

TFMatlab - Instructions

Table of Contents

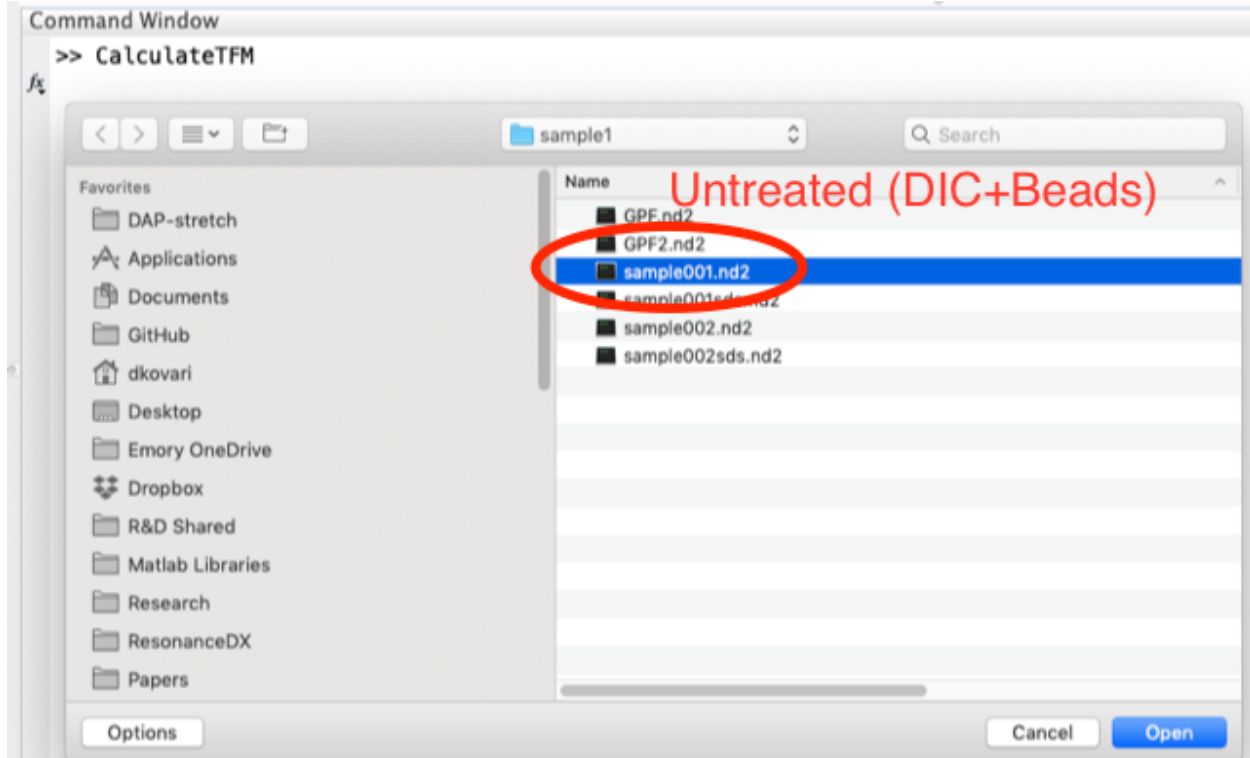
Basic Usage	2
Custom Images and Colormaps.....	8
View saved TFM Data	8
Trouble shooting bead images	9
Variable Description	11
Example Custom Images	12
Displaying only mag(Strain).....	12
Displaying only cell image	13
Display cell with Force vectors	14
Custom Animations	15

Basic Usage

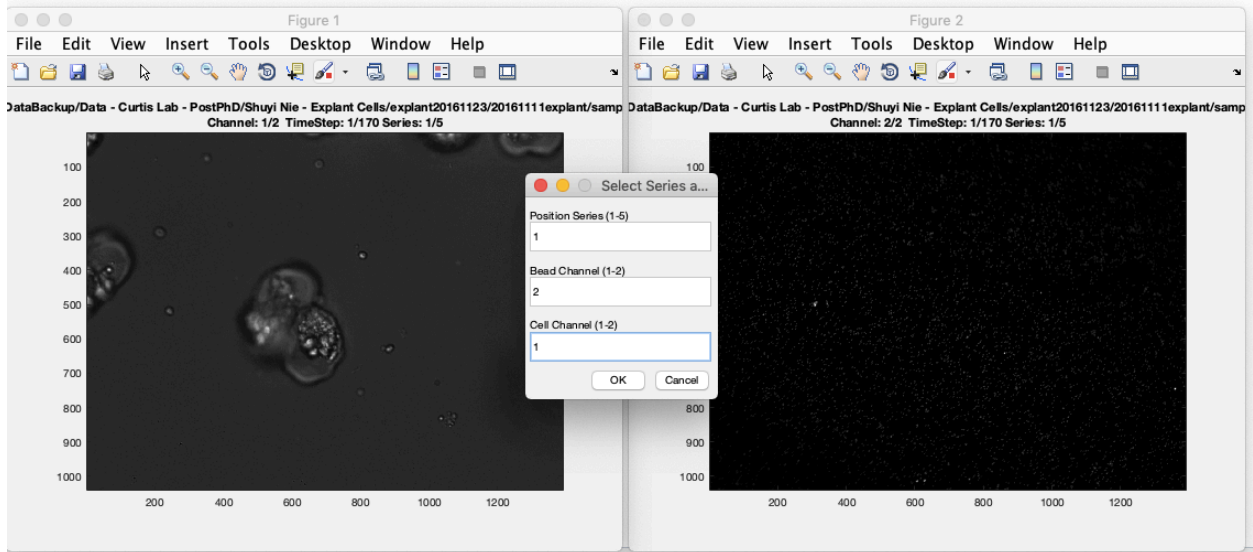
Using CalculateTFM (or CalculateTFM_piv):

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.

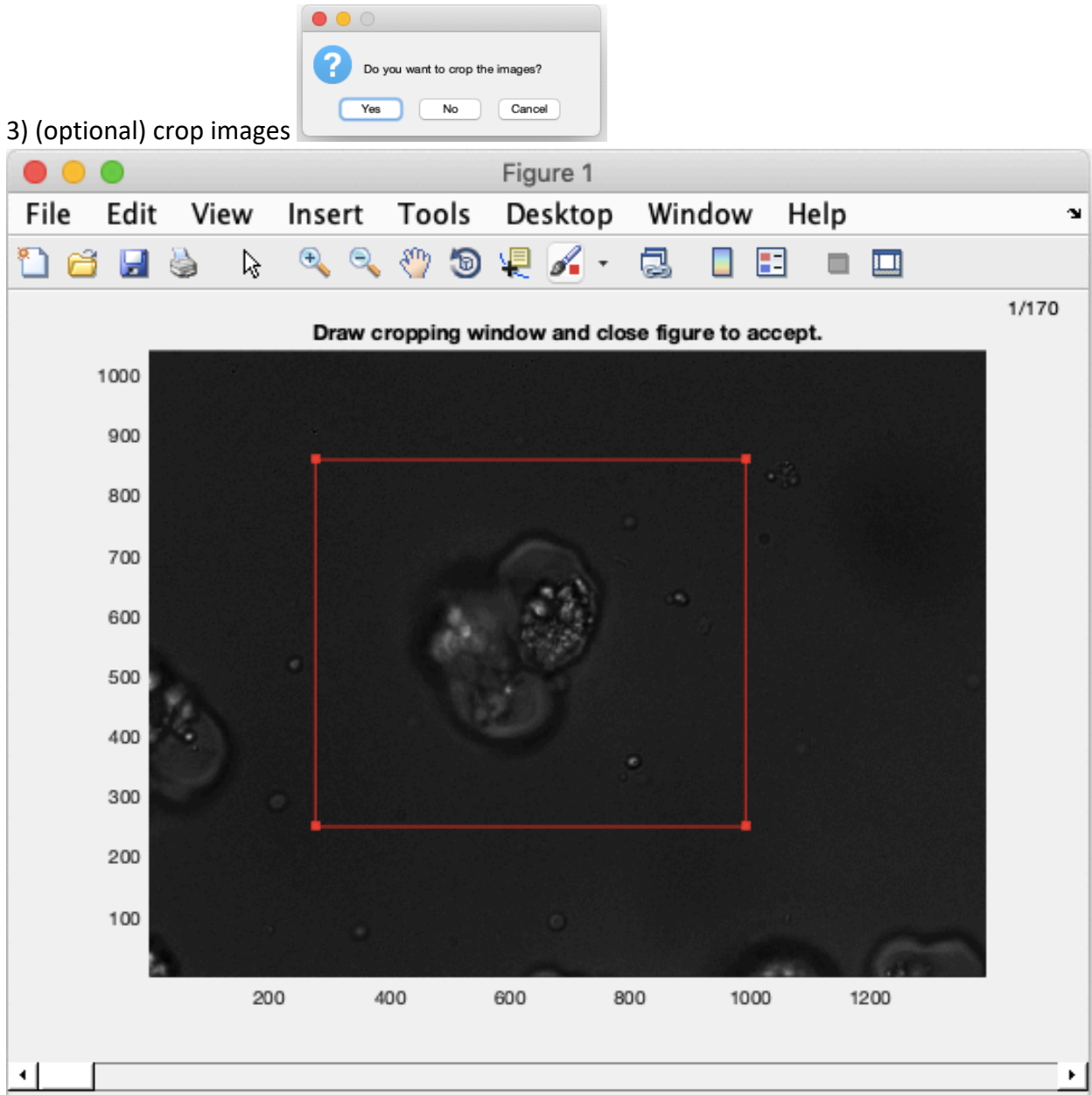
1) run the script and select your nd2 file for the untreated cells (no trypsin)



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



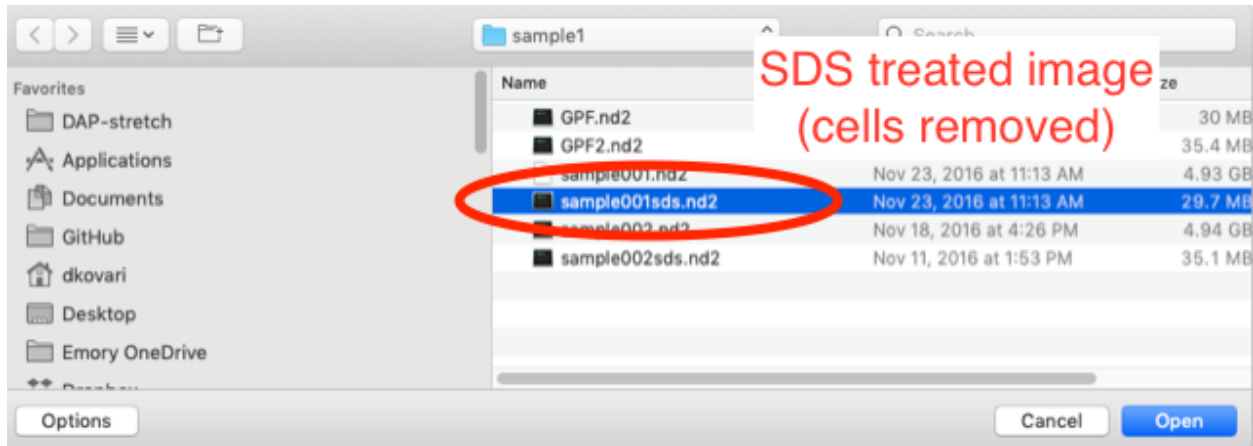
3) (optional) crop images



4) When prompted to select reference frame choose "Alternate Image"

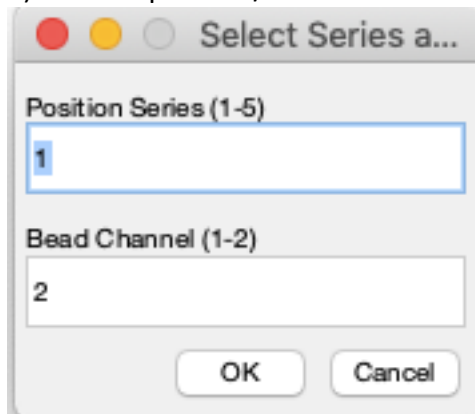


select your trypsin treated image (in my case I used SDS instead of trypsin)

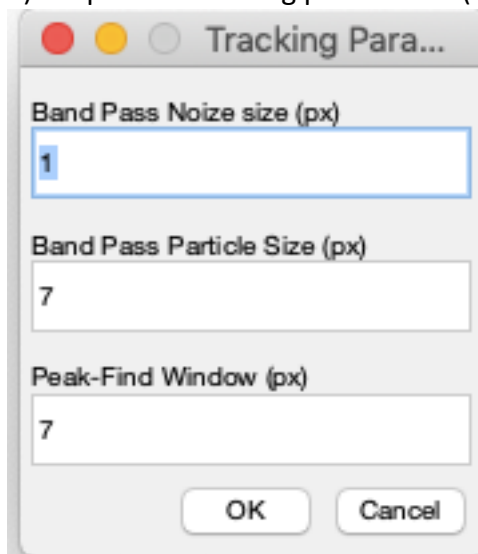


Alternatively, if the trypsin images are part of the same ND2 file as the other images (e.g. the last frame) then select “Choose Frame” and type in the appropriate frame number (i.e. the last frame)

5) confirm position/channel info for trypsin



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



A screenshot of a 'Tracking Parameters' dialog box. The dialog has a title bar with three colored buttons (red, yellow, grey) and the text 'Tracking Para...'. It contains three input fields: 'Band Pass Noise size (px)' with the value '1', 'Band Pass Particle Size (px)' with the value '7', and 'Peak-Find Window (px)' with the value '7'. At the bottom are 'OK' and 'Cancel' buttons.

Tracking Para...

Band Pass Noise size (px)

1

Band Pass Particle Size (px)

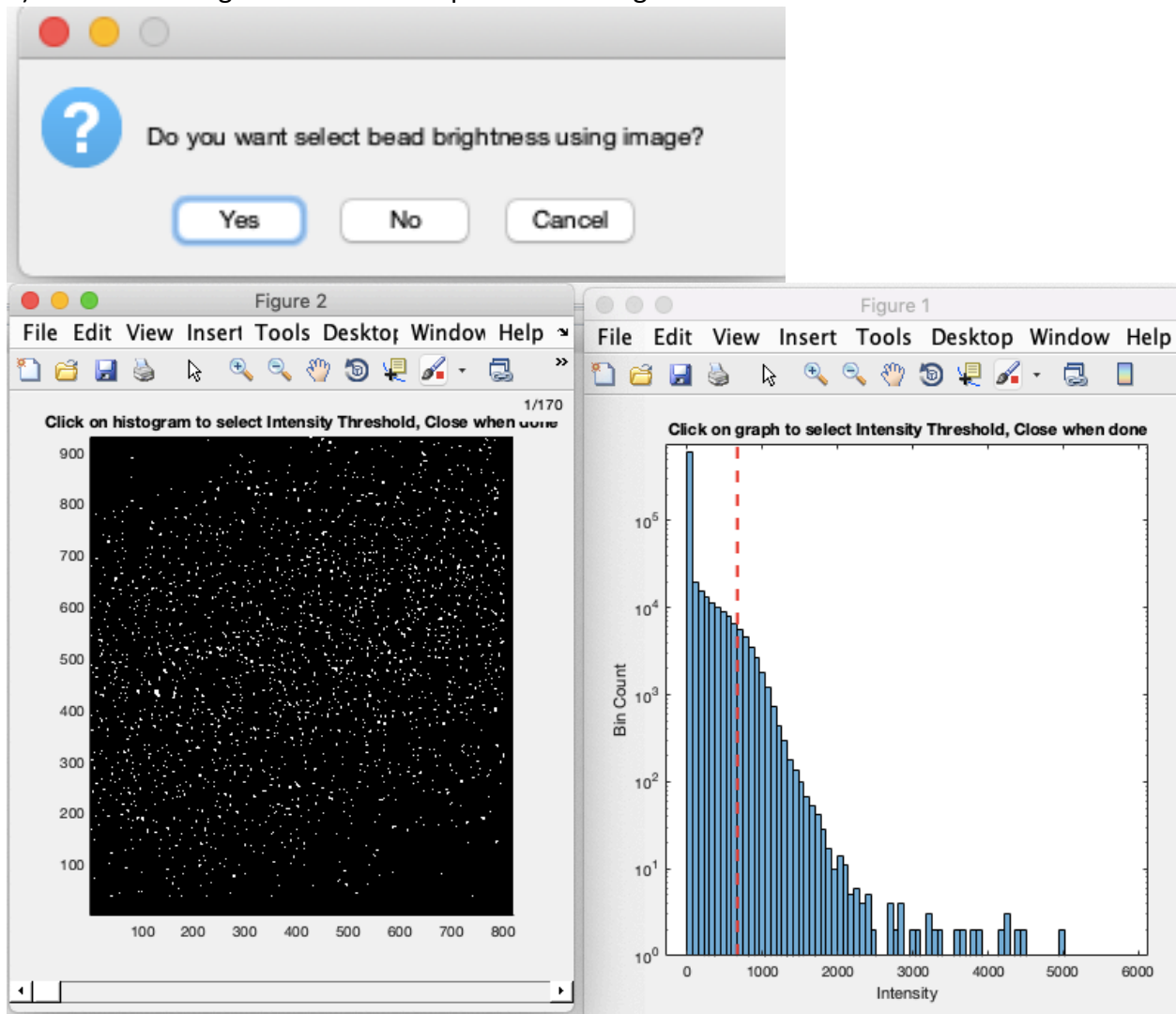
7

Peak-Find Window (px)

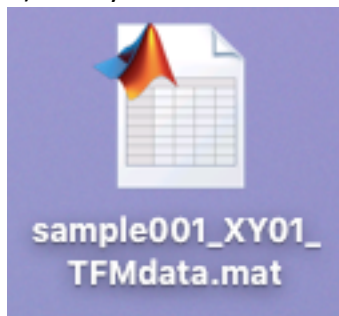
7

OK Cancel

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

Command Window

```
fx >> TFMdata = load('/Users/dkovari/Desktop/sample001_XY01_TFMdata.mat')
```

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:

Command Window

```
fx >> TFMForceViewer(TFMdata)
```


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

Command Window

```
>> help TFMForceViewer
Traction Force Viewer
Syntax:
    If no arguments are specified the user is prompted to open a saved
    traction force data file (*.mat format).

    Alternatively, you can specify a file or TFMdata struct and various
    display formatting parameters

Inputs
    TFMdata: a string specifying the file path to open OR TFMdata struct
    generated by CalculateTFM_*. If TFMdata=[] then user is prompted to
    select a file.

Parameters:
    'MoviePath': Specify a path to save force map movie
    'MovieQuality',[0,100] default=75
    'MovieFrameRate',## default=5
    'CloseAfterSave',true/false: if true (default), the viewer is closed
                                after the movie is saved.
    'FigureSize': [W,H] of the figure (in pixels)
    'CellImageCLim',[low,high]: image intensity color limits for cell
                                images. Alternatively specify 'average' or
                                'global' to automatically determine limits
    'PlotStrain',false/true: plot vectors indicating gel strain
    'StrainColor',rgb or color strings specifying default strain vect color
    'MapData',str:  data to use for the color overlay
                   'StressMag': (default) Overlay |Stress|
                   'StrainEnergyDensity': Overlay Strain Energy Density
                   'none': no overlay
    'MapLim',[low,high]/string: Specify color limits for |Stress| map
    'MapColor',rgb: color overlay for |Stress| map
    'StrainScale',double: factor by which to scale strain vectors (relative
                           to pixel size)
```

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')

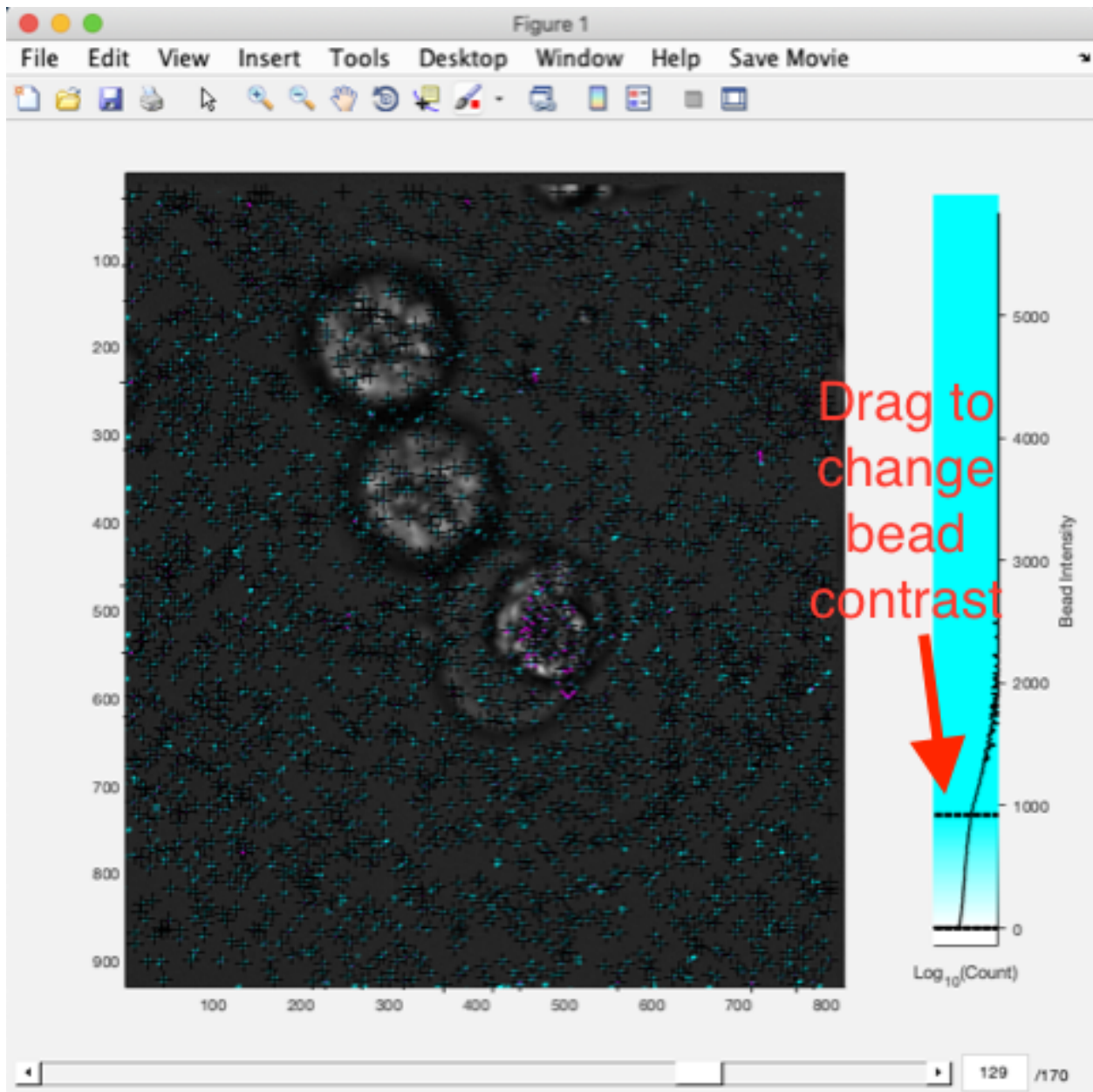
For example, if you want to overlay in magenta instead of red you would write

```
» 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, overlaid on top of the cell images using

```
» TFMBeadViewer(TFMdata)
```

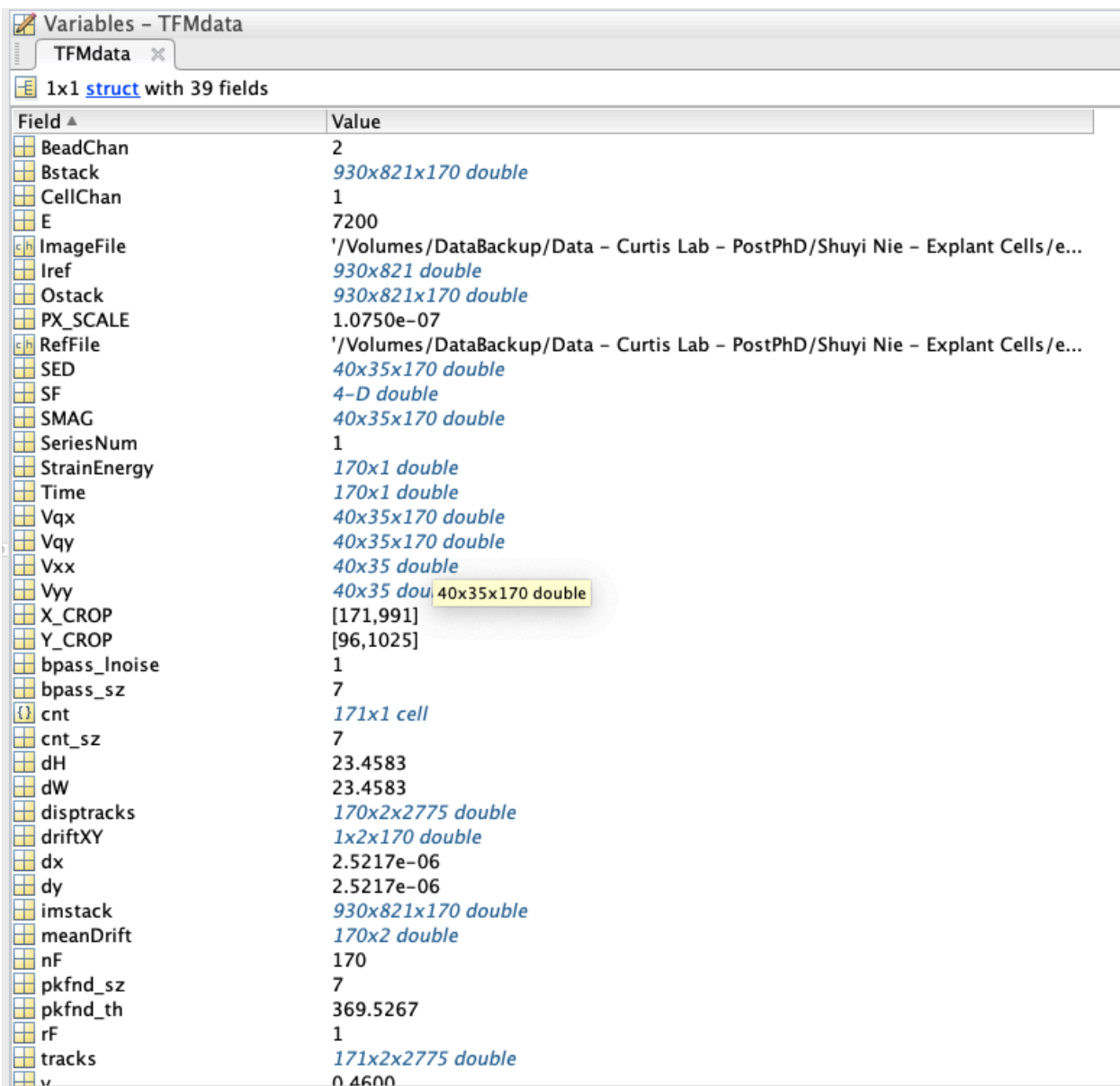


The tracker markers are black +, the displacement are magenta ->

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



Field ▲	Value
BeadChan	2
Bstack	930x821x170 double
CellChan	1
E	7200
ImageFile	'/Volumes/DataBackup/Data - Curtis Lab - PostPhD/Shuyi Nie - Explant Cells/e...
Iref	930x821 double
Ostack	930x821x170 double
PX_SCALE	1.0750e-07
RefFile	'/Volumes/DataBackup/Data - Curtis Lab - PostPhD/Shuyi Nie - Explant Cells/e...
SED	40x35x170 double
SF	4-D double
SMAG	40x35x170 double
SeriesNum	1
StrainEnergy	170x1 double
Time	170x1 double
Vqx	40x35x170 double
Vqy	40x35x170 double
Vxx	40x35 double
Vyy	40x35 double 40x35x170 double
X_CROP	[171,991]
Y_CROP	[96,1025]
bpass_inoise	1
bpass_sz	7
cnt	171x1 cell
cnt_sz	7
dH	23.4583
dW	23.4583
disptracks	170x2x2775 double
driftXY	1x2x170 double
dx	2.5217e-06
dy	2.5217e-06
imstack	930x821x170 double
meanDrift	170x2 double
nF	170
pkfnd_sz	7
pkfnd_th	369.5267
rF	1
tracks	171x2x2775 double
v	0.4600

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 ($\mu\text{m}/\text{px}$)

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 = $\text{mag}(\text{SF})$: [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

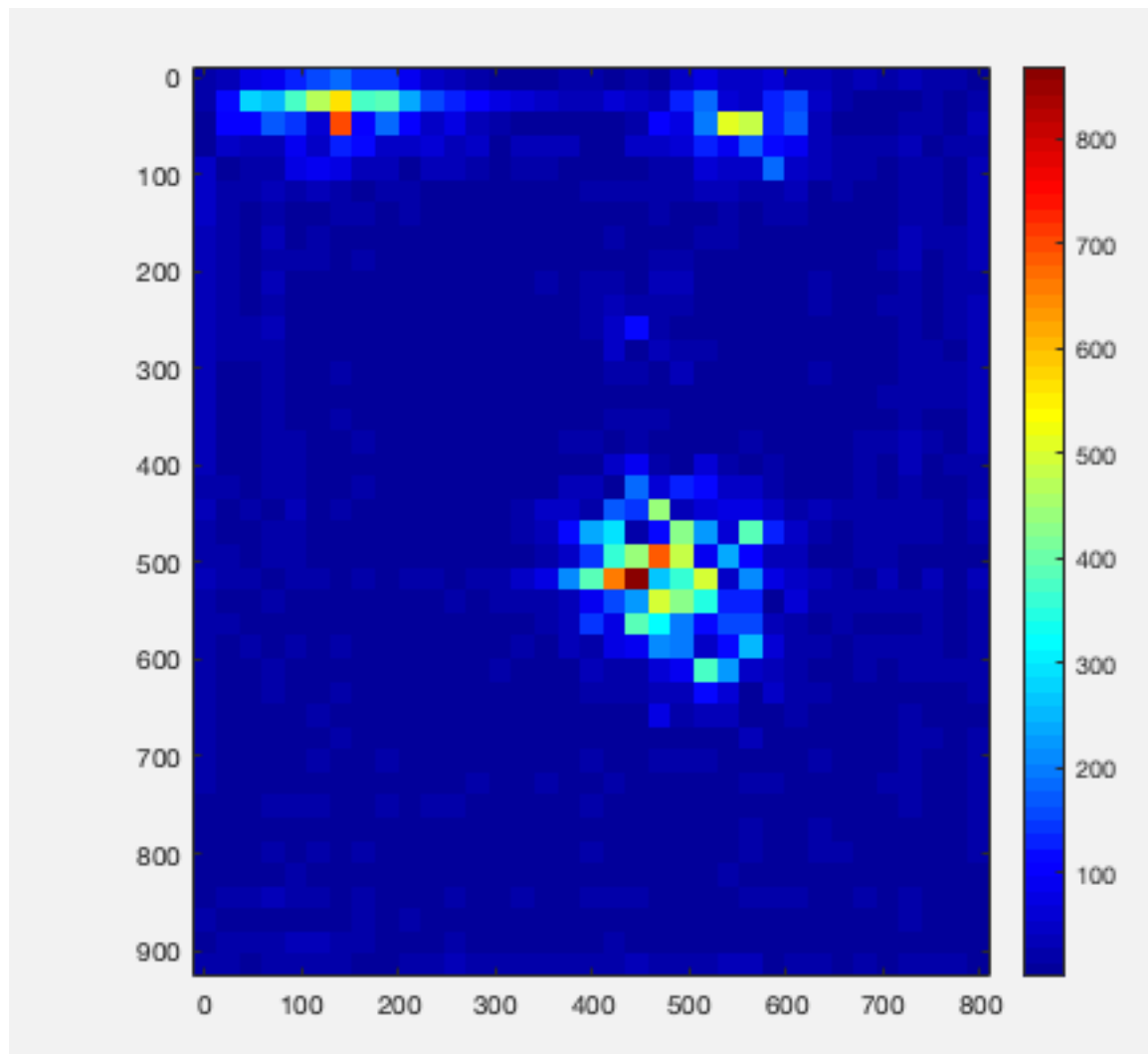
imstack: stack of grayscale images of beads (before image filtering)

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

Example Custom Images

Displaying only mag(Stress)

```
» 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
```

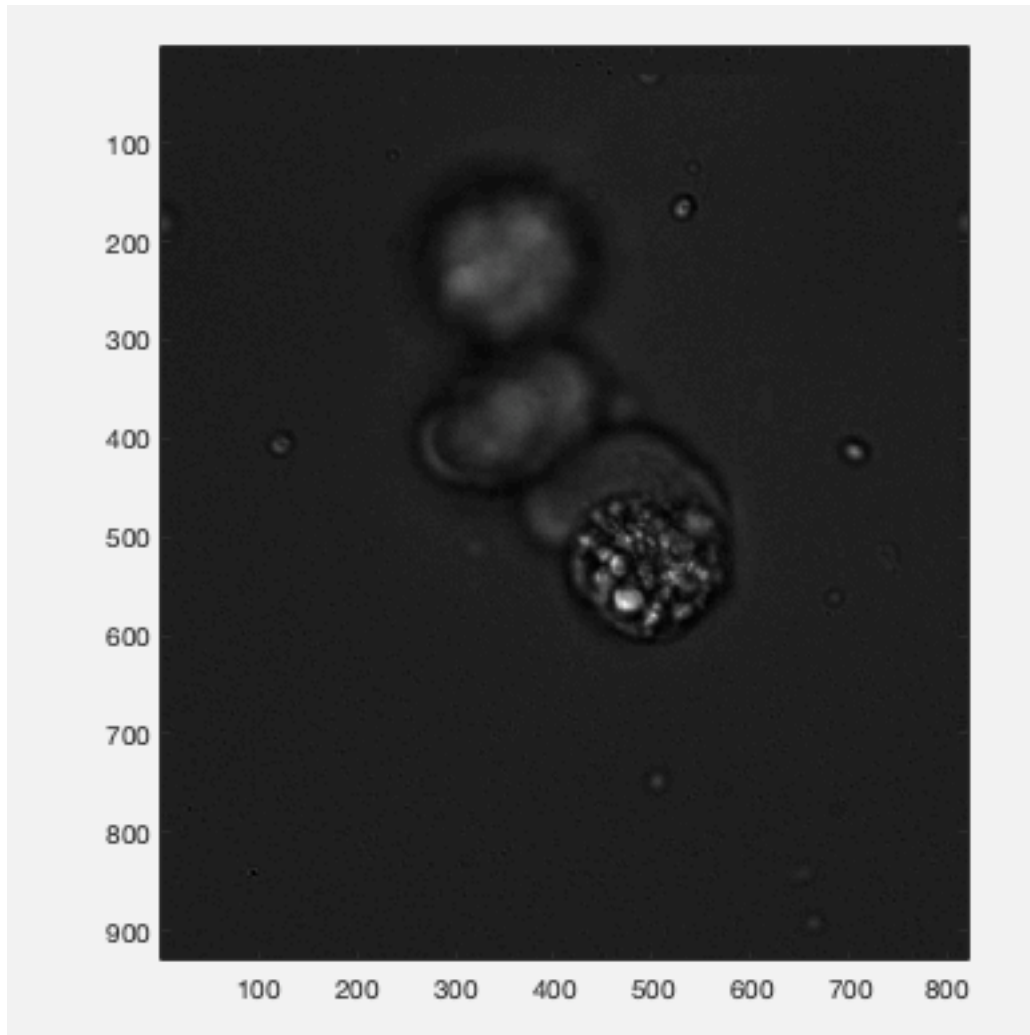


Displaying only cell image

```

» 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

```

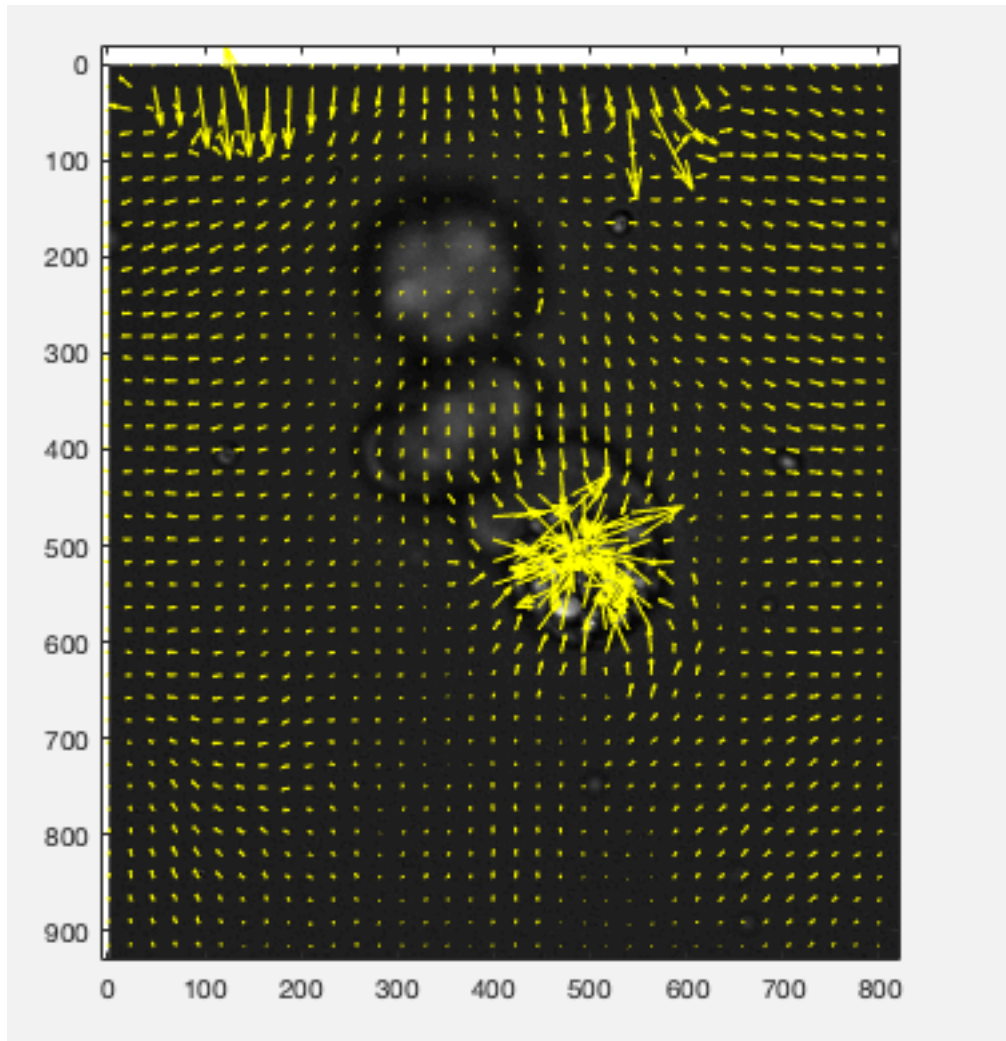


Display cell with Force vectors

```

» 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
```