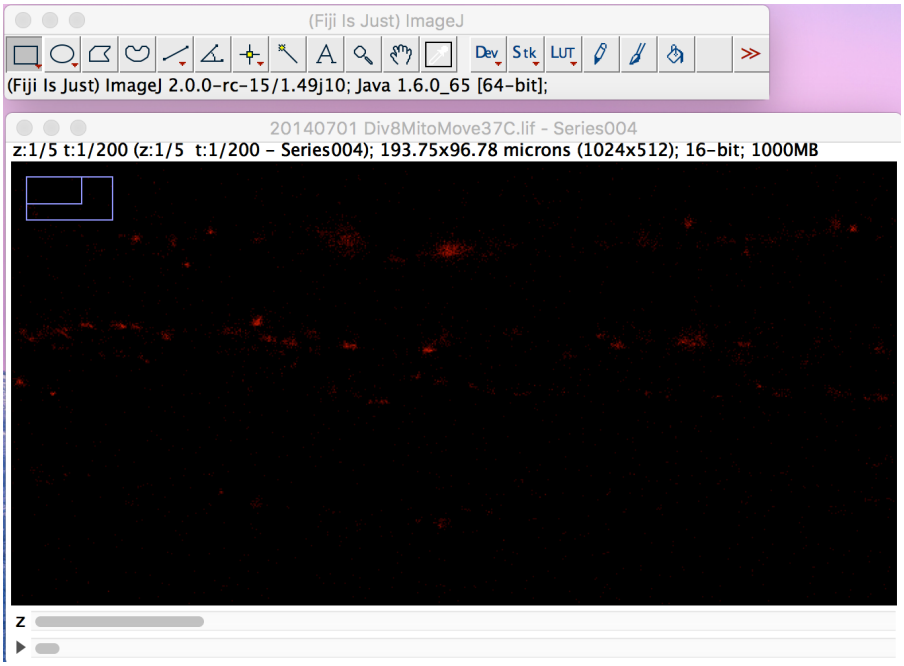

Figure 3. The 4D (xyz-t) images of moving mitochondrial opened in ImageJ.

Then choose **File** -> **Save as** -> **Tiff** to export the image file.

4. Image Loading and Preprocessing

Open MATLAB and use *tif2mat* function to open the image sequences. For instance, we use variable `Io` to store the original image as below. The spatial resolution and other information have been included into the TIFF file. Therefore, only the frame interval is required when opening the image. Here, the second argument frame interval is set to `1.5`, because we took z-stack images every 1.5 second during our microscopy imaging procedure.

```
>> Io = tif2mat('test.tif', 1.5);
Reading the image...
Done.
```

The goal of preprocessing is to improve the original image for segmentation. We use the *ParticleEnhancement* function to improve the signal to noise ratio (SNR) of the image. The function requires two arguments: the input original image and the particle size. Here, the argument `5` is the mitochondrial size in pixels for this imaging experiment. To estimate the average size of mitochondria in pixels, you may open the original image with ImageJ and count the pixels manually.

```
>> I = ParticleEnhancement(Io,5);
Particle enhancement filtering may take several minutes to
process.
Processing...
+-------------------------------------------------+
+++++++++++++++++++++++++++++++++++++++++++++++++++
done.
```

Note that this preprocessing step with *ParticleEnhancement* may take several minutes to process. Please be patient.

5. Mitochondrial Detection

MitoDetect function is designed to identify mitochondria in the enhanced image. This function requires two arguments: the input image sequence and a threshold to segment mitochondria from background. MitoQuant provides a graphic tool called *SegmentThresh* to estimate the threshold.

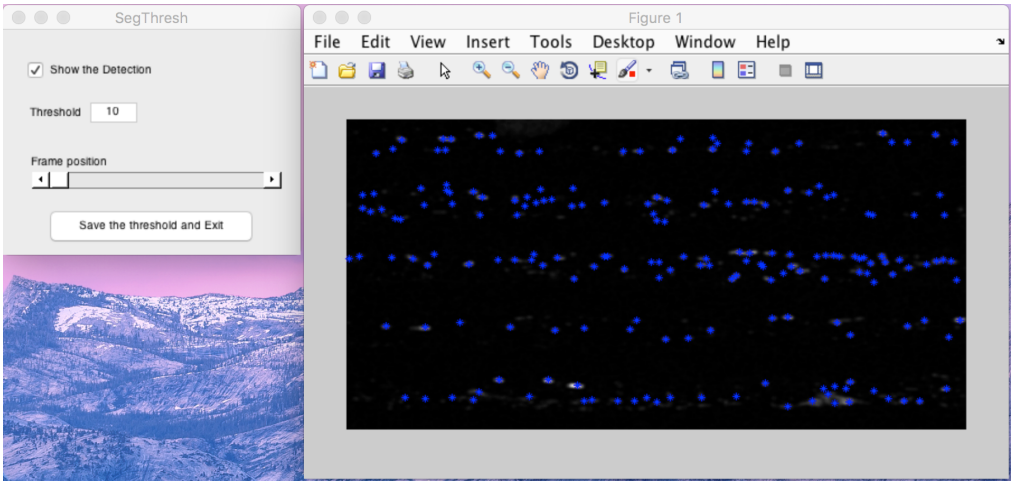

Figure 4. A graphic tool designed to determine the threshold for mitochondrial segmentation.

Open the filtered image with the tool to determine the threshold. Check and uncheck the *show detection* option to inspect the detection results. You are allowed to assign a new threshold to any frame, for best segmentation result. When a threshold is entered, thresholds for all the frames after the current frame will be set with the same value. It allows you to adjust the threshold according to the decreasing SNR, by simply assigning a few thresholds for frames at different time point, such as the first, the last and the middle frame. After finding the proper thresholds, simply click *save the threshold and exit* button. The threshold vector `st` will be save to current directory automatically. You should manually load the threshold vector into workspace. Then use MitoDetect to identify the mitochondria with the threshold vector you just got. Here, variable `D` was used to store detection results.

```
>> load st
>> D = MitoDetect(I,st);
Mito Detecting may take several minutes to process.
Processing...
+-------------------------------------------------+
+++++++++++++++++++++++++++++++++++++++++++++++++++
done.
```

6. Detection linking and rectifying trajectories.

This step links the detected mitochondria and generates 3D Trajectories. Use the *DetectionLink* function to link mitochondrial detections into movement trajectories. The *TrackRectify* function removes the fragmental tracks and links short tracks into longer ones. Rectifying helps *MiTracker* improve the integrity of trajectories. Here variable `D` was the structure storing the detection results from the previous step, and variable `T` was used to store the trajectories.

```
>> T = DetectionLink(D);
Linking the mito detections into trajectories...
Done.
>> T = TrackRectify(T, D);
Processing trajectories rectification...
Linking tracks done.
Done.
```

7. Transient speed analysis.

The *TransSpeedAnalysis* function calculates transient speed and sustained speed. This processing converts 3-D mitochondrial trajectories to points in a 2-D parameter space generated by the two components of mitochondrial speed (transient and sustained). This function requires three arguments: Image `I`, trajectories `T` and analysis window size. The window size has to be $2^k, k \in R$. The typical value may vary between 8 and 32. In our experiment, the window size was set to 16 points, which gave the best compromise between temporal resolution and speed resolution under our imaging conditions.

```
>> V = TransSpeedAnalysis(I, T, 16);
Performing transient speed analysis ...
Done.
```

8. Results output

1) Plot 3-D kymographs

The *MitoKymograph* function plots a 3-D kymograph of the original or enhanced image. Here, the enhanced image created earlier, `I`, was plotted.

```
>> MitoKymograph(I)
```

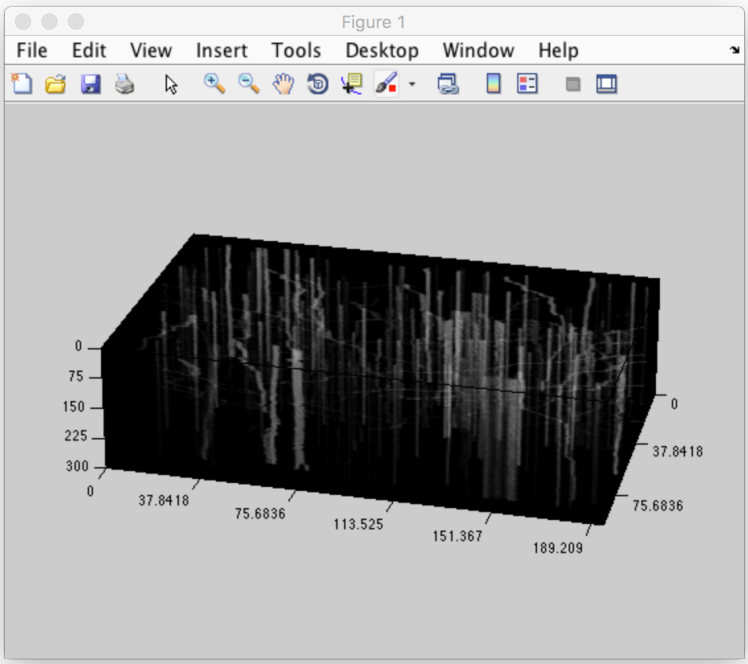

Figure 5. A 3-D kymograph

2) Plot 3-D trajectories

The *MitoTrajectories* function plots mitochondrial movement trajectories. Mitochondria in anterograde motion (AR) and retrograde motion (RR) will be plotted in red and blue color respectively. Dynamic Pauses (DP) will be marked as green and stationary (ST) will be marked as gray. It takes as input the detection `D`, the trajectories `T`, and the speed vector `V`. The optional fourth argument *option* is a 1-by-4 vector, which controls the visibility of four movement states `[ST, DP, AR, RR]` in trajectories map. For instance, `[0 0 1 1]` allows the function to output AR and RR trajectories only. If *option* is not assigned, the function outputs by default all of the trajectories in the four states. For instance, the follow command plots running (retrograde and anterograde) states trajectories only.

```
>> MitoTrajectories(D,T,V,[0 0 1 1])
```

The following command plots all four states trajectories.

```
>> MitoTrajectories(D,T,V,[1 1 1 1])
```

The figure below shows the result of these two commands, with the first on the left and the second on the right.

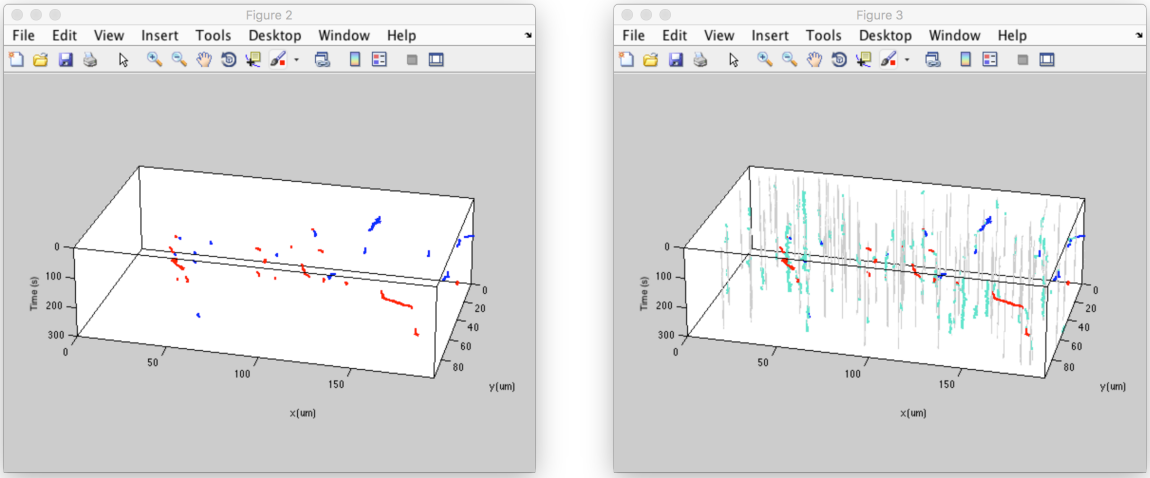

Figure 6. 3-D trajectories generated by MitoTrajectories.

3) Statistical results

The *StatisticsExport* function calculates the statistical results, including the proportion of mitochondria in each movement state and the average running speed. The input argument list of this function includes a threshold to distinguish AR, DP and RR from each other. In our experiment, the threshold is set to 0.05um/s. For more details, please refer to the manuscript.

The output of this function includes:

ProportionST :proportion of mitochondria in movement state of "Stationary";
ProportionDP :proportion of mitochondria in movement state of "Dynamic Pause";
ProportionAR :proportion of mitochondria in movement state of "Anterograde Running";
ProportionRR :proportion of mitochondria in movement state "Retrograde Running";
AvgSpdAR :average running speed of mitochondria in movement state "Anterograde Running";
AvgSpdRR :average running speed of mitochondria in movement state "Retrograde Running"

Here is an output example of this function.

```
>>  StatisticsExport(V,0.05)

ans =

        TotalNum: 3238
    ProportionST: 0.9021
    ProportionDP: 0.0778
    ProportionAR: 0.0105
    ProportionRR: 0.0096
       AvgSpdAR: 0.1266
       AvgSpdRR: -0.1032
```