Deep Learning Traction Force Microscopy

Software Installation

1. MATLAB and its Deep Learning Toolbox. MATLAB must be version 2019a or newer with the support for 3D convolution deep learning layer.
2. ImageJ or Fiji.
3. Iterative particle image velocimetry plugin for ImageJ, from https://sites.google.com/site/qingzongtseng/piv. Need to install system-dependent library file, OpenCV and javacv, as described in the sections of Downloads and Installation of the web page. See also the tutorial at https://sites.google.com/site/qingzongtseng/piv/tuto
4. Files for deep learning traction force microscopy from https://github.com/yuliwang8177/DL-TFM

Procedure

1. Plate cells on polyacrylamide substrates embedded with 0.2 μm diameter fluorescent beads. To prepare the substrate, see Polyacrylamide Sheets as Cell Culture Substrates at https://yuliwang.bme.cmu.edu/Methods/Materials/Artificial%20Materials/PAASubstrates.pdf. Density of the beads should be one bead per 5x5 to 10x10 pixels under a 40x objective lens.
2. Acquire a bead image underneath a cell using a 40x objective lens. The image should be focused near the surface of the substrate. The cell should be entirely within the image with a margin of at least 20 pixels all around. Also acquire a phase contrast image of the cell that shows cell border clearly.
3. Without changing the position of the sample, remove the cells (e.g., with trypsin solution), and acquire a bead image after removing the cell and its traction forces.
4. Place the two bead images in a stack in ImageJ (Image->Stacks->Images to Stack. The first slice of the stack should be the bead image after cell removal.
5. Align the two bead images as much as possible based on a motif of beads far away from the cell. See for example https://sites.google.com/site/qingzongtseng/template-matching-ij-plugin/tuto2
6. Crop the two bead images at identical locations to a square shape of NxN pixels where N must be a multiple of 104, 160 or 256. The cell should be entirely within the cropped area with a margin of 10-20 pixels all around. The multiplication factor should be between 5 and 10. If this is not possible due to the limited number of camera pixels, then crop the image to a multiple of 52, 80, or 128 pixels.
7. Crop the phase contrast image at exactly the same location without including it in the stack. Using the polygon selections tool of ImageJ, trace cell border in the phase contrast image as a single closed polygon. Save the trace as "XY Coordinates …", to generate a text file with X coordinates occupying the first column and Y coordinates occupying the second column.
8. Run ImageJ Plugins -> PIV -> iterative PIV (Basic). See files PIV_parameters104.pdf, PIV_parameters160.pdf or PIV_parameters256.pdf in the piv-fttc folder for setting the parameters. PIV3 interrogation window size should be 2x the multiplication factor in step #6. PIV2 interrogation window size should be 2x PIV3 interrogation window size, and PIV1 interrogation window size should be 2x PIV2 interrogation window size. Search window size should be 2x to x the corresponding PIV interrogation size.
9. Run the function pivToDspl in the piv-fttc folder, in the format of dspl = pivToDspl (pivFn, sz, spacing) where pivFn is the file generated by the above plugin, sz is N for an NxN image used in PIV analysis, spacing is the multiplication factor in step #6. An (N/spacing) (N/spacing) x 2

dense array dspl should be created in the Workspace, where the first plane carries X components of the strain field and second plane carries Y components.

10. If the strain field from #9 is at half size (52x52, 80x80, or 128x20 pixels), then generate full-size field using MATLAB interp2 function. See the program 80To160.m as an example for generating a 160x160 field from a 80x80 field.
11. Run the function txtToBrd to generate the array for cell border, in the format of [brdx, brdy] = txtToBrd (brdFn, sz, spacing), where brdFn is the file of cell border generated in step #7, sz is the size N of NxN image used for tracing, spacing is the multiplication factor in step #6.
12. Run the function filtDspl in the main folder, in the format of dsplFilt = filtDspl(dspl, brdx, brdy, 15, 3, 9, 3.5), where dspl is the tensor generated by step #9, brdx and brdy are border coordinates generated by step #10. The last four arguments control the noise filtration behavior and may be optimized by trial and error. The first argument (15) is the distance from cell border separating the near and far field. The second and third arguments (3, 9) specify the orders of median filter for filtering the near and far fields. The last argument is a filtration threshold for the gradient of strain applied throughout the field: if a strain vector shows a drastic difference from neighboring strain vectors, then it will be replaced with the average of neighboring vectors.
13. Run the function predictTrac in the main folder, in the format of trac = predictTrac(dspl, E), where dspl is the filtered displacement field from step #12, E is Young's modulus of the substrate. The output, trac, is a tensor of traction stress field of the same size as displacement tensor, where the first plane carries X components of the stress and second plane carries Y components.
14. Traction stress field may be visualized using the plotTrac function in the main folder, in the format of plotTrac(trac,brdx,brdy,scale,thinning), where trac is the traction tensor generated by step #13, brdx and brdy are border coordinates generated by step #11. The argument "scale" controls the lengths of vectors and thinning allows a lower density to be plotted than plotting at every pixel.