Brain microvascular endothelial cells resist elongation due to curvature and shear stress

Mao Ye, Henry M. Sanchez, Margot Hultz, Zhen Yang, Max Bogorad, Andrew D. Wong, and Peter C. Searson

Supplementary Information

Relationship between energy and cell morphology for HBMECs and HUVECs.

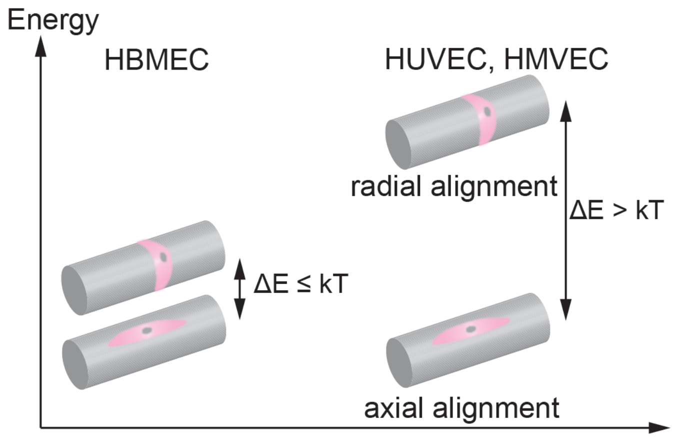


Figure S1. Schematic illustration showing the different energy states for axial and radial alignment of different endothelial cell types. We consider the energy associated with axial and radial alignment of an endothelial cell on a cylindrical surface. For HBMECs the energy difference (ΔE) between the two states (axial and radial) is less than the thermal energy (kT, where k is the Boltzmann constant and T is temperature) and hence there is no driving force for preferential alignment. In contrast, the energy for radial orientation is larger than for axial alignment, resulting in an energy barrier for radial alignment. As a result of this energy barrier, cells tend to align along the axial direction of the rod resulting in elongation and decreased circularity, as well as a small average orientation angle. The results shown in Figure 2 suggest that the energy barrier is dependent on curvature or rod diameter, with the energy barrier increasing with decreasing diameter.

Rod assay

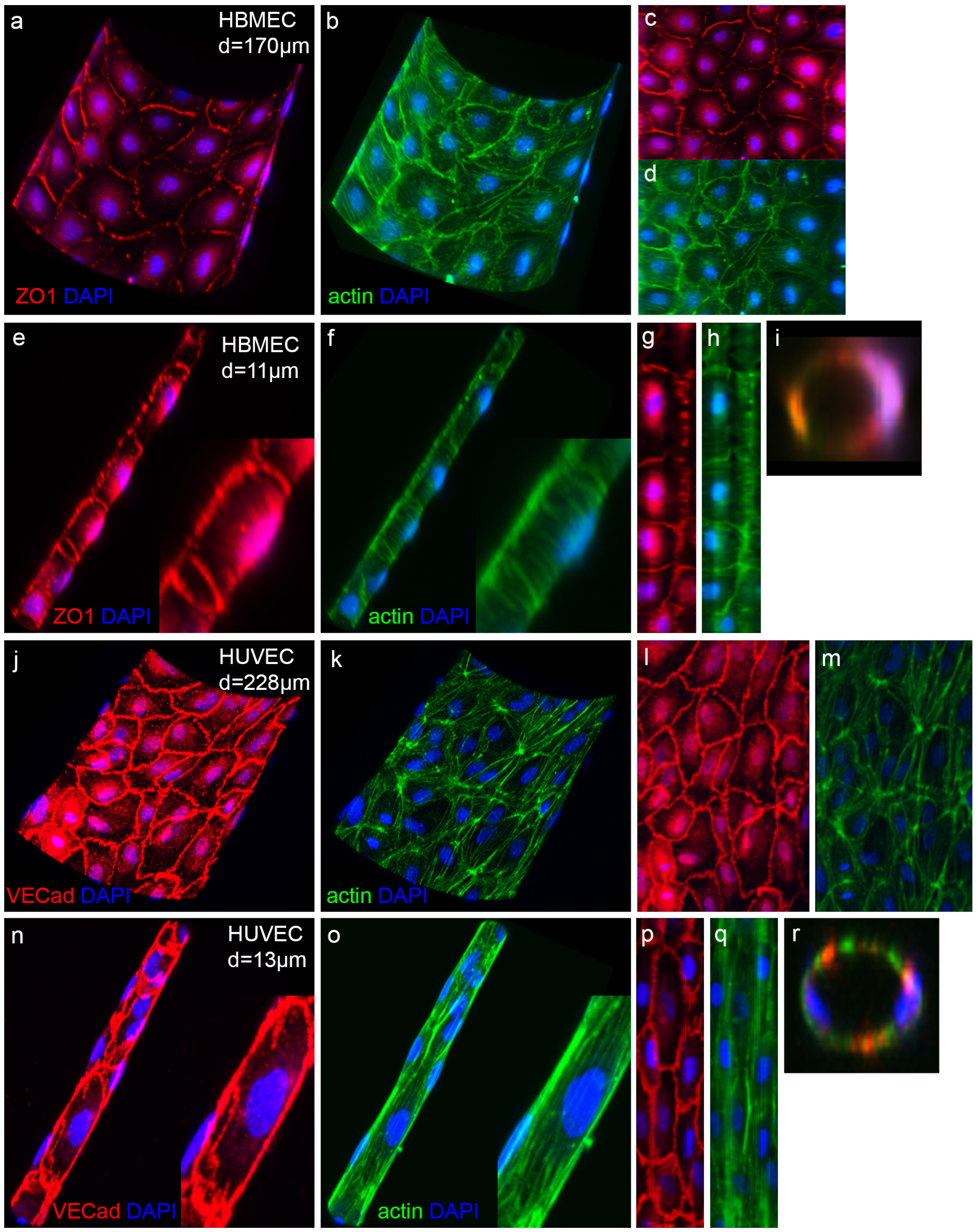


Figure S2. Confocal microscope images of HBMECs and HUVECs on large and small diameter glass rods. **HBMECs on a 170 μm diameter rod**: (a) ZO-1 (red), DAPI (blue); (b) actin (green), DAPI (blue); (c, d) corresponding unwrapped images. **HBMECs on an 11 μm diameter rod**: (e) ZO-1 (red), DAPI (blue); (f) actin (green), DAPI (blue); (g, h) corresponding unwrapped images; (i) cross-section ZO-1 (red), actin (green), DAPI (blue). **HUVECs on a 228 μm diameter rod**: (j) VE-cadherin (red), DAPI (blue); (k) actin (green), DAPI (blue); (l, m) corresponding unwrapped images. **HUVEC on a 13 μm diameter rod**: (n) VE-cadherin (red), DAPI (blue); (o) actin (green), DAPI (blue); (p, q) corresponding unwrapped images; (r) cross-section VE-cadherin (red), actin (green), DAPI (blue).

Endothelial cell morphology in 2D

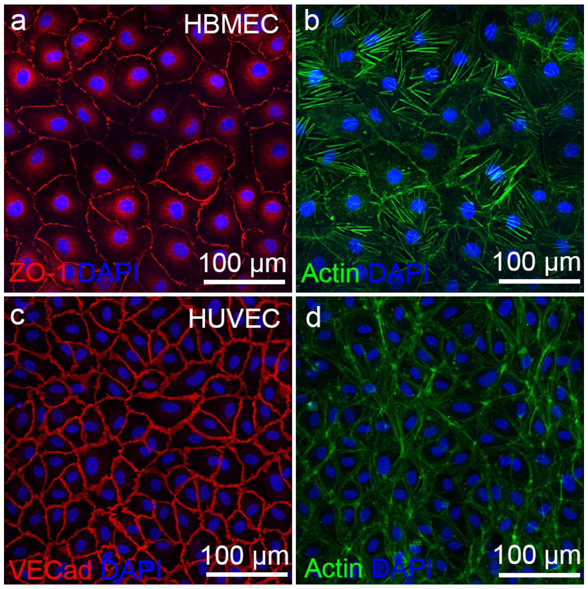


Figure S3. Fluorescence images of confluent monolayers of HBMECs and HUVECs in 2D. **HBMECs:** (a) ZO-1 (red), DAPI (blue); (b) actin (green), DAPI (blue). **HUVECs**: (c) VE-cadherin (red), DAPI (blue); (d) actin (green), DAPI (blue).

Morphological Data

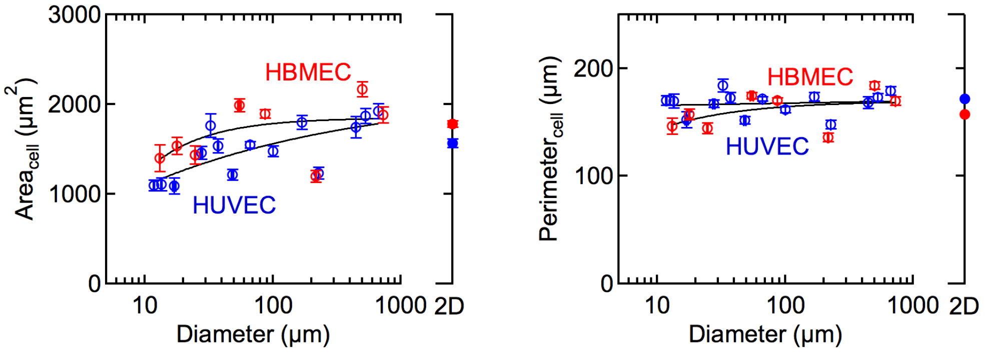


Figure S4. Cell area and perimeter for HBMECs and HUVECs on rods with different diameter and in 2D.

Finite size effects in orientation angle

The orientation angle of a cell is defined as the angle between the cell long axis (ℓ) and the rod axis (Figure S5). On large rods where the perimeter is much larger than the long axis of the cell ($\pi d >> \ell$), the cell can adopt any orientation angle between 0° and 90°. For a uniform distribution of orientation angles, the average value is 45°. If there is an energy barrier (see Figure S1) to wrapping around the perimeter of the vessel then the cells will be preferentially aligned along the vessel axis and the average orientation angle will be less than 45°. However, when the rod perimeter is less than the long axis of the cell ($\pi d \leq \ell$), then large angles are prohibited and the maximum allowed orientation angle is less than 90°, and hence the average angle is also less than 45°.

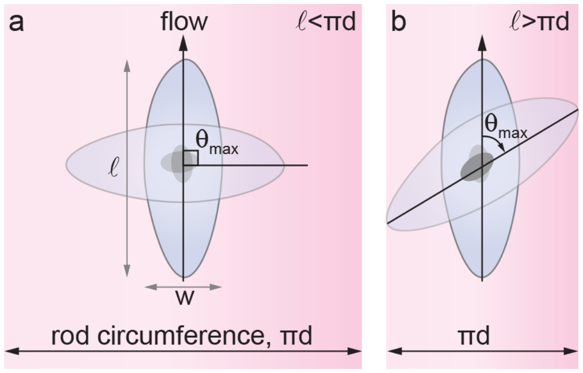


Figure S5. Schematic illustration showing how the finite size of a rod can limit the distribution of orientation angles of an endothelial cell.

The influence of curvature and finite size effects on the average orientation angle can be seen in scatter plots of cell length and orientation angle for individual cells on a given rod diameter (Figure S6). HBMEC cells span the full range of allowed angles on large diameter (499 μm) glass rods. For the range of cell lengths, approximately 40 - 80 μm, all orientation angles are allowed and the average angle is about 45°. On small diameter (18 μm) rods, the HBMECs span the full range of allowed angles, however, for the longer cells higher orientation angles are prohibited. HUVEC cells on large diameter (530 μm) rods exhibit the full range of orientation angles, although the frequency at smaller angles is significantly higher due to curvature driven alignment. On small diameter rods (13 μm) the HUVECs exhibit orientation angles considerably lower than the allowed range, illustrating the effect of curvature on cell alignment. In summary, the decrease in average orientation angle for HBMECs at small diameters is due to a finite size effect and not due to the influence of curvature. In contrast, the decrease in angle for HUVECs is due to curvature.

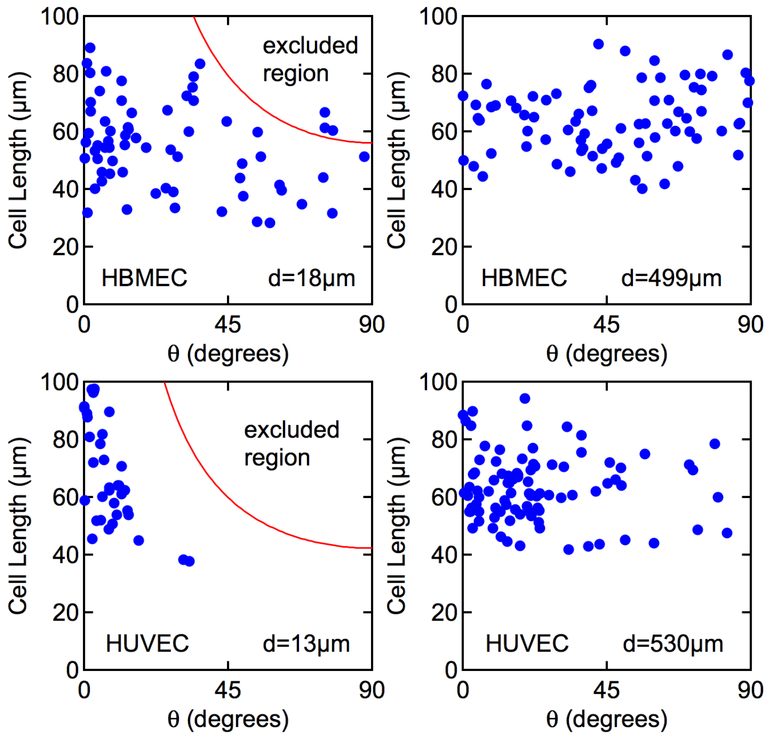


Figure S6. Cell length and orientation angle for HBMECs and HUVECs on small and large diameter rods. (a) HBMECs on 18 ± 0 μm (SE) glass rods (N = 66), (b) HBMECs on 499 ± 0 μm (SE) rods (N = 76), (c) HUVECs on 13 ± 0 μm (SE) rods (N = 39), and (d) HUVECs on 530 ±1 μm (SE) rods (N = 92). The solid lines represent $\ell\sin\theta = \pi d$ where d is the average rod diameter. A cell of length ℓ can adopt any orientation angle θ on a rod of diameter d as long as $\ell\sin\theta \leq \pi d$. When the cell length is larger than πd then all orientation angles are allowed, however, when $\ell \leq \pi d$ then some angles are prohibited. This finite size effect leads to a change in the distribution of orientation angles and a decrease in the average orientation angle. SE is the standard error.

Human dermal microvascular endothelial cells (HMVECs).

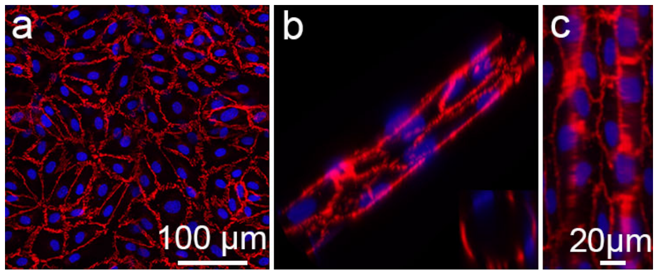


Figure S7. Fluorescence images of confluent monolayers of human dermal microvascular endothelial cells (HMVECs) in 2D and on a 24 μm diameter rod. (a) HMVECs in 2D. (b) HMVECs on a 24 μm diameter rod. (c) Corresponding unwrapped image for (b). VE-cadherin (red), DAPI (blue).

Comparison of morphology of HBMECs, HUVECs, and HMVECs.

Assuming that HBMECs are representative of endothelial cells in brain micovessels, HUVECs are representative of non-brain large vessels, and HMVECs are representative of non-brain microvessels, then data from the rod assay using these cells lines allows us to compare the morphological response to curvature of brain and non-brain microvessels. As shown in Figure S5, the inverse aspect ratio, circularity, and average orientation angle are very similar to HUVECs under all conditions. On small (~ 20 μm) diameter rods, these three parameters are significantly lower for HMVECs and HUVECs compared to HBMECs, suggesting that only brain microvascular endothelial cells are programmed to resist elongation due to curvature.

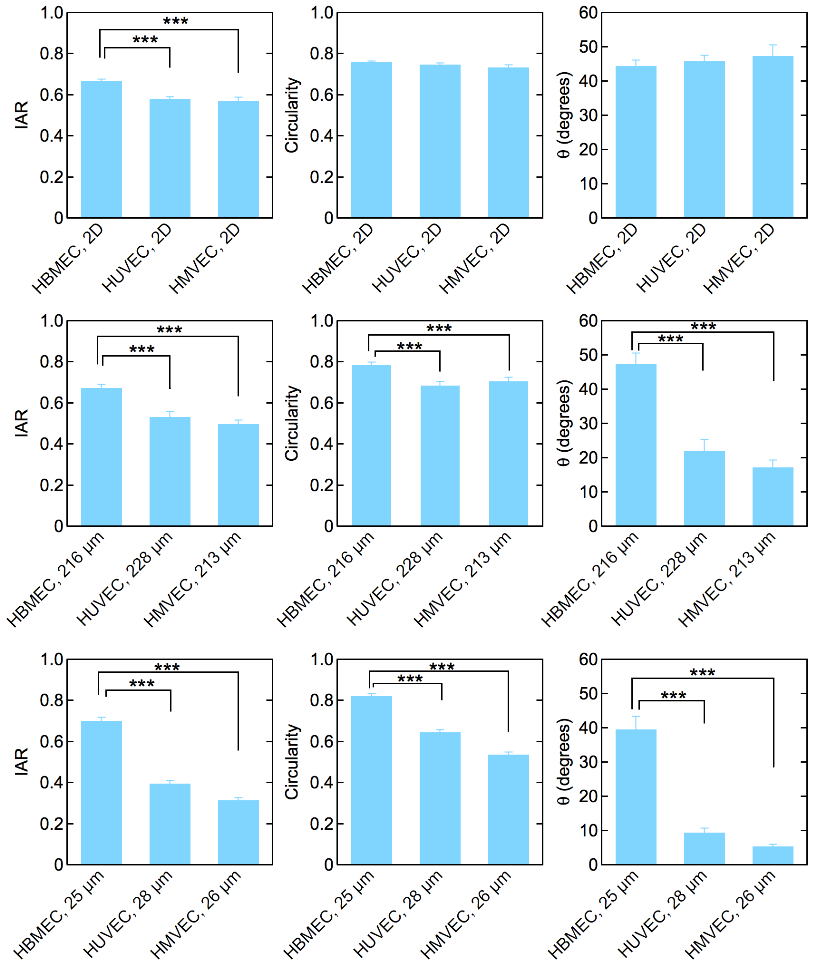


Figure S8. Cell morphological parameters for confluent monolayers of HBMEC, HUVEC and HMVECs in 2D and on large (~ 200 μm) and small (~ 20 μm) diameter glass rods. Parameters include: inverse aspect ratio (IAR), circularity, and average orientation angle. **HBMECs**: 2D (N = 238), d = 216 ± 3 μm (SE) (N = 32), d = 25 ± 0 μm (N = 48). **HUVECs**: 2D (N = 242), d = 228 ± 0 μm (N = 46), d = 28 ± 0 μm (N = 75). **HMVECs**: 2D (N = 64), d = 213 ± 5 μm (N = 60), d = 26 ± 0 μm (N = 110). *** P < 0.001. Error bars represent standard error (SE) .

Alignment of actin filaments

To quantitatively analyze actin filament alignment, we determined the radial intensity distribution from fluorescence images using fast-Fourier transforms. We compared HBMEC and HUVEC cells in 2D confluent monolayers and on rods under static and flow conditions. Fluorescence images of cells in 2D and on larger rods ($d \approx 200$ μm) were cropped to be 141 x 141 μm with a resolution of 0.44 μm per pixel. Images of cells on smaller rods ($d \approx 10$ μm) were cropped to be 41 x 41 μm with a resolution of 0.44 μm per pixel. The input images were cropped to be square to ensure equal contributions from vertical and horizontal axes.

We obtained the largest square image for each experimental condition, resulting in 5 cropped images for HUVEC and HBMEC cells in 2D, and 4 cropped images for HUVEC and HBMEC cells on large diameter rods ($d \approx 200$ μm), and 5 cropped images for HUVEC and HBMEC cells on small diameter rods ($d \approx 10$ μm).

Control experiments. To study how image area influences the radial distribution in the frequency domain, we cropped the image of HUVEC cells in 2D into 4 small squares, and compared the radial distribution in the frequency domain, and found that the relative change is less than 20%, smaller than the relative difference between cells in different conditions (e.g. HUVEC cells on large diameter rods under static conditions compared to small diameter rods under static condition). To study how the resolution influences the radial distribution in the frequency domain, we compared the images of HUVEC cells in 2D at resolutions of 0.36 and 0.44 μm per pixel. The relative change in radial distribution in the frequency domain was less than 15%.

The square images, $f(x, y)$ ($0 \leq x, y \leq N - 1$, N is the number of pixels for the image square) were transformed into the frequency domain using *fft2* in MATLAB. The FFT image was produced by shifting $F(0, 0)$ to the middle using *fftshift*, and calculating the magnitude of $F(u, v)$ - $|F(u, v)|$ using the *abs* routine in MATLAB.

The actin stress fibers in the fluorescence images can be considered as a superimposition of 2D intensity waves, whereas each pixel in the frequency domain ($|F(u, v)|$) can be considered as a single wave of intensities in frequency domain. The intensity for each pixel in the frequency domain, $|F(u, v)|$, represents the strength of each single wave. To characterize the directionality of the actin stress fibers in the fluorescence image, we divided the FFT image into 18 bins, each with an angular range of 10°, and the intensities in each bin were added together, and divided by the total intensities of all bins (the center pixel or pixels excluded). The intensity fraction of all bins was plotted as a bar graph. If the actin stress fibers were uniformly distributed in the fluorescence image, each bin is expected to have an intensity fraction of about 5.6% ($\approx 100/18$) (see dotted line in bar graphs in Figure S7).

The actin fiber alignment is determined from the parallel and perpendicular indices. The parallel index represents the degree of alignment of actin fibers along the rod axis (vertical) and is defined by the sum of the intensities at $0 \pm 10°$ (i.e. the sum of the intensities in the 0 - 10° and 170 - 180° bins). The perpendicular index represents the degree of alignment perpendicular to the rod axis (horizontal) and is defined by the sum of the intensities at $90 \pm 10°$ (i.e. the sum of the intensities in the 80 - 90° and 90 - 100° bins).

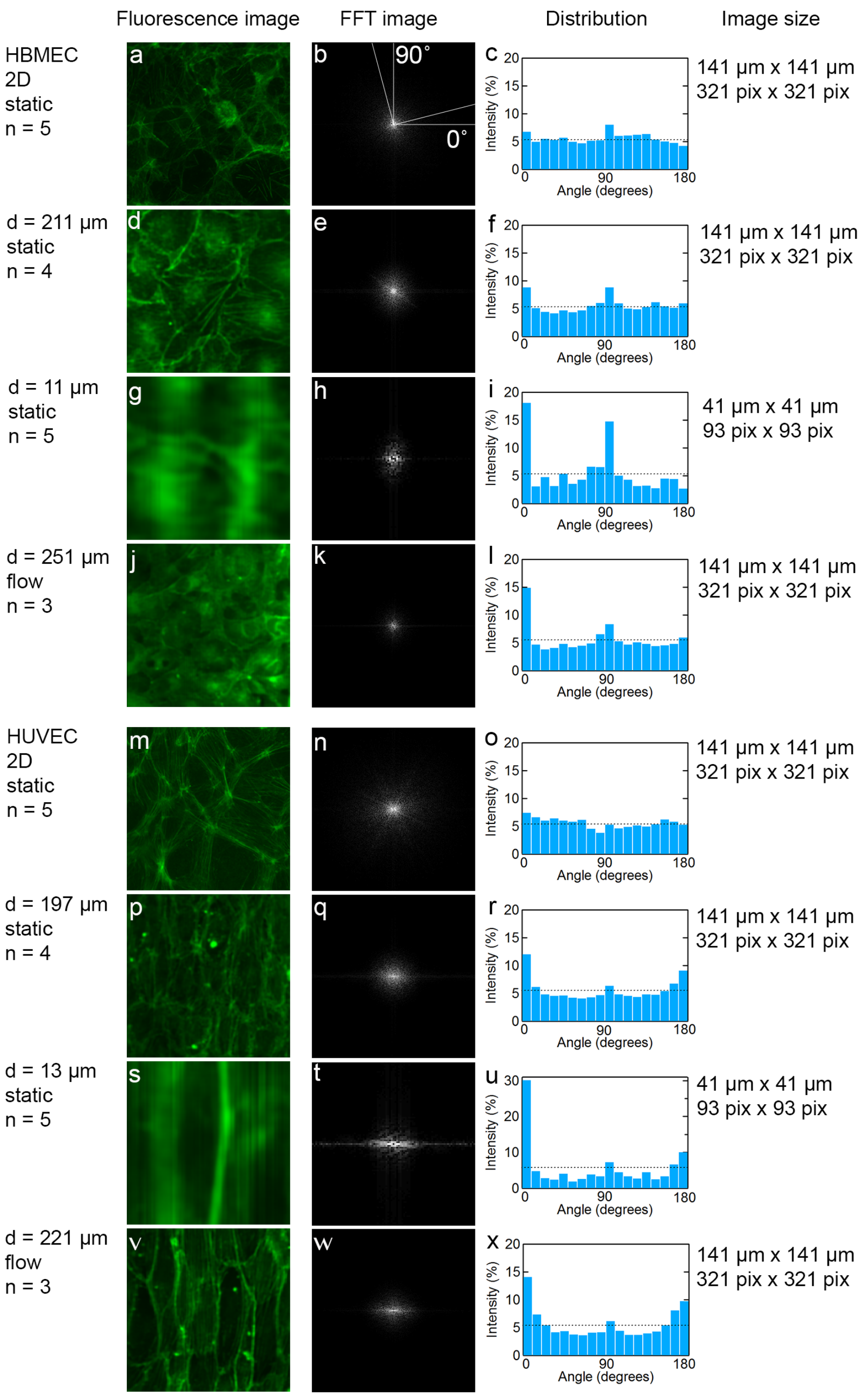


Figure S9. Actin fiber alignment. Fluorescence images of confluent monolayers of HBMEC and HUVEC cells in 2D, and on large (d ≈ 200 μm) and small (d ≈ 10 μm) diameter rods under static conditions. Also shown are fluorescence images of HBMECs and HUVECs on large diameter rods under shear stress (d ≈ 200 μm). All images were saved at a resolution of 0.44 μm per pixel. The FFT images show the distribution of intensity in reciprocal space, with the zero-frequency pixel in the center. The bar graphs show the radial intensity distributions.

Quantitative analysis of actin filament alignment

For HBMEC and HUVEC cells in 2D the radial distributions show no preferential alignment – the parallel and perpendicular indices are about 11%, characteristic of a uniform distribution (100%/9). For HBMECs on rods, both the parallel and perpendicular indices increase with decreasing diameter, indicating both parallel and perpendicular alignment of the actin filaments. In contrast, for HUVECs on rods, the parallel index increases significantly with decreasing diameter, showing strong axial alignment. For HBMEC cells on large diameter rods ($d \approx 200$ μm), shear stress results a small decrease in the perpendicular index and a larger increase in the parallel index. Similar results are observed for HUVECs.

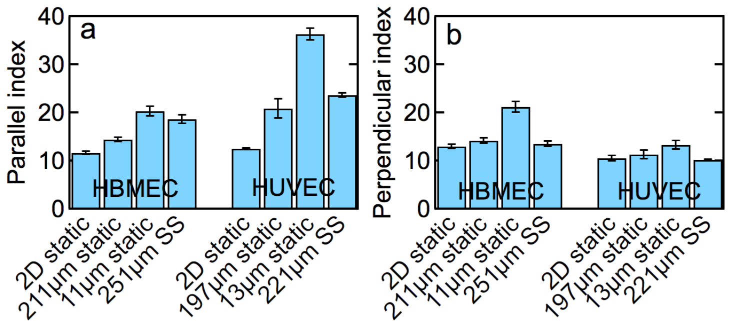


Figure S10. Parallel and perpendicular indices for actin fiber distribution. Data obtained from analysis of the radial intensity distributions for confluent monolayers of HBMEC and HUVEC cells in 2D, on large and small diameter rods under static conditions, and on large diameter rods under shear stress (SS). The average index for no preferential orientation is 11.1 (100%/9). **HBMECs: (**2D static) image number = 5; (211 μm static) d = 211 ± 14 μm (SE), image number = 4; (11 μm static) d = 11 ± 0 μm (SE), image number = 5; (251 μm SS) 251 ± 2 μm (SE), image number = 3. **HUVECs: (**2D static) image number = 5; (197 μm static) d = 197 ± 14 μm (SE), image number = 4; (13 μm static) d = 13 ± 0 μm (SE), image number = 5; (221 μm SS) d = 221 ± 12 μm (SE), image number = 3.

Cross-section velocity profile in a microfluidic channel.

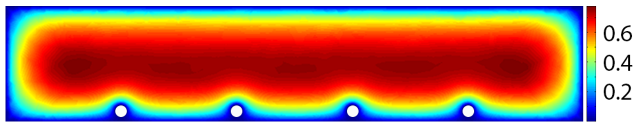


Figure S11. Cross-section velocity profile (m/s) around 200 μm diameter glass rods located 100 μm above the bottom of a microfluidic channel.

UNWRAP User Document

This MATLAB application takes a set of confocal microscope z-stack images of fluorescently labeled cells in a 3D cylindrical geometry and creates an unwrapped 2D image from the "tube" surface (Figure 1). The application works best for a confluent monolayer of cells on a cylindrical surface. The application can accommodate up to three channels (e.g. R, G, B), for example a junctional protein, a cytoskeleton protein, and a nuclear stain. This code has been tested for MATLAB R2013b in Windows 7, and should work on other similar platforms.

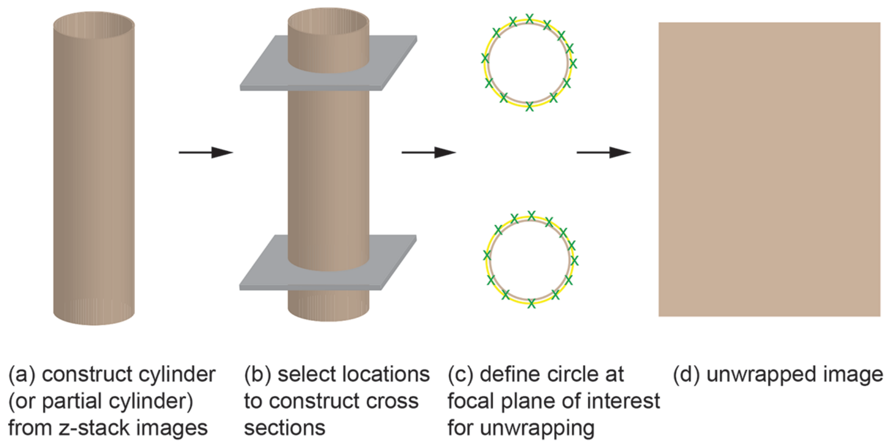
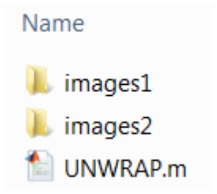

(a) construct cylinder (or partial cylinder) from z-stack images
(b) select locations to construct cross sections
(c) define circle at focal plane of interest for unwrapping
(d) unwrapped image

Figure 1. Workflow of the UNWRAP application.

To make best use of this tutorial, download the folder "UNWRAP" from the Searson Group website (http://www.jhu.edu/searson/). In the working folder "UNWRAP", there are two sample image folders ("images1" and "images2") and the application MATLAB file (UNWRAP.m), see Figure 2. The MATLAB script (UNWRAP.m) can also be generated by copying the code from this document (see MATLAB Code). The sample images include a confocal z-stack image of cells on a complete cylinder ("images1") and on a partial cylinder ("images2").



Name
images1
images2
UNWRAP.m

Figure 2. Contents in the working folder "UNWRAP".

Instructions

Step 1. For the purpose of the tutorial, the sample image folder "images1" is used as the input for "UNWRAP". To analyze your own images, create your own image folder (e.g. "imagesX") inside the working folder, then create a sub-folder (e.g. "z_stack") inside your image folder, transfer the z-stack images for your cylindrical structure from the software associated with the confocal microscope into your sub-folder (e.g. “z_stack”). The z-stack images should be stored properly according to the following specifications (see Step 4): (1) the z-stack images should be named sequentially as “common filename” + “numbers” + “.format” (e.g. “all_” + “001” + “.tif”, the numbers should have the same number of digits), (2) the cylindrical structure should be vertically oriented on the screen (important for the program to handle images properly). An example of a z-stack image near the middle of the cylinder from the folder “images1” is shown in Figure 3.

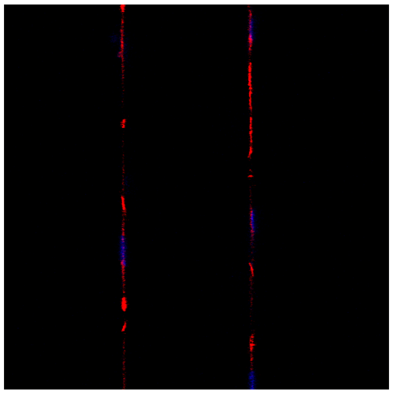


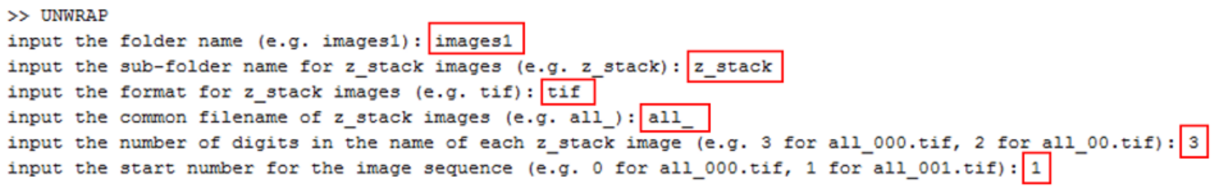
Figure 3. An example of a z-stack image near the middle of the cylinder oriented vertically on the screen.

Step 2. In windows, use Notepad to create a text file (.txt) inside your image folder (e.g. “imagesX”) with name “scale.txt” and enter two numbers (separated by a carriage return): the first is the xy resolution (in units of µm/pixel) of each z-stack image, and second one is the spacing (in units of µm) between z-stack images (Figure 4). Make sure the resolution information is saved correctly, otherwise you may get a distorted 3-D image in the following steps. The hierarchy of your folder and files should be similar to the sample image folder “images1” or “images2” (i.e. a “scale.txt” file and a sub-folder of the image sequence inside your image folder).

```
0.41
0.2
```

Figure 4. The resolution information stored in file “scale.txt”. In this example, 0.41 represents the xy resolution (0.41 µm/pixel), and 0.2 represents the spacing between z-stack images (0.2 µm).

Step 3. Open UNWRAP.m in MATLB (can simply double click the file, or open it through the MATLAB “file” tab in the top left corner), then left click run . After running UNWRAP.m, a series of prompts will appear in the command window (Figure 5). Enter the relevant information after each prompt and press enter. Caution: you should type names that match exactly with your folder and file names; otherwise, program will fail. Figure 5 shows an example of the screen for the sample image folder “images1”. In case of a problem, press “Ctrl” + “C” in the Command Window and restart the program by clicking .



```
>> UNWRAP
input the folder name (e.g. images1): images1
input the sub-folder name for z_stack images (e.g. z_stack): z_stack
input the format for z_stack images (e.g. tif): tif
input the common filename of z_stack images (e.g. all_): all_
input the number of digits in the name of each z_stack image (e.g. 3 for all_000.tif, 2 for all_00.tif): 3
input the start number for the image sequence (e.g. 0 for all_000.tif, 1 for all_001.tif): 1
```

Figure 5. Prompts appeared in the command window for sample image folder "images1". Names that need to be typed in for sample image folder "images1" are in red boxes.

Step 4. A sequence of numbers will appear in the Command Window while the program is reading all the input z-stack images and scale information from your image folder ("images1"). In about 30 seconds, a cross-section image will be displayed for the user to locate the user-defined focal plane. This is the first of two cross-section images (one at each end of the cylinder) that are used to define the focal plane of the cylinder for unwrapping. The second cross-section image will appear in Step 5. If your resolution information is correct and the z-stack images are stored properly (see Steps 1 and 2), you will get a circular cross-section image (Figure 6).

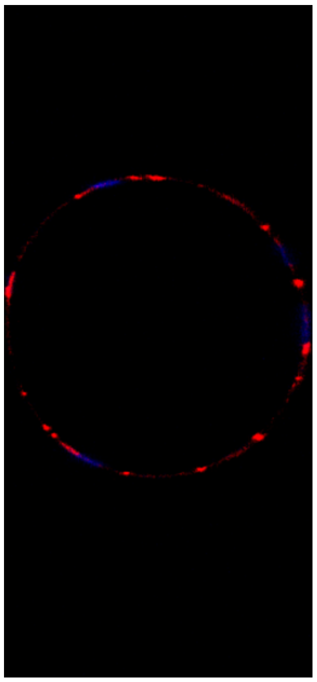


Figure 6. A cross section image of the cylindrical structure is shown on the screen.

Step 5. Create a rectangular region of interest (ROI) around the cross-section by moving your cursor to the top-left side of your circular cross-section image and holding your mouse and dragging to the bottom-right (see Figure 7 for an example). The rectangular region of interest (ROI) MUST include the image circle. If you are not satisfied with your initial ROI, adjust the rectangle by clicking the edge of it and dragging such that it contains the whole cross section of interest with an additional margin. Double left click inside the rectangle and a zoomed-in image will be generated (Figure 7 and 8).

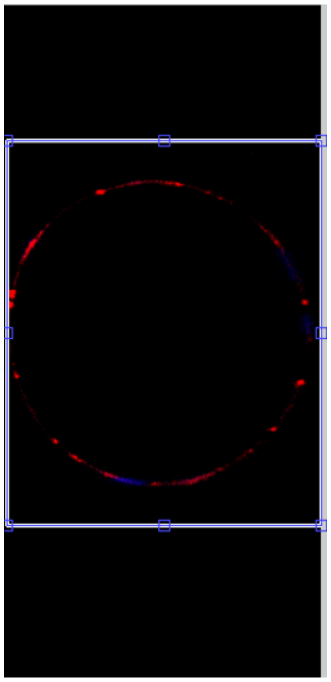


Figure 7. A region of interest (ROI) of the cross section is created for the cylindrical structure.

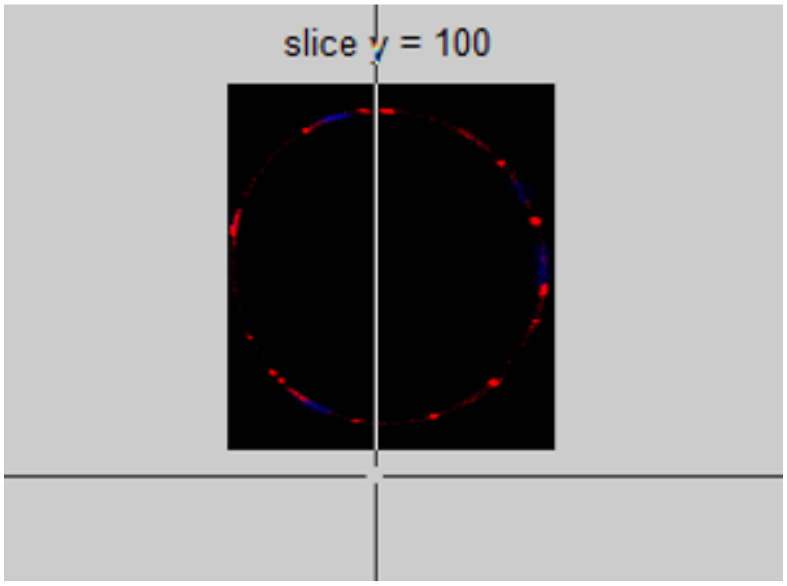


Figure 8. A zoomed-in figure of the cross section image is generated according to a user selected region of interest.

Step 6. Maximize the zoomed-in image (Figure 9) and left click 10 – 20 points uniformly distributed on the circular cross-section (Figure 10a and 10b). After each click, you will see a green cross "x" in the clicked position in the image. If you want to change points, simply start over from Step 3. After finishing selecting points, right click on the image. A yellow fitted circle will be generated based on the input points you selected (Figure 11). This circle will be used as the first reference for unwrapping the cylindrical structure.

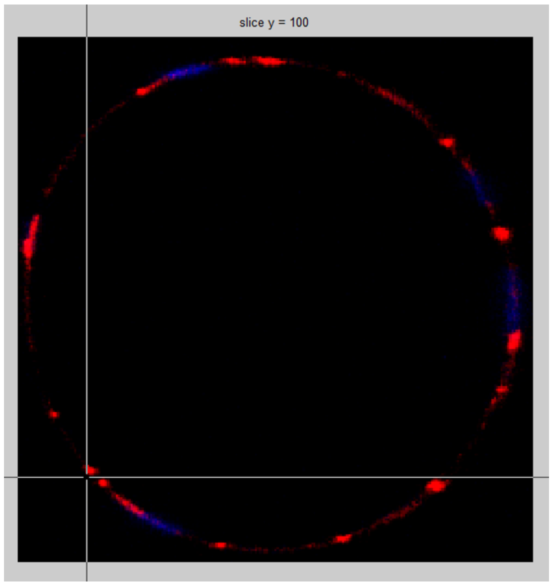


Figure 9. The zoomed-in figure is maximized for convenient point-clicking.

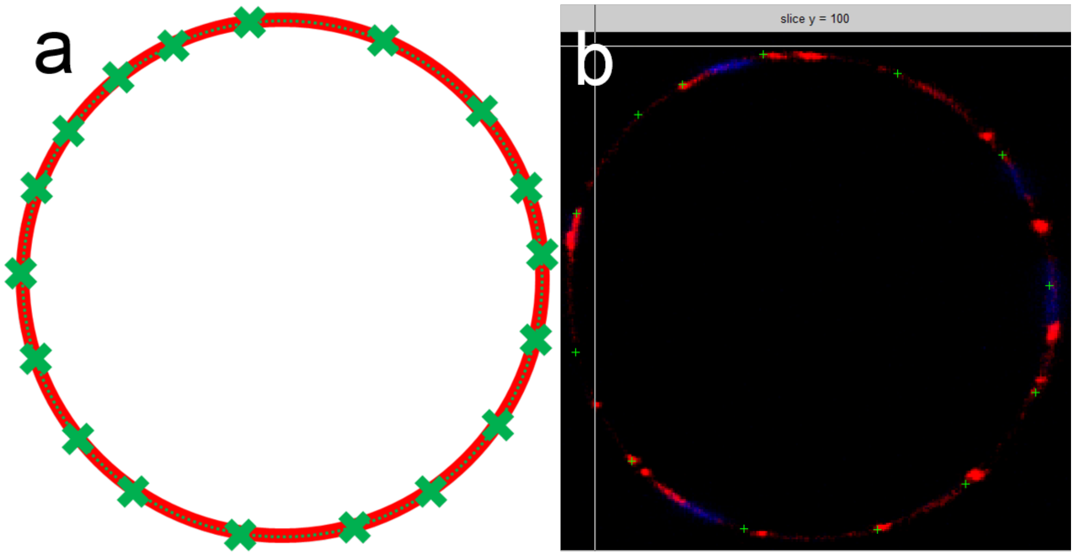


Figure 10. (a) Green "x" markers represent the positions the user should click to identify the cross section. (b) Green "+" markers are displayed on the screen as the user is clicking positions.

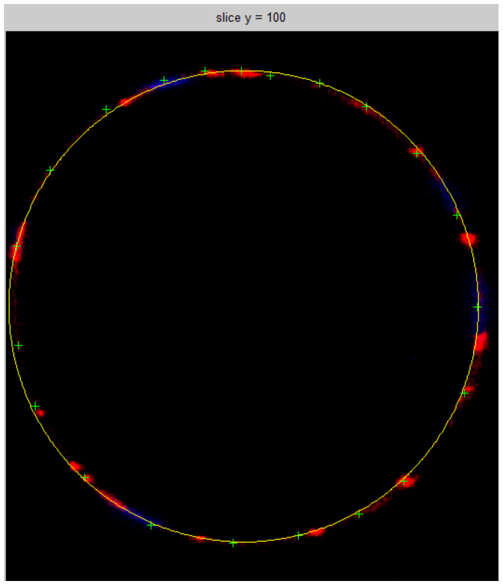


Figure 11. A yellow fitted circle is shown as an overlay.

Step 7. Repeat steps 5 and 6 for the second cross section image.

Step 8. An unwrapped image of all channels (i.e. R, G, B) will appear on the screen as your result (Figure 12). In the meantime, a sub-folder called "unwrapped_images" (Figure 13) and an "info.txt" file (Figure 14) will be generated inside your image folder (e.g. "images1"). Inside the "unwrapped_images" folder, the unwrapped images are separated into different combinations of channels (RGB, RB, GB, B with 1 image alone, and 2 images side-by-side). The "info.txt" file (Figure 13) will provide information regarding your cylinder including resolution (μm/pixel), cylinder diameter (μm), and cylinder length (μm).

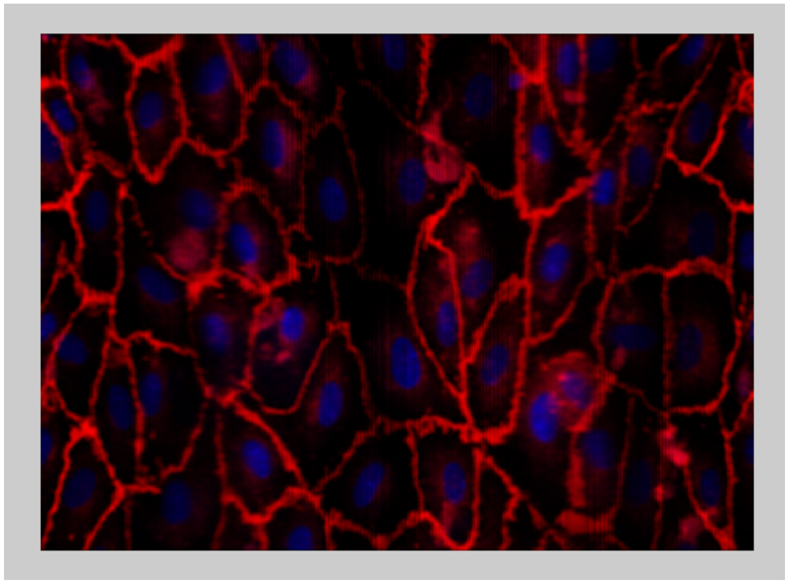


Figure 12. An unwrapped image of the cylindrical structure is shown on the screen after Step 7. This image is also saved in the output folder "unwrapped_images".

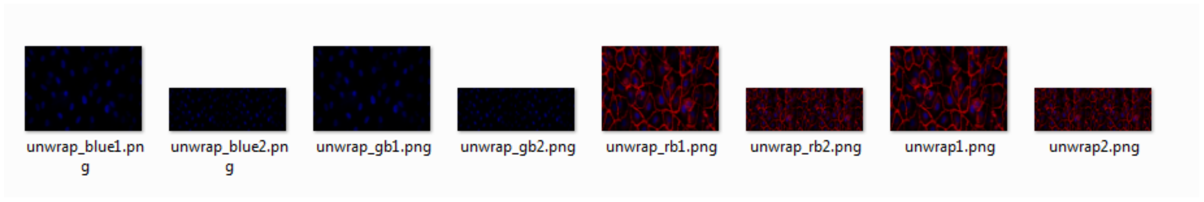
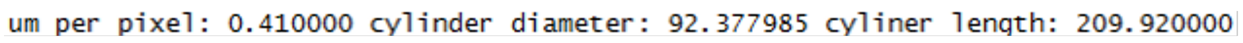


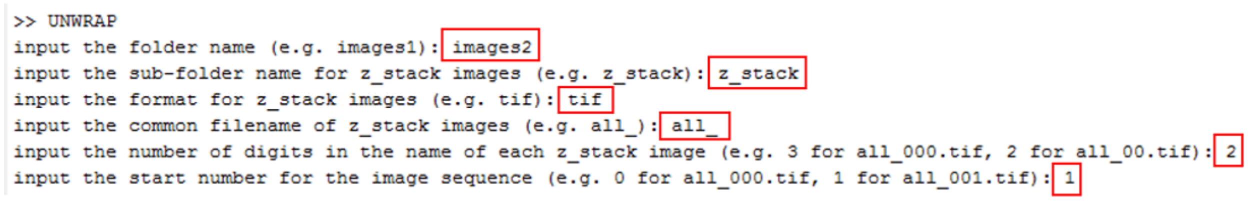
Figure 13. All unwrapped images are stored in the "result" folder.



```
um per pixel: 0.410000 cylinder diameter: 92.377985 cyliner length: 209.920000
```

Figure 14. Information for your input cylindrical structure, including the resolution (0.41 µm/pixel), cylinder diameter (92.4 µm), the length of the cylinder in axial direction (209.9 µm).

To apply this program to another image set, simply create a folder with the image sequence and "scale.txt" file in the same structure as described above, into the working folder (folder names can be different). For example, a practice image folder named "images2" is provided in the working folder. In the example in the "images2" folder, the z-stack represents only part of the cross section of a cylinder. In this case, the points selected to define the cross section should only be located on the section of the cylinder that is imaged. The information you need for this input folder is provided below (Figure 15).



```
>> UNWRAP
input the folder name (e.g. images1): images2
input the sub-folder name for z_stack images (e.g. z_stack): z_stack
input the format for z_stack images (e.g. tif): tif
input the common filename of z_stack images (e.g. all_): all_
input the number of digits in the name of each z_stack image (e.g. 3 for all_000.tif, 2 for all_00.tif): 2
input the start number for the image sequence (e.g. 0 for all_000.tif, 1 for all_001.tif): 1
```

Figure. 15. Prompts in the command window for the sample image folder "images2". Input data is indicated by the red boxes.

General Instructions

For sample images, if something goes wrong, simply start over to Step 3 by clicking ▷. For your own input images, if something goes wrong, check your folder carefully according to Step 1 and 2.

MATLAB Code

```
% UNWRAP
%
% Mao Ye, Zhen Yang, and Peter C. Searson
% Johns Hopkins University
%
% UNWRAP takes a series of z-stack images of a cylindrical object and unwraps
% the image to create a set of 2D images for quantitative analysis.
% The original images can contain up to three separate channels.  This
% application is useful for unwrapping images of cylindrical objects such
% as small blood vessels.
%
% The main steps in the applications are:
% • Specify data folder, image format and scale info
% • Input a series of z-stack images (up to three channels)
% • Isotropic resample of the 3D volume
% • Crop the volume to focus on the cylinder
% • Fit to a cylinder
% • Unwrap the image on the surface of the cylinder to obtain a set of 2D
images
% • Save unwrapped images and info

function UNWRAP()
%% Specify data folder, image format and scale info
s1 = input('input the folder name (e.g. images1): ','s');
SubjFolder = [s1, '/'];

s2 = input('input the sub-folder name for z_stack images (e.g. z_stack):
','s');
ImgFolder = [SubjFolder s2, '/'];

s3 = input('input the format for z_stack images (e.g. tif): ','s');
ImgFmt = s3; % format of input z-stack images

s4 = input('input the common filename of z_stack images (e.g. all_): ','s');
filename = s4; % common name of input images, CHANGE if filename is different!

s6 = input('input the number of digits in the name of each z_stack image (e.g.
3 for all_000.tif, 2 for all_00.tif): ','s');
digit = ['%0', s6, 'd'];% number of digits contained in names of input images,
CHANGE if number of digits is different!

s7 = input('input the start number for the image sequence (e.g. 0 for
all_000.tif, 1 for all_001.tif): ','s');
start = 1; % if 0, name starts from "all_000.tif"; if 1, start from
"all_001.tif", CHANGE if the start number is different!


imageNames = dir(fullfile(ImgFolder,'images','*.tif'));
imageNames = {imageNames.name}';



% information for picking two slices in y-direction (perpendicular to
cylinder)
slice_show1 = 100; % slice numberfrom one end (no need to change)
slice_show2 = 100; % slice number from the other end (no need to change)

% create a result folder if there's none
RsltFolder = [SubjFolder 'unwrapped_images/'];
if ~exist(RsltFolder); mkdir(SubjFolder, 'unwrapped_images'); end


ImgName = [ImgFolder '*.' ImgFmt];
d = dir(ImgName);
```

```
SliceNos = 1 : (length(d));
SliceNum = length(SliceNos);

%image scale information
A  =  load([SubjFolder  'scale.txt']);  %  first  row:  xy  resolution  (µm/px);
second row: z-spacing between z-stack images (µm)
yScale = A(1);
zScale = A(2);
zRatio = zScale/yScale;  % z resolution / xy resolution um/px

%% Input a series of z-stack images (up to three channels)
% read one z-stack image to get the image size
s = 1;
FileName = [ImgFolder filename sprintf(digit, s) '.' ImgFmt];
Img = imread(FileName);
[Ny,Nx,Nc] = size(Img);

% prepare 3D volume for 3 channels (RGB)
RVol = zeros(Ny,Nx,SliceNum);
GVol = zeros(Ny,Nx,SliceNum);
BVol = zeros(Ny,Nx,SliceNum);

% load z-stack images of a cylindrical 3D object
display('read in slices. z = : ')
for i = 1: SliceNum
    s = start + SliceNos(i) - 1;
    fprintf('%d\t', s)
    if mod(s,5)==4
        fprintf('\r')
    end
    FileName = [ImgFolder filename sprintf(digit, s) '.' ImgFmt];
    %display(s);
    RGBImg = imread(FileName);
    RVol(:,:,i) = RGBImg(:,:,1);
    GVol(:,:,i) = RGBImg(:,:,2);
    BVol(:,:,i) = RGBImg(:,:,3);
end
fprintf('\r\r')

% normalize the image intensity to [0 1]
RVol = double(RVol)/256;
GVol = double(GVol)/256;
BVol = double(BVol)/256;
[Ny, Nx, Nz0] = size(RVol);

%% Isotropic resample of the 3D volume
Nz  =  round(Nz0*zRatio);  %  target  dimension  in  z-direction  after  isotropic
resampling
% prepare empty isotropic 3D volumes
IsoRVol = zeros(Ny,Nx,Nz);
IsoGVol = zeros(Ny,Nx,Nz);
IsoBVol = zeros(Ny,Nx,Nz);

% resize slice by slice in y direction
display('generate isotropic volume. y = : ')
```

```
for i = 1 : Ny
    fprintf('%d\t', i)
    if mod(i,5)==4
        fprintf('\r')
    end
    % R channel
    I = squeeze(RVol(i,:,:));
    I = imresize(I,[Nx,Nz], 'bicubic');
    IsoRVol(i,:,:) = I;
    % G channel
    I = squeeze(GVol(i,:,:));
    I = imresize(I,[Nx,Nz], 'bicubic');
    IsoGVol(i,:,:) = I;
    % B channel
    I = squeeze(BVol(i,:,:));
    I = imresize(I,[Nx,Nz], 'bicubic');
    IsoBVol(i,:,:) = I;
end
fprintf('\r\r')

%% Crop the volume to focus on the cylinder
display('crop image')
y = slice_show1; % pick a slice in y-direction

hf = figure;
% create the color image for the slice for better visualization
I = cat(3, squeeze(IsoRVol(y,:,:)), ...
           squeeze(IsoGVol(y,:,:)), ...
           squeeze(IsoBVol(y,:,:)));
[X,Y,I2,rect] = imcrop(I);

% record the cropped rectangle
zmin = rect(1);
zmax = rect(1) + rect(3);
xmin = rect(2);
xmax = rect(2) + rect(4);

% convert to positive integer
zmin = max(1, floor(zmin));
zmax = min(Nz, ceil(zmax));
xmin = max(1, floor(xmin));
xmax = min(Nx, ceil(xmax));

% crop the 3d volume according to the rectangle
IsoRVol = IsoRVol(:, xmin:xmax, zmin:zmax);
IsoGVol = IsoGVol(:, xmin:xmax, zmin:zmax);
IsoBVol = IsoBVol(:, xmin:xmax, zmin:zmax);
[Ny,Nx,Nz] = size(IsoRVol);

%% Fit to a cylinder

LEFT = 1; MIDDLE = 2; RIGHT = 3;
t = 0:pi/360:2*pi;

display('fit circle on one end')
```

```
y = slice_show1; % pick a slice near one end of the cylinder

hf = figure;
% create the color image for the slice for better visualization
I = cat(3, squeeze(IsoRVol(y,:,:)), ...
           squeeze(IsoGVol(y,:,:)), ...
           squeeze(IsoBVol(y,:,:)));
imshow(I, 'initialmagnification',50)
title(['slice y = ' num2str(y)])
hold on
P = [];
% pick points
[x,y,Button] = ginput(1);
while(Button == LEFT)
    P = [P; [x y]];
    plot(x, y, 'g+');
    [x,y,Button] = ginput(1);
end
% fit circle
% call the Circle Fit (Taubin method) from MatLab File Exchange
CirPar = FUN_CircleFitByTaubin(P);
% plot fitted circle
X = CirPar(1) + CirPar(3)*cos(t);
Y = CirPar(2) + CirPar(3)*sin(t);
plot(X,Y,'y')
hold off
% record the parameters for the first circle
CirPar1 = CirPar;


display('fit circle on the other end')
y = Ny-slice_show2; % pick a slice near the other end of the cylinder

figure;
% create the color image for the slice for better visualization
I = cat(3, squeeze(IsoRVol(y,:,:)), ...
           squeeze(IsoGVol(y,:,:)), ...
           squeeze(IsoBVol(y,:,:)));
imshow(I)
title(['slice y = ' num2str(y)])
hold on
P = [];
% pick points
[x,y,Button] = ginput(1);
while(Button == LEFT)
    P = [P; [x y]];
    plot(x, y, 'g+');
    [x,y,Button] = ginput(1);
end
% fit circle
CirPar = FUN_CircleFitByTaubin(P);
% plot fitted circle
X = CirPar(1) + CirPar(3)*cos(t);
Y = CirPar(2) + CirPar(3)*sin(t);
plot(X,Y,'y')
hold off
```

```
% record the parameters for the second circle
CirPar2 = CirPar;

% average the parameters of the two circles
CirPar = (CirPar1 + CirPar2)/2;

%% Unwrap the image on the surface of the cylinder to obtain a set of 2D
images

Rc = CirPar(3); % radius of the cylinder in the image
% number of angles to sample around the cylinder axis
dTheta = 1/Rc;
MinTheta = 0;
MaxTheta = 2*pi-dTheta;
Theta = MinTheta : dTheta : MaxTheta;
ThetaNum = length(Theta);

% number of sample points along cylinder axis
dHeight = dTheta*Rc;
MinHeight = 1;
MaxHeight = Ny;
Height = MinHeight : dHeight : MaxHeight;
HeightNum = length(Height);

Rot = eye(3);
t0 = [CirPar(2)  0  CirPar(1)]';

% smoothing parameter
r = 2; %2
s = (2*r+1)^3;
sigma = 1;

% prepare empty unwrap images for three channels
IsoRGrid = zeros(HeightNum, ThetaNum);
IsoGGrid = zeros(HeightNum, ThetaNum);
IsoBGrid = zeros(HeightNum, ThetaNum);

XX = zeros(HeightNum, ThetaNum);
YY = zeros(HeightNum, ThetaNum);
ZZ = zeros(HeightNum, ThetaNum);

for it = 1 : ThetaNum % sample around cylinder axis
    theta = Theta(it);
    for ih = 1 : HeightNum % sample along cylinder axis
        % compute the spatial coordinate of sample point p0
        height = Height(ih);
        p0 = [Rc*sin(theta)
              height
              Rc*cos(theta)];
        p0 = Rot*p0 + t0;
        p0 = round(p0);
        x0 = p0(1); y0 = p0(2); z0 = p0(3);

        XX(ih,it) = x0;
        YY(ih,it) = y0;
        ZZ(ih,it) = z0;

        % coordinate span of p0's neighborhood
        xspan = max(x0-r,1):min(x0+r,Nx);
        yspan = max(y0-r,1):min(y0+r,Ny);
        zspan = max(z0-r,1):min(z0+r,Nz);

        if isempty(xspan)||isempty(yspan)||isempty(zspan)
            IsoRGrid(ih,it) = 0;
            IsoGGrid(ih,it) = 0;
            IsoBGrid(ih,it) = 0;
        end

        % average pixel intensity in p0's neighborhood and assign to the
        % correponding pixel in the unswrapped image
        I = IsoRVol(yspan, xspan, zspan);
        IsoRGrid(ih,it) = sum(I(:))/s;
        I = IsoGVol(yspan, xspan, zspan);
        IsoGGrid(ih,it) = sum(I(:))/s;
        I = IsoBVol(yspan, xspan, zspan);
        IsoBGrid(ih,it) = sum(I(:))/s;
    end
end

%% Save unwrapped images and info
RGBGrid1 = cat(3,IsoRGrid, IsoGGrid, IsoBGrid);
RGBGrid2 = cat(3,repmat(IsoRGrid,1,2), ...
    repmat(IsoGGrid,1,2), ...
    repmat(IsoBGrid,1,2));

RGBGrid3 = cat(3,zeros(size(IsoRGrid)), zeros(size(IsoGGrid)), IsoBGrid);
RGBGrid4 = cat(3,repmat(zeros(size(IsoRGrid)),1,2), ...
    repmat(zeros(size(IsoGGrid)),1,2), ...
    repmat(IsoBGrid,1,2));

RGBGrid5 = cat(3,IsoRGrid, zeros(size(IsoRGrid)), IsoBGrid);
RGBGrid6 = cat(3,repmat(IsoRGrid,1,2), ...
    repmat(zeros(size(IsoRGrid)),1,2), ...
    repmat(IsoBGrid,1,2));

RGBGrid7 = cat(3,zeros(size(IsoRGrid)), IsoGGrid, IsoBGrid);
RGBGrid8 = cat(3,repmat(zeros(size(IsoRGrid)),1,2), ...
    repmat(IsoGGrid,1,2), ...
    repmat(IsoBGrid,1,2));

figure
imshow(RGBGrid1)
axis image

% save unwrapped images
imwrite(RGBGrid1, [RsltFolder 'unwrap1.png'], 'png');
imwrite(RGBGrid2, [RsltFolder 'unwrap2.png'], 'png');
imwrite(RGBGrid3, [RsltFolder 'unwrap_blue1.png'], 'png');
imwrite(RGBGrid4, [RsltFolder 'unwrap_blue2.png'], 'png');
imwrite(RGBGrid5, [RsltFolder 'unwrap_rb1.png'], 'png');
imwrite(RGBGrid6, [RsltFolder 'unwrap_rb2.png'], 'png');
imwrite(RGBGrid7, [RsltFolder 'unwrap_gb1.png'], 'png');
imwrite(RGBGrid8, [RsltFolder 'unwrap_gb2.png'], 'png');

% save unwrap info
fid = fopen([SubjFolder 'info.txt'], 'w');
fprintf(fid, 'um per pixel: %f \n', yScale);
fprintf(fid, 'cylinder diameter: %f \n', 2*Rc*yScale);
fprintf(fid, 'cyliner length: %f \n', Ny*yScale);
fclose(fid);


end




% The following routine is obtained from MATLAB file exchange.
%       http://www.mathworks.com/matlabcentral/fileexchange/22678-circle-fit-
taubin-method

function Par = FUN_CircleFitByTaubin(XY)

%--------------------------------------------------------------------------
%
%     Circle fit by Taubin
%      G. Taubin, "Estimation Of Planar Curves, Surfaces And Nonplanar
%                  Space Curves Defined By Implicit Equations, With
%                  Applications To Edge And Range Image Segmentation",
%      IEEE Trans. PAMI, Vol. 13, pages 1115-1138, (1991)
%
%      Input:  XY(n,2) is the array of coordinates of n points x(i)=XY(i,1),
y(i)=XY(i,2)
%
%     Output: Par = [a b R] is the fitting circle:
%                           center (a,b) and radius R
%
%     Note: this fit does not use built-in matrix functions (except "mean"),
%           so it can be easily programmed in any programming language
%
%--------------------------------------------------------------------------

n = size(XY,1);      % number of data points

centroid = mean(XY);   % the centroid of the data set

%     computing moments (note: all moments will be normed, i.e. divided by n)

Mxx = 0; Myy = 0; Mxy = 0; Mxz = 0; Myz = 0; Mzz = 0;

for i=1:n
    Xi = XY(i,1) - centroid(1);  %  centering data
    Yi = XY(i,2) - centroid(2);  %  centering data
    Zi = Xi*Xi + Yi*Yi;
    Mxy = Mxy + Xi*Yi;
    Mxx = Mxx + Xi*Xi;
    Myy = Myy + Yi*Yi;
    Mxz = Mxz + Xi*Zi;
    Myz = Myz + Yi*Zi;
    Mzz = Mzz + Zi*Zi;
end

Mxx = Mxx/n;
Myy = Myy/n;
Mxy = Mxy/n;
Mxz = Mxz/n;
Myz = Myz/n;
Mzz = Mzz/n;

%    computing the coefficients of the characteristic polynomial

Mz = Mxx + Myy;
Cov_xy = Mxx*Myy - Mxy*Mxy;
A3 = 4*Mz;
A2 = -3*Mz*Mz - Mzz;
A1 = Mzz*Mz + 4*Cov_xy*Mz - Mxz*Mxz - Myz*Myz - Mz*Mz*Mz;
A0 = Mxz*Mxz*Myy + Myz*Myz*Mxx - Mzz*Cov_xy - 2*Mxz*Myz*Mxy + Mz*Mz*Cov_xy;
A22 = A2 + A2;
A33 = A3 + A3 + A3;

xnew = 0;
ynew = 1e+20;
epsilon = 1e-12;
IterMax = 20;

% Newton's method starting at x=0

for iter=1:IterMax
    yold = ynew;
    ynew = A0 + xnew*(A1 + xnew*(A2 + xnew*A3));
    if abs(ynew) > abs(yold)
       disp('Newton-Taubin goes wrong direction: |ynew| > |yold|');
       xnew = 0;
       break;
    end
    Dy = A1 + xnew*(A22 + xnew*A33);
    xold = xnew;
    xnew = xold - ynew/Dy;
    if (abs((xnew-xold)/xnew) < epsilon), break, end
    if (iter >= IterMax)
        disp('Newton-Taubin will not converge');
        xnew = 0;
    end
    if (xnew<0.)
        fprintf(1,'Newton-Taubin negative root:  x=%f\n',xnew);
        xnew = 0;
    end
end
```

```
%  computing the circle parameters

DET = xnew*xnew - xnew*Mz + Cov_xy;
Center = [Mxz*(Myy-xnew)-Myz*Mxy , Myz*(Mxx-xnew)-Mxz*Mxy]/DET/2;

Par = [Center+centroid , sqrt(Center*Center'+Mz)];

end    %    CircleFitByTaubin
```