GUV fluctuation analysis functions

Functions by
Raghuveer Parthasarathy
The University of Oregon
raghu@uoregon.edu / http://physics.uoregon.edu/~raghu/
Some functions (Legendre polynomial determination, e.g.) begun by Kendra Nyberg

Document:
Raghuveer Parthasarathy
Created Nov. 2010
Significant revisions March 27, 2011
Last modified: Nov. 15, 2011 (very minor)

Data

Series of giant unilamellar vesicle (GUV) images. Load into a 3D array ("depth" = frame number), e.g. with TIFFseries.m

Necessary (or strongly recommended) functions

- allGUVedge.m – calls getGUVedge.m for all images in a series
- getGUVedge.m – determines the contour of a GUV in a single image; can call ridge-finding function
- calcrphi.m – called by getGUVedge.m
- autocorrguv.m – determines autocorrelation function of the contour
- legendre_kn.m – decomposes the autocorrelation into Legendre polynomials
- calc_kappa.m – calculates the bending rigidity (κ) from the autocorrelation / Legendre decomposition
- **`FrangiFilter2D.m`** – ridge finding filter (optional) from MATLAB File Exchange, by Marc Schrijver, D.Kroon. (http://www.mathworks.com/matlabcentral/fileexchange/24409-hessian-based-frangi-vesselness-filter) Calls functions **`Hessian2D.m, eig2image.m`**

Other useful functions

- viewseries.m – viewing images in a series
- TIFFseries.m – cropping and loading images into an array

Analysis procedure

View images in a series: **`viewseries.m`**
Load all images into an array: **`TIFFseries.m`**
(Crop images; other parameters = defaults)
Example: `outA = TIFFseries;`
Set imaging parameters
Examples, and comments:

```
% Parameters
scale = 0.11;  % um/px for image
res = 0.75;    % estimate "resolution" (um) – i.e. the segment size of
contours to fit
```

```
gausssig = 1;  % width of Gaussian smoothing filter (px).  Set to 0 for
no filtering
ridgesig = 1;  % width of Ridge-finding filter (px).  Set to zero for
no ridge finding.
% CMthresh : Intensity threshold for determining "center of mass" edge
position.  Ignore points dimmer than CMthresh*maxIntensity)  Default
0.35
```

Determine the contour for all the images: **`allGUVedge.m`**

Example ("outA" is the image matrix):

```
guv = allGUVedge(outA, CMthresh, scale, res, gausssig, ridgesig,
false);
```

Output: `guv`, a structured variable containing the contour information

Optional: Look at the contours

Example:

```
h = figure;
Nf = size(outA,3);
% display images, with found edges
for k=1:10:Nf
    figure(h); clf
    imagesc(outA(:,:,k)); colormap(gray);
    hold on
    plot(guv(k).ctr(1) + cos(guv(k).phi).*guv(k).R, guv(k).ctr(2) -
sin(guv(k).phi).*guv(k).R, 'yx');
    title(sprintf('Frame %d', k))
    pause(0.1)
end
```

Calculate the autocorrelation function for the edge, in each frame:

```
xi = autocorrguv(guv);
```

Determine the Legendre coefficients for each autocorrelation.

Example:

```
n_terms = 14;   % number of Legendre coefficients to calculate
B_n = legendre_kn(xi, n_terms, false);
figure; plot(1:n_terms, mean(B_n,1), 'ko')
```

Calculate the bending rigidity, etc.

First, more parameters:

```
meanR_um = mean([guv(:).R])*scale;  % mean vesicle radius, microns
fps = 23;  % frames per second
```

Example:

```
[kc, sigkc, sigma, sigsigma] =  calc_kappa(B_n, [], [], meanR_um, [],
fps, true);
```

Revise based on the quality of the fits, etc., especially limiting the range of `n` *that are fit. E.g.:*

```
[kc, sigkc, sigma, sigsigma] =  calc_kappa(B_n, [3 11], [], meanR_um,
[], fps, true);
```

NOTE: `calc_kappa.m` plots and displays a lot of "rough" information having to do with various ways to extract the rigidity, etc., from the Legendre coefficients of the edge autocorrelations. It's not yet clear what all of this means. I recommend "using" the values for rigidity and tension that come from the chi^2 fit, but paying attention to the other plots. Note also that "low n" data can suffer due to poor statistics, since these modes are slow, and "high n" data can suffer due to limited spatial and temporal resolution, since these modes are fast and small.

Functions

getGUVedge.m

```
% Function to determine the contour of a giant vesicle from a fluorescence
% image.  Determines the local maximum of intensity (I) vs. radial
% coordinate (rho) at a given angle (phi), using I(rho) at the previous phi
% to define the bounds of the search.
% Examines phi from -pi to pi Counterclockwise and then clockwise; takes
% the average of the two weighted by the edge intensity.
% If the input image is the "first" in a series, the user is asked to
% roughly determine the vesicle center and the initial search bounds.

function [R, phi, Amp, newc, dR, closedflag] = ...
    getGUVedge(A, ctr, roughR, dR, CMthresh, scale, res, showplots)
```

Example:

```
[first modify getGUVedge.m to set diagnosticplots = true;
  cd 'C:\Documents and Settings\Raghu\My Documents\Experiments and
  Projects\COPII\GUV fluctuations'
  A = imread('23Dec10 7a tr 60x 10msexp 23fpst028.tif');
  scale = 0.11;  % um/px for image
  res = 0.75;    % estimate “resolution” (um)
  gausssig = 1;  % width of Gaussian smoothing filter (px)
  ridgesig = 1;  % width of Ridge-finding filter (px)
  CMthresh = 0.35;  % center of mass intensity threshold
  n_terms = 14;   % number of Legendre coefficients to calculate
  fps = 21; % frames per second
  showplots = true;
  [R, phi, Amp, newc, dR, closedflag] = getGUVedge(A, [], [], [], CMthresh,
  scale, res, showplots);
```

Calls **`calcrphi.m`**

% allGUVedge.m

```
% Takes a series of images, loaded by TIFFseries, and tracks
% the GUV edges using getGUVedge.m
% For each image, first apply a Gaussian smoothing filter, and then perform
% ridge-finding using FrangiFilter2D.m.  Then, sends this to getGUVedge.m
% For the first image, getGUVedge will ask for user-input approximate
% vesicle center and initial edge location.
function [guv] = allGUVedge(outA, CMthresh, scale, res, gausssig, ridgesig,
dispopt)
```

% calcrphi.m

```
function [r, phi] = calcrphi(A, x, y)
```

autocorrguv.m

Determine the vesicle edge autocorrelation function

Example: `xi = autocorrguv(guv);`

Old functions

`viewHoughseries.m`

Run this to determine circle-fitting parameters (radius, threshold, obj. size, Hough thresh)

`getallcenters.m`

Find the vesicle center in each image

Example: `H = getallcenters(outA, 8, 0.9, 0.999, [31 33], 'gauss');`

`trackedge_kn.m`

For each image, call `slopeEdge_kn.m`

Determines vesicle center using `houghring_rp.m` (written earlier)

Calculates radial distance to vesicle center, and angle, using `calcrphi.m`

For each φ (binned) finds R ≡ r at which intensity vs. r is maximal.

Outputs this R(φ) in a structured array ("guv"). Elements ↔ images

`guv(k).R` = R values (px), `guv(k).phi` = φ values (radians) for image `k`.

Example: `[guv outH] = trackedge_kn(outA, H, 40, 3, 2);`

[**unfinished – ignore what's below!**]

`legendre_kn.m`

Decomposes R(θ) into Legendre polynomials, returns coefficients

Note that the indexing of the coefficients starts with n=1 for the "cosθ" term

Example:
```
m_terms = 5;
    A_m = legendre_kn(guv, m_terms);
```

`calc_kappa.m`

Determines bending modulus tension from the Legendre polynomial coefficients

Note that the indexing of the coefficients starts with n=1 for the "cosθ" term

Example: `[kc, sigkc, sigma, sigsigma] = calc_kappa(A_m,[], [], [], true);`

Be sure to save a **.mat** file of the output, especially the R(θ) array!

Issues:

How to deal with bad frames? For now: `calc_kappa.m` shows the Legendre coefficients for all frames. One can see if particular frames are bad, and re-run with a subset of the A_m matrix, e.g.:

```
[kc, sigkc, sigma, sigsigma] = calc_kappa(A_m(20:end,:),[], [], [], true);
```