TFMatlab -Instructions

Table of Contents

[Basic Usage 2](#_Toc535312622)

[Custom Images and Colormaps 7](#_Toc535312623)

[View saved TFM Data 7](#_Toc535312624)

[Trouble shooting bead images 8](#_Toc535312625)

[Variable Description 9](#_Toc535312626)

[Example Custom Images 11](#_Toc535312627)

[Displaying only mag(Strain) 11](#_Toc535312628)

[Displaying only cell image 12](#_Toc535312629)

[Display cell with Force vectors 13](#_Toc535312630)

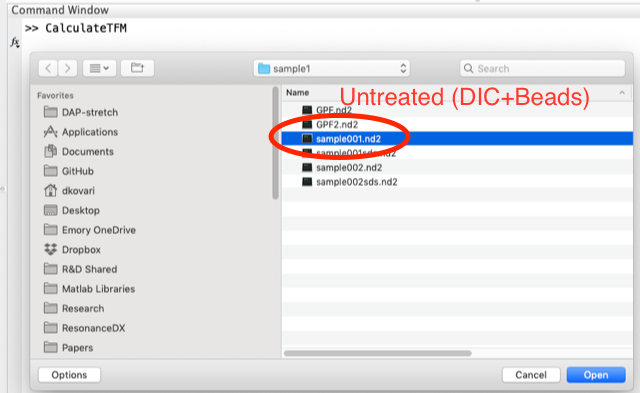
[Custom Animations 14](#_Toc535312631)

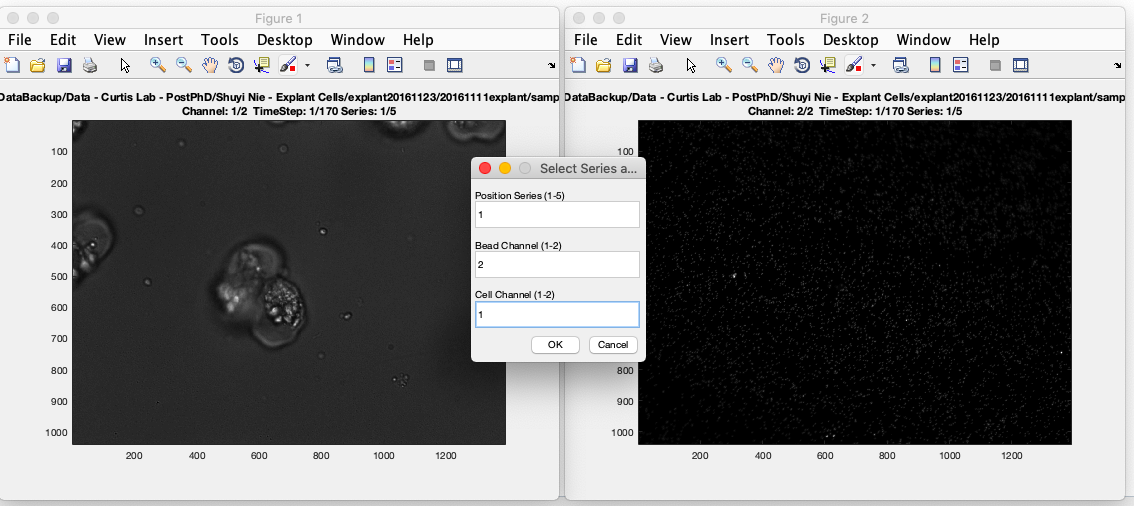
# Basic Usage

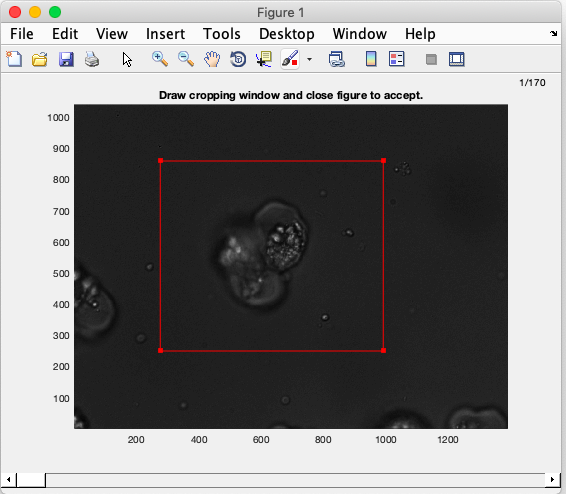
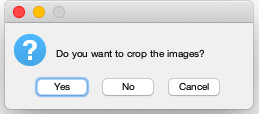
Using CalculateTFM (or CalculateTFM\_piv):

Note: CalculateTFM uses particle tracking to determine bead displacement, CalculateTFM\_piv use “particle image velocimetry” to determine bead displacement. Depending on image quality, one may work better than the other. Give both a try.

1) run the script and select your nd2 file for the untreated cells (no trypsin)



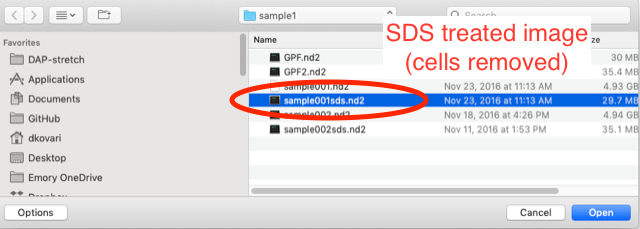
2) Choose the channels (DIC-> Cell channel, Fluorescence->beads)

3) (optional) crop images 

4) When prompted to select reference frame choose “Alternate Image”

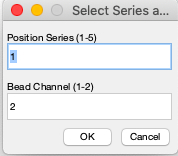


select your trypsin treated image (in my case I used SDS instead of trypsin)

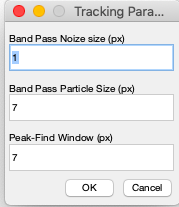


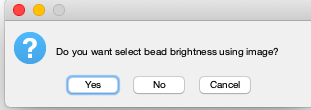
Alternatively, if the trypsin images are part of the same ND2 file as the other images (e.g. the last frame) then select “Choose Frame” and type in the appropriate frame number (i.e. the last frame)

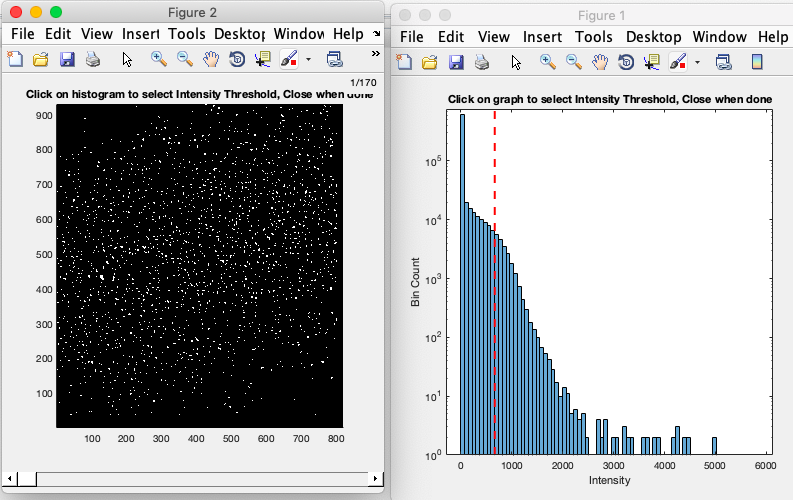
5) confirm position/channel info for trypsin



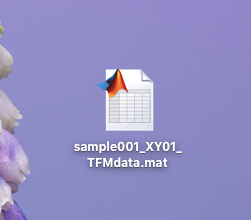
6) set particle tracking parameters (default are probably fine)



7) Select bead brightness cut-off for particle tracking



8) Save your results to mat file



# Custom Images and Colormaps

I chose the Red-overlay because it allowed me to condense create one image (cell and TFM) that showed all the info, as oppose to taking up valuable figure space with separate images for cells and TFM. It also allowed me to generate nice looking movies (which you can do from the view window using the “Save Movie” menu).

After you have computed the TFM data you can do further analysis using the data saved in the MAT file

## View saved TFM Data

1) Load the saved MAT file

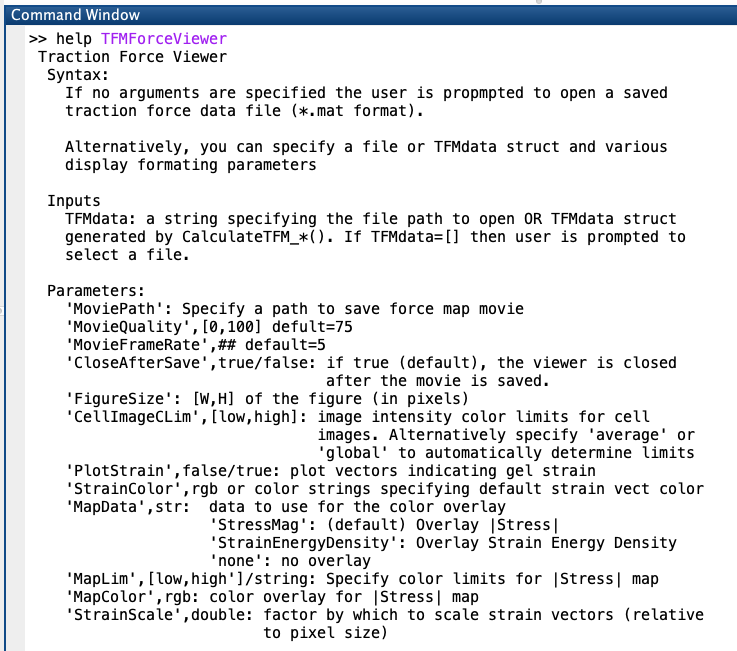
Notice: you will now have a structure variable in your workspace called TFMdata

That structure contains all of the info generated during image analysis

If you want to display the Traction force overlay window again just call:



There are several options you can specify to change how the figure appears



Currently I have not implemented the ability to overlay an arbitrary colormap. Instead you can only overlay a single color (lower map-lim -> transparent, upper map-lim-> non-transparent ‘MapColor’

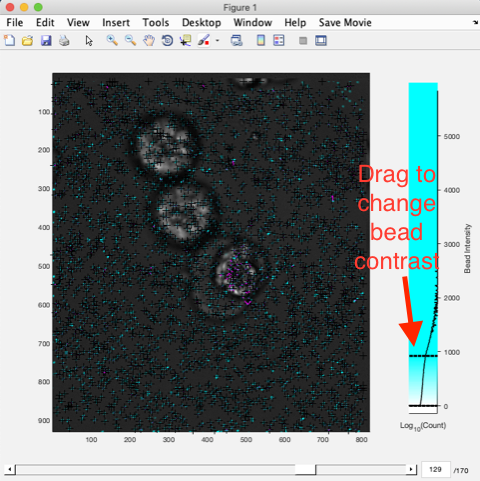
For example, if you want to overlay in magenta instead of red you would write

* TFMForceViewer(TFMdata,’MapColor’,[1,0,1]);

## Trouble shooting bead images

If for some reason the TFM maps don’t look reasonable, a good place to start is looking at what happened to the beads. You can view the “drift-corrected" bead images, overlayed on top of the cell images using

* TFMBeadViewer(TFMdata)

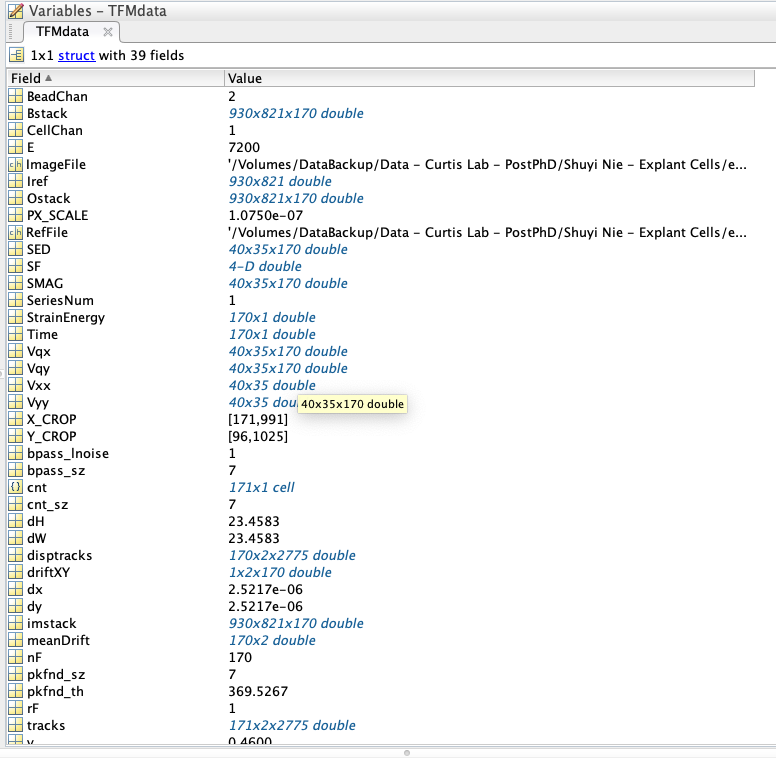


The tracker markers are black +, the displacement are magenta ->

Bead image signal is overlayed in cyan. You can drag the the lines on the colorscale to change the contrast

## Variable Description

If you want to generate custom displays or figures, your best bet is to work with the variables in TFMdata



Here’s a key for what each of those variables mean

**BeadChan**: Channel in ND2 file containing beads

**Bstack**: Stack of grayscale images for the beads after image filtering [Width X Height X nFrames]

**CellChan**: Channel in ND2 file containing cells

**E**: Young’s modulus of gel

**v**: poisson’s ratio

**ImageFile**: path to original ND2 file

**Iref**: Grayscale bead reference image [Width X Height]

**Ostack**: stack of grayscale images corresponding cell channel  [Width X Height X nFrames]

**PX\_SCALE**: Pixel scale (µm/px)

**RefFile**: path to ND2 file for reference image

**SED**: strain energy density [TFMgridX x TFMgridY x nFrames]

**SF**: XY-vector stress field of TFM results [gridX x gridY x 2 x nFrames]

**SMAG**: magnitude of stress = mag(SF): [gridX x gridY x nFrames]

**SeriesNum**: series position index in ND2 file

**StrainEnergy**: StrainEnergy, scalar total energy exerted at a give timepoint [nFrames x 1]

**Time**: timepoint for each frame [nFrames x 1]

**Vxx**: location of TFM grid coordinates along X [gridX x gridY]

**Vyy**: location of TFM grid coordinates along Y [gridX x gridY]

**Vqx**: x-value of strain displacement vectors [gridX x gridY x nFrames]

**Vqy**: y-value of strain displacement vectors [gridX x gridY x nFrames]

**X\_CROP**: X-coordinates of image crop box

**Y\_CROP**: y-coordinates of image crop box

**disptracks**: time-XY particle tracking coordinates for all beads, after drift correction [nFrames, 2, nBeads]

**driftXY**: XY-time average frame drift, relative to reference image [2, nFrames]

**meanDrift**: same as driftXY

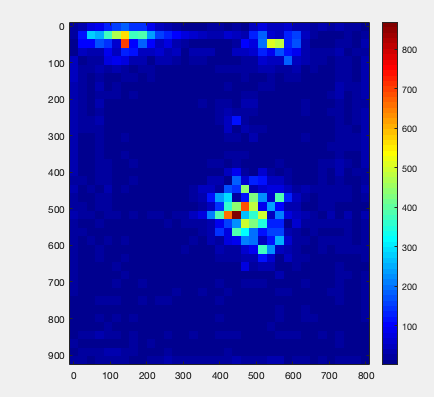
**imstack**: stack of grayscale images of beads (before image filtering)

**tracks**: particle tracks before drift correction [nFrames, 2, num-particles]

## Example Custom Images

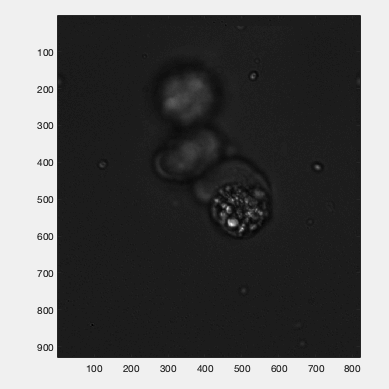
### Displaying only mag(Strain)

* thisFrame = 10; %specify frame
* figure; %create new window
* imagesc([TFMdata.Vxx(1,1),TFMdata.Vxx(1,end)],[TFMdata.Vyy(1,1),TFMdata.Vyy(end,1)],TFMdata.SMAG(:,:,thisFrame)); %display image, rescaled from gridXY to image size
* axis image; %force square pixels on window
* colormap jet; %use ‘jet’ color scheme
* colorbar; %display colorbar



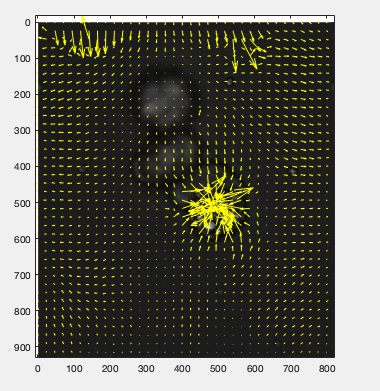
### Displaying only cell image

* thisFrame = 10; %specify frame
* figure; %create new window
* imagesc(TFMdata.Ostack(:,:,thisFrame)); %display image, rescaled from gridXY to image size
* axis image; %force square pixels on window
* colormap gray; %use ‘gray’ color scheme



### Display cell with Force vectors

* thisFrame = 10; %specify frame
* figure; %create new window
* imagesc(TFMdata.Ostack(:,:,thisFrame)); %display image, rescaled from gridXY to image size
* axis image; %force square pixels on window
* colormap gray; %use ‘gray’ color scheme
* hold on; %hold image in window
* forceScale = 5; %scale force vectors by 5
* quiver(TFMdata.Vxx,TFMdata.Vyy,TFMdata.Vqx(:,:,thisFrame),TFMdata.Vqy(:,:,thisFrame),forceScale,'-y’)



### Custom Animations

If you want to make customized videos, composit\_animfig.m is the program at the heart of the TFMForceViewer.

It can overlay any number of RGB color image stacks into a composite animation.

consult help for more info:

* help composit\_animfig