1. Open your image in FIJI (Schindelin et al., 2012) and split the phase and FDAA channels for each image. Save them into separate folders with the same name for each image
2. Start Oufti (Paintdakhi et al., 2016) from terminal

In OUFTI:

1. Load phase images and load signal 1 (FDAA images). Unclick stack and click on independent frames
2. Create and load a Parameters.set according to the Oufti manual (http://oufti.org/) that best suits selection of single cells of your specific bacterial species
3. Click -> File and create an output.mat file
4. Check segmentation and toggle parameters to fit as many cells (viewing the phase images) as possible, and reduce the number of random clumps being called as cells
5. Once segmentation looks good
   1. Click ->All frames. This will make outlines around all cells. This can take a few minutes
6. Clicking through each frame:
   1. Delete unwanted outlines (those around clumps) by clicking on them (they turn red) and hitting the delete button.
   2. Delete cells that are not in focus as well

*OR you can manually select cells*

* 1. *Click ->Add (at the bottom of the Oufti window)*
  2. *Click on one pole of the cell in the phase image. A red line segment will be made. Click along the cell to the opposite pole. Double click and an outline will be made*
  3. If you manually selected the cells you may want to look at them individually. If an outline is not fitting a cell properly, select it and click refine. You may need to click refine a few times depending on the cell

1. When done deleting unwanted outlines
   1. Click-> Refine all
2. When that is done, click -> subtract background (all)
3. Click ->reuse meshes
4. Now click ->compute phase profile as [0]
   1. Compute signal 1 profile as [1]
5. Click -> all frames
6. Click -> save analysis
7. Open MATLAB
8. Follow run\_Kasia.m

**run\_Kasia.m**

cellList\_KB207=cellList.meshData;

cellList\_KB207=cellListMod(cellList\_KB207);

cellList\_KB207 = peakfinder(cellList\_KB207,0.10,0.10,12,[],[],[],[],[]);

cellList\_KB207=adjust\_peakfinder(cellList\_KB207);

%switch pole

cellList\_KB207=plotsignal(cellList\_KB207,1);

%cell Length in pix

cellLength\_KB207=getCellLength(cellList\_KB207);

%cell length in um

cellLength\_KB207\_um=cellLength\_KB207\*0.07222198;

% 0.07222198 is the um/pixel for the images

cellList\_KB207\_GF=getFluorescence(cellList\_KB207, 0.07228598, 0);

cellList\_KB207\_GF\_sep=getFluorescence(cellList\_KB207, 0.07228598, 1);

%get profiles for each of the quadrants

KB207\_profile=plotprofile\_kasia(cellList\_KB207,[20 200])

**adjust\_peakfinder.m**

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% This function offset the peakfinder output by one pixel and correct the

% error in the positions field.

% The author of this function is Hoong Chuin Lim, 2015, Harvard Medical

% School, HoongChuin\_Lim@hms.harvard.edu

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function cellList2=adjust\_peakfinder(cellList2)

for frame=1:length(cellList2)

for Cell=1:length(cellList2{frame})

if ~isempty(cellList2{frame}{Cell}) && isfield(cellList2{frame}{Cell},'spots')

vector=cellList2{frame}{Cell}.lengthvector;

spotpos=cellList2{frame}{Cell}.spots.l;

pos=zeros(1,length(spotpos));

l=zeros(1,length(spotpos));

for i=1:length(spotpos)

ind=find(vector==spotpos(i));

pos(1,i)=ind+1; %shift spot index by 1

if ind<length(vector)

l(1,i)=vector(ind+1); %record the correct vectorial position

end

end

cellList2{frame}{Cell}.spots.l=l;

cellList2{frame}{Cell}.spots.positions=pos;

end

end

end

end

**plotsignal.m**

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% plotsignal is a quality control script that does 2 things. 1: Orient the

% cells based on fluorescence at the poles, 2: make 100 plots of individual

% cells for user to inspect if peakfinder and cell reorientation are

% working properly.

%

% This function takes cellList that has been processed with peakfinder and

% adjust\_peakfinder.m as input and

%

% The author of this function is Hoong Chuin Lim, 2015, Harvard Medical

% School, HoongChuin\_Lim@hms.harvard.edu

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function cellList=plotsignal(cellList, plotter)

counter=0;

n=1;

for frame=1:length(cellList)

if ~isempty(cellList{frame})

for Cell=1:length(cellList{frame})

if isfield(cellList{frame}{Cell},'length')

if cellList{frame}{Cell}.length>20

signal=cellList{frame}{Cell}.signal1; %get signal

steparea=cellList{frame}{Cell}.steparea; %get step area

steplength=cellList{frame}{Cell}.steplength; %get steparea

if length(signal)== length(steparea) % quality control

% switch pole if required to put lower signal pole

% at the left

if sum(signal(1:9))>sum(signal(end-8:end))

signal= signal(end:-1:1);

steparea= steparea(end:-1:1);

steplength= steplength(end:-1:1);

cellList{frame}{Cell}.signal1=signal;

cellList{frame}{Cell}.steparea=steparea;

cellList{frame}{Cell}.lengthvector=cumsum(steplength);

spots=cellList{frame}{Cell}.spots;

%adjust spots position accordingly

if ~isempty(spots)

l=spots.l;

pos=spots.positions;

CL=cellList{frame}{Cell}.length;

cellList{frame}{Cell}.spots.l=CL-l;

cellList{frame}{Cell}.spots.positions=length(signal)-pos+1;

end

end

if plotter && counter<100

subplot(5,2, n), counter=counter+1;

plot(cellList{frame}{Cell}.signal1./...

cellList{frame}{Cell}.steparea)

n=n+1;

if isfield(cellList{frame}{Cell},'spots')

%get position of peaks

spots=cellList{frame}{Cell}.spots.positions;

hold on, plot(spots,...

repmat(0.01,1, length(spots)),'.',...

'markersize',5)

end

if n==11

figure, n=1;

end

end

end

end

end

end

end

end

end

**getCellLength.m**

function data=getCellLength(lst)

data=[];

for frame=1:length(lst)

for Cell=1:length(lst{frame})

if isfield(lst{frame}{Cell},'length')

data=[data; lst{frame}{Cell}.length];

end

end

end

end

**getFluorescence.m**

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% This function takes cellList that has been processed with peakfinder and

% adjust\_peakfinder.m., pixel size of the images and septum as inputs.

% To analyze cells with septum: Septum=1, else 0

% Outputs: data which contains 5 matrices, data1, 2 and 3, each recording

% signals for a given number of pixels

% data1, sum signal in 5 pixel

% data2, sum signal in 10 pixel

% data3, sum signal in 15 pixel

% Each of the data set is a matrix, with each row being a cell. Column 1

% records integrated signal for slow growing pole, Column 2 records

% integrated signal for fast growing pole, Column 3 records integrated

% signal within the mid cell regions, Column 4 records the corresponding

% cell length while Column 5 is the ratio between the signal at the fast

% pole vs the signal at the slow pole

%

%

%

% The author of this function is Hoong Chuin Lim, 2015, Harvard Medical

% School, HoongChuin\_Lim@hms.harvard.edu

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function data=getFluorescence(cellList, pixsize, septum)

%% split cells into three groups

%figure

data1=nan(1000,4);

data2=nan(1000,4);

data3=nan(1000,4);

data=[];

n=1;

for frame=1:length(cellList)

if ~isempty(cellList{frame})

for Cell=1:length(cellList{frame})

if isfield(cellList{frame}{Cell},{'length', 'spots'})

signal=cellList{frame}{Cell}.signal1;

steparea=cellList{frame}{Cell}.steparea;

spots=cellList{frame}{Cell}.spots.positions;

cellLength=cellList{frame}{Cell}.length;

if ~isempty(spots) && cellLength>25

skipp=isseptum(spots, cellLength); %detect cells with septums

if septum==1

skipp=~skipp; %

end

if length(signal)==length(steparea) && skipp% get cells with a septum

%normalize signal by length

nsignal=signal./steparea;

%cut off for signal--higher for brighter dyes

kk=nsignal>1;

nsignal(kk)=0;

%sum signal for slow pole, at different depth

data1(n,1)=sum(nsignal(1:5));

data2(n,1)=sum(nsignal(1:10));

data3(n,1)=sum(nsignal(1:15));

%sum signal for fast pole, at different depth

data1(n,2)=sum(nsignal(end-4:end));

data2(n,2)=sum(nsignal(end-9:end));

data3(n,2)=sum(nsignal(end-14:end));

%signal for body

nn=round(length(nsignal)/2);

data1(n,3)=sum(nsignal(nn-2:nn+2));

data2(n,3)=sum(nsignal(nn-4:nn+4));

data3(n,3)=sum(nsignal(nn-7:nn+7));

data1(n,4)=cellLength\*pixsize;

data2(n,4)=cellLength\*pixsize;

data3(n,4)=cellLength\*pixsize;

n=n+1;

end

end

end

end

end

end

data.data1=data1;

data.data2=data2;

data.data3=data3;

data.data1(:,5)=data.data1(:,2)./data.data1(:,1); %ratio: fast over slow

data.data2(:,5)=data.data2(:,2)./data.data2(:,1); %ratio: fast over slow

data.data3(:,5)=data.data3(:,2)./data.data3(:,1); %ratio: fast over slow

end

function skipp=isseptum(peaks, cellLength)

%finding WT cells with single septa that is between 0.3-0.7 in normalized

%position, would not worked for mutants with multiple septa or mutants with

%severely impaired septum positioning

npeaks=peaks/cellLength;

if sum(npeaks>0.3 & npeaks<0.7)>0

skipp=0;

else

skipp=1;

end

end

**plotprofile\_kasia.m**

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% This function takes cellList that has been processed with peakfinder and

% adjust\_peakfinder.m as input.

% Outputs: a figure with n graphs showing average signal profiles and

% data1 which contains the raw data that go into plotting each graph.

%

% The author of this function is Hoong Chuin Lim, 2015, Harvard Medical

% School, HoongChuin\_Lim@hms.harvard.edu

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function data1=plotprofile\_kasia(cellList,cutoff)

%% split cells into three groups

%figure

data1=[]; %final output

xx=0:0.01:1;

data=nan(500,length(xx)); %matrix for each bin

datasepta=nan(500,length(xx)); %special matrix for each bin

n=1; %counter

ns=1; %counter for septa

%cutoff=[50 75 100 200]; % range in the unit pixel length for binning

% bin one: between 50 pix and 75 pix,

% bin two: between 75 pix and 100 pix and so on

plotlng=length(cutoff);

for ii=2:length(cutoff)

fign=ii-1;

for frame=1:length(cellList)

if ~isempty(cellList{frame})

for Cell=1:length(cellList{frame})

if isfield(cellList{frame}{Cell},{'length', 'spots'})

if cellList{frame}{Cell}.length>cutoff(ii-1) &&...

cellList{frame}{Cell}.length<=cutoff(ii)

signal=cellList{frame}{Cell}.signal1;

steparea=cellList{frame}{Cell}.steparea;

spots=cellList{frame}{Cell}.spots.positions;

cellLength=cellList{frame}{Cell}.length;

if ~isempty(spots)

skipp=isseptum(spots, cellLength);

% pass quality and no septa

if length(signal)==length(steparea)

%normalize signal

nsignal=signal./steparea;

%normalize length

nlength=cellList{frame}{Cell}.lengthvector/...

cellList{frame}{Cell}.length;

%interpolate signal

qsignal=interp1(nlength, nsignal, xx);

if skipp

data(n,:)=qsignal'; n=n+1;

else

datasepta(ns,:)=qsignal'; ns=ns+1;

end

end

end

end

end

end

end

end

subplot(plotlng,1,fign), hold on

for zz=1:200

plot(xx, data(zz,:), 'color', [.25 .25 .25], 'linewidth', .04)

end

plot(xx, nanmean(data),'g', 'linewidth', 5), ylim([0 0.025])

sample=sum(~isnan(nanmean(data')));

title(['n = ' num2str(sample)], 'fontsize', 14)

data1{fign}.data=data;

data1{fign}.cutoff=[cutoff(ii-1), cutoff(ii)];

data1{fign}.xx=xx;

data1{plotlng+1}.mean(fign,:)=nanmean(data);

data=nan(200,length(xx)); n=1;

end

%plot septa

subplot(plotlng,1,plotlng), hold on

%assign mean to output

for zz=1:200

plot(xx, datasepta(zz,:), 'color', [.45 .45 .45], 'linewidth', .04)

end

plot(xx, nanmean(datasepta),'g', 'linewidth', 5), ylim([0 0.025])

data1{plotlng}.data=datasepta; %assign septa to output

data1{plotlng}.xx=xx;

data1{plotlng+1}.mean(fign+1,:)=nanmean(datasepta);

sample=sum(~isnan(nanmean(datasepta')));

title(['n = ' num2str(sample)], 'fontsize', 14)

end

function skipp=isseptum(peaks, cellLength)

npeaks=peaks/cellLength;

if sum(npeaks>0.3 & npeaks<0.7)>0

skipp=0;

else

skipp=1;

end

end

Paintdakhi, A., Parry, B., Campos, M., Irnov, I., Elf, J., Surovtsev, I., & Jacobs-Wagner, C. (2016). Oufti: an integrated software package for high-accuracy, high-throughput quantitative microscopy analysis. *Molecular Microbiology*, *99*(4), 767–777. http://doi.org/10.1111/mmi.13264

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., et al. (2012). Fiji: an open-source platform for biological-image analysis. *Nature Methods*, *9*(7), 676–682. http://doi.org/10.1038/nmeth.2019