Bleb or GPMV Analysis

Thomas S. van Zanten

29/04/2021

1 Analysis Aims

The aim of the analysis software is to obtain an intensity profile along the membrane of a GPMV or bleb. The profile itself can subsequently be used for cross-correlation or any other mathematical operation with respect to another channel of the same GPMV/bleb. The intensity can also be summed up (for total photons count) or the mean can be calculated (which can be converted to count rate if the camera integration time is know). Radii and circularity of each GPMV or bleb will be added to the property list.

2 Program Design

The required set of Matlab analysis files contains .m-files that are both specific:

- main_bleb.m
- OpenUpBleb.m

and general:

- tiffread2.m
- ImageCorrection.m
- PhotoConvertIm.m (will additionally need DIPlib and PCFO if the users wants to automatically determine the used EMCCD gains)
- DualCh_align.m
- ROIselect.m

The set of files require some basic understanding of Matlab but is annotated to help understand the flow. The code is semi-automated because the user still is interacting at several steps and the code is recommended to be run section by section from the main file: main_bleb.m.

2.1 Main code: main_bleb.m

To be able to quantitatively address several features of cell-attached blebs that were generated using Jasplakinolide or GPMVs via the PFA/DTT-method there are several imaging conditions to take care of. Obviously detector saturation should be avoided but the signal should be above noise and background in all channels. Ideally the data should be obtained with a confocal microscope. Whereas the input for the code should be a single slice, z-stacks of cell-attached will allow the identification of more blebs from a single cell. If bleb identification and image cropping is done using another software like imageJ section 2 can be skipped. In that case do make sure each channel from the respective bleb image is saved as *Bleb(i).im1* and *Bleb(i).im2* for channel 1 and 2, respectively). For full single slice images the below code is divided in 5 main sections:

- Section 0: Initialisation
- Section 1: Variable and image loading and conversion
- Section 2: Indicating the Blebs to be analysed per image
- Section 3: Centering, aligning and opening up the Bleb for analysis
- Section 4: Analyzing all the Blebs and extracting relevant data

The first section starts by initialising MatLab and indicating all the paths where the functions have been saved. Please make sure you do not have multiple functions with the same name at different locations as this could make any adaptations you make to any function ineffective (MatLab could take the other function during the process).

```
%% Section 0: Initialisation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
addpath(genpath('path where all the general functions are located'))
addpath(genpath('path where all the specific functions are located'))
pathname=uigetdir; pathname=[pathname '/']; cd(pathname)
```

The second section sets the variables (line 10-20), camera details (line 29 & 30), the filename series (line 22 & 24) and the places for background and Gfactor (line 37-38 & 44-45). The example below follows a Laurdan experiment. Please note that the individual background and Gfactor images from each channel are saved in a BG-folder that is placed in the path of the data/experiment files. These individual files are ideally the average of a series of 5-10 images.

```
9   %% Section 1: Variable and image loading and conversion %%%%%%%%%%%%%%%%%%
10  ws=80;%width of the ROI. Is either 60 or 80 during bleb selection process
11  minR=5;%minimum radius
12  maxR=30;%maximum radius: depends largely on the ROI size
13  blur=3;%gaussian blur during for circle determination
14  av=6;%size rolling average along the membrane for re-allignment
15  %--> should be an EVEN number
16  width=10;%width around the bleb membrane to allow deformations to be fitted
17  %--> should be an EVEN number
18  avM=6;%thickness of the membrane + PFS (to include in the membrane summation)
19  %--> should be an EVEN number and smaller than width
20  Bleb.area=[];%initializing the Bleb array to prevent errors in ROIselect.m
```

The functions used in Section 1 include *PhotoConvertIm.m* to convert all the camera data from ADU to photo-electrons and *ImageCorrection.m* which corrects all the images in terms of background and Gfactor, if required. No alignment is performed at this stage because it is assumed that distinct blebs might show different displacements between channels. Thererfore, aligning is done at the end on a per-bleb basis.

```
22  temp_im1=tiffread2b([pathname 'Ldn_blue.tif'],1,40);
23  for i=1:length(temp_im1), im1(i,:,:)=temp_im1(i).data; end
24  temp_im2=tiffread2b([pathname 'Ldn_red.tif'],1,40);
25  for i=1:length(temp_im2), im2(i,:,:)=temp_im2(i).data; end
26  %temp_im3=tiffread2b([pathname 'Mov640_xlink.tif'],1,40);
27  %for i=1:length(temp_im3), im3(i,:,:)=temp_im3(i).data; end
28
29  cam.mode=0; cam.serialNo='0000'; cam.tInt='200ms';%camera settings for more info
30  cam.ROI=[0 0 size(im1,2) size(im1,3)];%see PhotoConvertIm.m
31
32  [im1, gain_ch1, offset_ch1] = PhotoConvertIm (im1, cam);
33  [im2, gain_ch2, offset_ch2] = PhotoConvertIm (im2, cam);
34  %[im3] = PhotoConvertIm (im3, cam, gain_ch2, offset_ch2);
35  %assuming ch3 is taken with camera of ch2
36
37  im1BG=tiffread2([pathname 'BG/BG_blue.tif']);im1BG=double(im1BG.data);
38  im2BG=tiffread2([pathname 'BG/BG_red.tif']);im2BG=double(im2BG.data);
39  %im3BG=tiffread2([pathname 'BG/BG_640.tif']);im3BG=double(im3BG.data);
40  %im3BG = PhotoConvertIm (im3BG, cam, gain_ch2, offset_ch2);
41  im1BG = PhotoConvertIm (im1BG, cam, gain_ch1, offset_ch1);
42  im2BG = PhotoConvertIm (im2BG, cam, gain_ch2, offset_ch2);
43  BG(1,:,:) = im1BG; BG(2,:,:) = im2BG;
44  im1GF=tiffread2([pathname 'BG/GF_blue.tif']);im1GF=double(im1GF.data);
45  im2GF=tiffread2([pathname 'BG/GF_red.tif']);im2GF=double(im2GF.data);
46  im1GF = PhotoConvertIm (im1GF, cam, gain_ch1, offset_ch1);
47  im2GF = PhotoConvertIm (im2GF, cam, gain_ch2, offset_ch2);
48  GF(1,:,:) = im1GF; GF(2,:,:) = im2GF;
49
50      im1=ImageCorrection(im1,im1BG);
51      [im2 GFactor]=ImageCorrection(im2,BG,'Ldn',GF);
52      %im3=ImageCorrection(im3,im3BG);
```

Section 2 allows the user to pinpoint the blebs in each image of the image-stack. After identifying the background in an image by selecting a portion of the image that does not contain signal blebs can (but dont have to) be selected. On confirming *Do you want to select another single ROI* the user can continue to select blebs. Once *NO* is selected the next image will be offered, etc..

```
59  %% Section 2: Indicating the Blebs to be analysed per image %%%%%%%%%%%%%%%
60  for j=1:size(im1,1)
61
62      ch1(:,:)=im1(j,:,:);
63      ch2(:,:)=im2(j,:,:);
```

```
64          %ch3(:,:)=im3(j,:,:);
65
66          No=size(Bleb,2);
67
68          figure(9)
69          set(gcf, 'Position',  [400 50 950 900])
70          imagesc(ch1+ch2);
71          %imagesc(ch1+ch2+ch3);
72              title(['Image ' num2str(j) ' of ' num2str(size(im1,1)) ' images: PLEASE DRAW BACKGROUND'])
73
74      roi = imfreehand%%%Draw a single region that will become the image-associated background
75
76      BW=createMask(roi);
77      BG(1)=sum(sum(BW.*ch1))/sum(sum(BW));
78      BG(2)=sum(sum(BW.*ch2))/sum(sum(BW));
79      %BG(3)=sum(sum(BW.*ch3))/sum(sum(BW));
80      delete(roi);
```

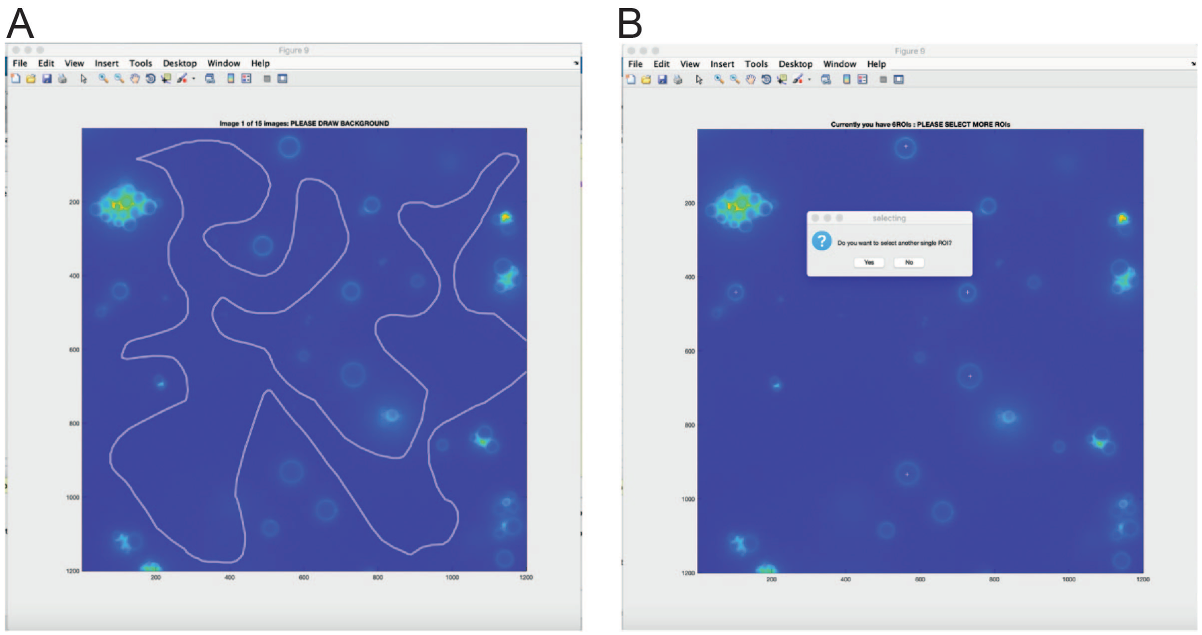

Image 1 of 15 images: PLEASE DRAW BACKGROUND

Currently you have 6ROIs : PLEASE SELECT MORE ROIs

Do you want to select another single ROI?

Figure 1: **Identifying background and blebs/GPMVs (A)** Draw a region where there is no signal so as to get a handle on the background in the different channels of the image. **(B)** Identify all the different blebs or GPMVs in each image.

After all the blebs have been identified the bleb-ROIs will be saved. Next, the program will move to section 3 and go bleb-by-bleb aligning all the channels (line 108-109) and will identify the exact center and radius of the bleb.

```
100     %% Section 3: Centering, aligning and opening up the Bleb for analysis %%%
101     for i=1:size(Bleb,2)
102     %% Alligning different channels in Bleb ROI
103         ch1(:,:)=Bleb(i).im1; ch1(find(ch1<0))=0;
104         ch2(:,:)=Bleb(i).im2; ch2(find(ch2<0))=0;
105         %ch3(:,:)=Bleb(i).im3; ch3(find(ch3<0))=0;
106     ws=size(ch2,1);
107
108             [ch2] = DualCh_align (ch1, ch2, 'translation', 1);
109             %[ch3] = DualCh_align (ch1, ch3, 'translation');
```

The center and radius of the bleb is identified automatically with *imfindcircles* and will be indicated in the figure as a red circle. This works very well for free standing GPMVs but becomes more difficult for crowded images or cell-attached blebs. For these more tricky situations there is the possibility to manually adjust the values (uncomment line 132-143) or to manually draw a rectangle on the bleb (uncomment line 145-147). When drawing the rectangle be aware that the top-left corner will be associated to the center and the horizontal width of the rectangle to the radius of the circle. Next in this section the bleb will be converted from a circle to a rectangular carpet using *OpenUpBleb.m*. See details on this specific functions below.

```
112     %% FREESTANDING / AUTOMATED Bleb find
113     imblur = imgaussfilt(im,blur);
114     [centers_out, radii_out, metric_out] = imfindcircles(imblur,[minR maxR],...
115             'ObjectPolarity','bright','Sensitivity',0.85,'EdgeThreshold', 0.1);
116     %'Sensitivity' and/or 'EdgeThreshold' both need a value between 0:1 and proper
117     %identification is dependent on raw data
118     tempmin=mean((ws/2-centers_out).^2,2);%in the possible case that multiple
119     %circles are found the one closest to the slected center position is chosen
120     centers_out=centers_out(find(tempmin==min(tempmin)),:);
121     radii_out=radii_out(find(tempmin==min(tempmin)),:);
122     metric_out=metric_out(find(tempmin==min(tempmin)),:);
123     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124         figure(11)
125         set(gcf, 'Position',  [730, 55, 950, 900])
126         imagesc(im)
127         title(['Bleb ' num2str(i) ' of ' num2str(size(Bleb,2))])
128
129         hold on
130         h=viscircles(centers_out, radii_out,'EdgeColor','r');
131     %% Possibility to change the calculated values
132     %       A{1,1} = num2str(1);
133     %                 A{1,1} = num2str(centers_out(1));
134     %                 A{2,1} = num2str(centers_out(2));
135     %                 A{3,1} = num2str(radii_out);
136     %
137     %                 A{1,2} = 'x-pos';
138     %                 A{2,2} = 'y-pos';
139     %                 A{3,2} = 'radius';
140     % prompt = A(1:3,2); def = A(1:3,1); AddOpts.Resize = 'on';
141     % AddOpts.WindowStyle = 'normal'; AddOpts.Interpreter = 'tex';
142     % Parameters(1:3) = inputdlg(prompt,'Please change if required:', 1, def, AddOpts);
143     % centers_out=[str2num(Parameters{1}), str2num(Parameters{2})]; radii_out=str2num(Parameters{3});
144     %% CELL ATTACHED / MANUAL Bleb find
145     % r=imrect;
146     % Parameters=r.getPosition; delete(r)
147     % centers_out=[Parameters(1), Parameters(2)]; radii_out=Parameters(3);
148     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149     close Figure 11
150         figure(1)
151         set(gcf, 'Position',  [730, 55, 950, 900])
152         subplot(2,3,1)
153         imagesc(im)
154         title(['Bleb ' num2str(i) ' of ' num2str(size(Bleb,2))])
155
156         hold on
157         h=viscircles(centers_out, radii_out,'EdgeColor','b');
158     %% Open up Bleb circle to a Bleb rectangular image
159         [Bleb(i).ini, Bleb(i).ind] = OpenUpBleb(im, av, width, centers_out, radii_out);
160         [Bleb(i).ch1, ind] = OpenUpBleb(ch1, av, width, centers_out, radii_out, Bleb(i).ind);
161         [Bleb(i).ch2, ind] = OpenUpBleb(ch2, av, width, centers_out, radii_out, Bleb(i).ind);
162         %[Bleb(i).ch3, ind] = OpenUpBleb(ch3, av, width, centers_out, radii_out, Bleb(i).ind);
```

If during the process there is an error, section 3 can be run again after specifically running the following bit of code:

```
170     clear r A def Parameters prompt h AddOpts imblur im ch1 ch2 ch3...
171         ind centers_out radii_out metric_out tempmin
172     close Figure 1
```

After this the new starting bleb should get the number of the troublesome bleb (the No where the user got stuck) for a retry or that number +1 for the series to skip the troublesome bleb. This number should be changed in line 101, see above (in place of 1). Also after section 3 is done there is the possibility to remove blebs that were not analysed to the users satisfaction. During the analysis the number of those bleb can be written down from the title in any figure (For example: Bleb number 11 of n) and after the analysis is finished write (or uncomment the following lines):

```
%Bleb([11, 18])=[];%This will delete the indicated blebs from the file
```

Also if at any step later on in the code the user feels that a particular GPMV or bleb was not analyzed properly the adjusted line from above can be typed into the command window to erase the data from that Bleb. This will only be useful if the occasional GPMV is not analyzed properly. For consistent issues please find the reason and adjust the code so as to solve the issue.

Finally, the last section, section 4, extracts all the relevant information from each of the bleb and allows any kind of further quantification. The data manipulation is off course not limited to what is shown below:

```
%% Section 4: Analyzing all the Blebs and extracting relevant data %%%%%%%%
j=1;
for i=1:length(Bleb)
    if length(Bleb(i).ch1)>0
%trace extraction
    Bleb(i).ch1Trace=sum(Bleb(i).ch1((width/2)+1-(avM/2):...
        (width/2)+1+(avM/2),:),1)-(avM+1)*Bleb(i).BG(1);
    Bleb(i).ch2Trace=sum(Bleb(i).ch2((width/2)+1-(avM/2):...
        (width/2)+1+(avM/2),:),1)-(avM+1)*Bleb(i).BG(2);
    %Bleb(i).ch3Trace=sum(Bleb(i).ch3((width/2)+1-(avM/2):...
        %(width/2)+1+(avM/2),:),1)-(avM+1)*Bleb(i).BG(3);
    Bleb(i).totTrace=Bleb(i).ch1Trace+Bleb(i).ch2Trace;

%An value for anisotropy calculation
    %Bleb(i).anTrace=(Bleb(i).ch1Trace-Bleb(i).ch2Trace)./Bleb(i).totTrace;
%GP value for Laurdan calculation
    Bleb(i).gpTrace=(Bleb(i).ch1Trace-Bleb(i).ch2Trace)./Bleb(i).totTrace;

%trace correlation
    %[Bleb(i).IntR, Bleb(i).IntP] = corrcoef(Bleb(i).totTrace,Bleb(i).ch3Trace);
    %[Bleb(i).anR, Bleb(i).anP] = corrcoef(Bleb(i).anTrace, Bleb(i).ch3Trace);
    %[Bleb(i).gpR, Bleb(i).gpP] = corrcoef(Bleb(i).gpTrace, Bleb(i).ch3Trace);

%whole bleb values
    Bleb(i).meanInt=mean(Bleb(i).totTrace); Bleb(i).stdInt=std(Bleb(i).totTrace);
    Bleb(i).totInt=sum(Bleb(i).totTrace);
    %Bleb(i).AN=mean(Bleb(i).anTrace); Bleb(i).ANstd=std(Bleb(i).anTrace);
    Bleb(i).GP=mean(Bleb(i).gpTrace); Bleb(i).GPstd=std(Bleb(i).gpTrace);

%rotational dependence of the measured values
    CircTOT(:,j) = interp1([1:length(Bleb(i).totTrace)],Bleb(i).totTrace,...
        linspace(1, length(Bleb(i).totTrace), 360),'spline');
    %CircCH3(:,j) = interp1([1:length(Bleb(i).ch3Trace)],Bleb(i).ch3Trace,...
        %linspace(1, length(Bleb(i).ch3Trace), 360),'spline');
    %CircAN(:,j) = interp1([1:length(Bleb(i).anTrace)],Bleb(i).anTrace,...
        %linspace(1, length(Bleb(i).anTrace), 360),'spline');
    CircGP(:,j) = interp1([1:length(Bleb(i).gpTrace)],Bleb(i).gpTrace,...
        linspace(1, length(Bleb(i).gpTrace), 360),'spline');
    j=j+1;

    clear CC CCg
    end
end
CircTOT=CircTOT./nanmean(CircTOT,1); %CircCH3=CircCH3./nanmean(CircCH3,1);
%CircAN=CircAN./nanmean(CircAN,1);
CircGP=CircGP./nanmean(CircGP,1);
```

At the end of each section a .mat file is saved at the indicated path. These are named: *bleb_fullimages.mat*, *bleb_ROIs.mat*, *blebs_identified.mat*, and *blebs_analysed_final.mat*.

2.2 Specific function: OpenUpBleb.m

This function opens up a circular intensity profile into a rectangular carpet based on a published code (link). The function will also work for elliptical intensity profiles. Nevertheless, for an elliptical object the line-cuts will no longer be exactly perpendicular to the intensity profile but slightly under an angle.

```
19      function [imRcorr, ind, R] = OpenUpBlebCP(im, av, width, centers_out, radii_out, ind)
20
21      fig=0;%with ind as an input argument the function will return the data at ind
22
23      if nargin==5%with ind ABSENT as an input argument the function will perform the fitting
24          ind=[];
25          fig=1;
26      end
```

If the index numbers (*ind*) are given as an input-variable the function will only extract the intensity carpet which is width/2 above and below the peak value per column of the opened-up bleb (line 93). If not added as input a figure will open and identify how the bleb will be opened up. Next the peak intensities within the set boundaries will be determined. Be aware that there extra pixels are taken around the identified radius of the bleb (currently *width/2* extra, see line 42).

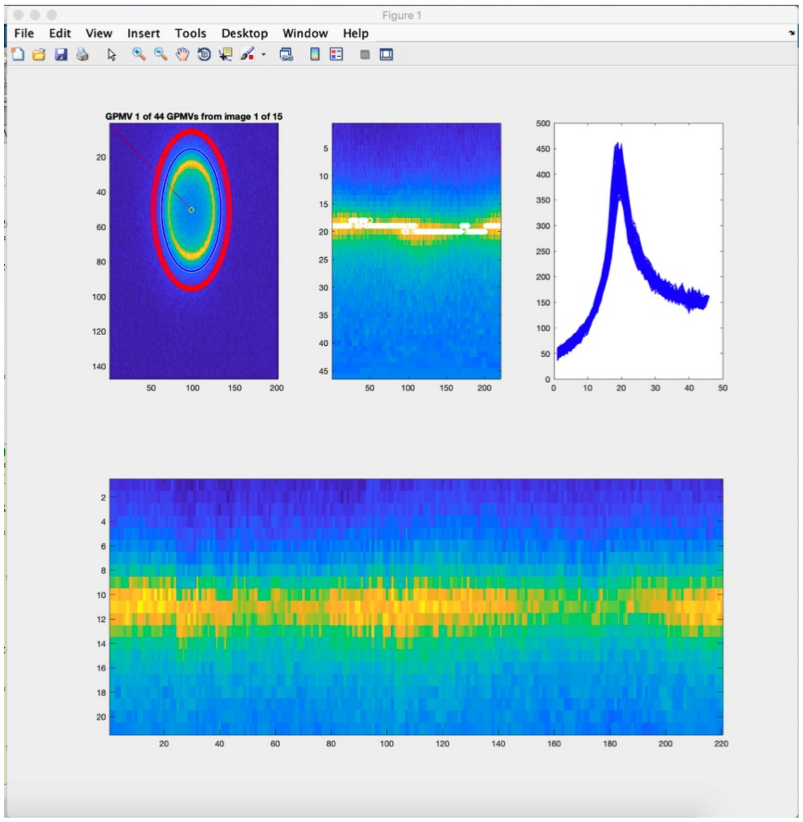
Figure 1
File Edit View Insert Tools Desktop Window Help
GPMV 1 of 44 GPMVs from image 1 of 15

Figure 2: **Opening up a bleb from circular to rectangular (top left)** Example of a GPMV with the following features identified: i) yellow circle indicates the found centre ii) blue circle shows the measured radius iii) circle of red circles shows the starting points (plus the extra pixels) of each measurement towards the center yellow circle iv) red line shows the line profile direction at 225°(starting point is vertically down). **(top middle)** Opened up GPMV along the identified positions discussed in top left. The white dots indicate the identified membrane position as determined from peak fitting. **(top right)** Column wise averaged intensity profile where the maximum is identified as the membrane position. **(bottom)** The same intensity profile carpet as top middle but now aligned with the membrane in the center of the carpet that has a width of *width*

2.3 General function: tiffread2.m

The original tiffread2 function was written by Francois Nedelec and can be downloaded from various sources. The input should contain the filepath and number of the desired first and last frame.

2.4 General function: PhotoConvertIm.m

The goal of this function is to convert the Analogue-to-Digital Units (ADU) of a camera image into photo-electrons. The current version allows automatic detection of the EMCCD camera gain and offset (requires DIPlib and PCFO) and can do image corrections for pre-calibrated sCMOS cameras (the correction images need to be available at a known location, see below).

```
15  function [image_photons, gain, offset] = PhotoConvertIm (im, cam, gain, offset)
16
17  addpath('path where tiffread2 is located')
18  cameraPath='path associated to all sCMOS calibration images';
19
20  mode=cam.mode; serialNo=cam.serialNo; ROI=cam.ROI;
```

The input arguments contain the image(:,:) or image-stack in the form of im(i,:,:) where the first dimension corresponds to the number of frames. The *cam* argument contains three part related to the camera. In the case of the cameras we are using there is an EMCCD (*cam.mode=0*) and several sCMOS cameras that can be run at three distinct modes: 12bit Sensitivity (*cam.mode=1*), 12bit Balanced (*cam.mode=2*), 12bit Full well (*cam.mode=3*), and 16bit HDR (*cam.mode=4*). To identify the camera its serial number should be added, for an EMCCD just enter: (*cam.serialNo='0000'*). The third and final part should contain the ROI of the camera that is used in the experiment for the default full field of view use: (*cam.ROI=[0 0 size(im,1) size(im,2)]*).

```
27  %% EMCCD %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28  if nargin<4 && mode==0
29      addpath('path associated to the Single Shot Gain program (PCFO)')
30      addpath(genpath('path associated to DIPlib'))
31      dip_initialise%this initialises the DIPlib software within MatLab
32          for i=1:frames
33              if frames==1
34                  temp_im(:,:)=double(im);
35              elseif frames>1
36                  temp_im(:,:)=double(im(i,:,:));
37              end
38              [gain(i), offset(i)] = pcfo(temp_im, 0.9, 0, 1, 0, [3 3]);
39          end
40  gain=nanmean(gain);
41  offset=nanmean(offset);
42  end
43
44      wb = waitbar(0,'Converting the images to photo-e');
45  if mode==0
46              for i=1:frames
47                  waitbar(i/frames,wb);
48                  if frames==1
49                      temp_im=double(im);
50                      image_photons=(double(temp_im)-offset)/gain;
51                  elseif frames>1
52                      temp_im(:,:)=im(i,:,:);
53                      image_photons(i,:,:)=(double(temp_im)-offset)/gain;
54                  end
55              end
```

To be able to use the single shot gain estimation program both the MatLab distribution of the program needs to be downloaded and the DIPlib software package (tested with 2.9) needs to be installed. Please follow the ReadMe files from the authors, also for the settings in line 38. The average gain and average offset values associated to the image-stacks will be calculated and used further down in the code (from line 46-55). If the gain and offset values are already known and given as the input arguments the gain and offset will not be separately calculated but directly used in the code (from line 46-55). Since sCMOS cameras have distinct gain-values for each pixel the image needs to be corrected using a calibrated image file, and not a single gain and offset value.

```
56  %% sCMOS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57  elseif mode==1
58      td=tiffread2([cameraPath serialNo '_Sensitivity_dark.tif']);
59      offset=td.data(ROI(2)+1:ROI(2)+ROI(4),ROI(1)+1:ROI(1)+ROI(3)); offset=double(offset);
60      tg=tiffread2([cameraPath serialNo '_Sensitivity_gain.tif']);
```

```
    gain=tg.data(ROI(2)+1:ROI(2)+ROI(4),ROI(1)+1:ROI(1)+ROI(3)); gain=double(gain)/10000;
            for i=1:frames
                waitbar(i/frames,wb);
                if frames==1
                    image_photons=(double(im)-offset)./gain;
                elseif frames>1
                    temp_im(:,:)=double(im(i,:,:));
                    image_photons(i,:,:)=(temp_im-offset)./gain;
                end
            end
gain=nanmean(nanmean(gain));
offset=nanmean(nanmean(offset));
```

The images required to correct the sCMOS camera at the specified mode can be acquired with CameraGainCalibration program. In order for the above code to function the dark and gain files should be saved as *serialNo_cam.mode_dark.tif* and *serialNo_cam.mode_gain.tif*, respectively, in the cameraPath that was identified earlier in line 18 (a filename example: *A18M203009_Sensitivity_gain.tif*). Beware that the gain image was saved in uint16 after a 10000-fold multiplication in the CameraGainCalibration program (hence the divisions at the respective positions in the code). The current function also allows conversion of only a part of the sCMOS chip as defined by the *cam.ROI*.

2.5 General function: DualCh_align.m

The goal of this function is to generate a transform matrix for aligning two images (Ch2 unto Ch1) and was adapted from the original code of Pontus Nordenfelt.

```
function [ch2_aligned, tFORM] = DualCh_align (ch1, ch2, method, vis, tFORM)
```

An important distinction is that the adapted code allows for adjusting the optimizer radius if the alignment is deemed not satisfactory by setting the fourth input, vis, at 1 (instead of 0). The alignment method can be defined as *'translation'*, *'rigid'*, *'similarity'*, or *'affine'* and if the transform matrix is provided as the last input argument the transform will be performed without checking the allignment.

2.6 General function: ImageCorrection.m

The goal of this function is to background correct an image. If the Gfactor image and/or the transform-function is given as an input the image will also be Gfactor corrected and/or aligned according to the specifications.

```
function [im_corr GFactor] = ImageCorrection (im, imBG, type, imGF, tFORM)
```

The function can handle single images or stacks in the form of im(i,:,:) where the first dimension corresponds to the number of frames. When only given 2 input arguments the single channel, single image background will be smoothed with a [9 9] median filter (line 24) before the raw images will be background subtracted. In the need of Gfactor correction both Gfactor and Background images should contain 2 channels. All 4 images will be smoothed with a median filter (line 39, 42, 46, 52, and 56) before being used. A median-filter of [9 9] was found to decrease noise-based fluctuations in the Anisotropy (or GP-value) images and still be small enough to account for irregularities in the optical path of the channels (polarisation or spectral). In the absence of a transform-function a dummy transform-function is generated and the image will not be aligned.

```
if nargin==4
    tFORM=affine2d([1 0 0; 0 1 0; 0 0 1]);
end
```

The function is able to accommodate both emission polarization based anisotropy and Laurdan General Polarisation corrections. The type variable for the input needs to be set at *'Ani'* for anisotropy based measurements and *'Ldn'* for Laurdan GP based measurments. The Gfactor image for anisotropy should be taken from a small fast tumbling (with respect to its lifetime) fluorophore solution such as FITC or Rho6G at 1µM. The Gfactor will be calculated as follows:

$$Gfactor = \frac{GFimage_{ch1}}{GFimage_{ch2}}$$

Ch1 and Ch2 are the polarisations that are parallel and perpendicular to the excitation polarisation, respectively. The Laurdan Gfactor images should come from the blue-shifted and red-shifted spectral emission bands (Ch1 and Ch2, respectively) of a 1µM solution of Laurdan in DMSO. The Gfactor will be calculated as follows:

$$gp = \frac{GFimage_{ch1} - GFimage_{ch1}}{GFimage_{ch1} + GFimage_{ch2}}$$

$$Gfactor = \frac{0.208 + 0.208 \cdot gp - gp - 1}{gp + 0.208 \cdot gp - 0.208 - 1}$$

It is important to note that the background images for the Gfactor and data are expected to be exactly the same. If this is not the case please adjust accordingly. Non-existent (NaN), infinite, and negative values in the images will be corrected to become 0.

2.7 General function: ROIselect.m

The goal of this function is to allow selections of multiple regions of interest. The regions of interest can be *objects* around which a window of size *ws* on the original image is taken or a *ws*-sized *square* region from which data can later be collected.

```
function [ROI, fin] = ROIselect(im, ws, identity, ROI)
```

The ROI input variable will get ROI.area added after a point is selected during the execution of the function. The output argument *fin* will allow the user to continue until the desired amount of ROIs have been selected using the following code in the main file:

```
figure(9)%  No 9 is specifically allocated because in the ROIselect.m function figure(9)
%           is used for ROI identification
set(gcf, 'Position',  [400 50 950 900])
imagesc(ch1+ch2);

ROI=[]; fin=0;
while fin==0
   [ROI, fin] = ROIselect(ch1+ch2, ws, ROI);
end
```

The area will be *ws/2* around the selected point and can be used to crop or extract this region from the larger image:

```
ROI(i).ch1=ch1(ROI(i).area(3):ROI(i).area(4),ROI(i).area(1):ROI(i).area(2));
ROI(i).ch2=ch2(ROI(i).area(3):ROI(i).area(4),ROI(i).area(1):ROI(i).area(2));
```

3 Troubleshooting

- My data is not being loaded or converted.

Please make sure that the filename structures are appropriately set to your own image names in the code and the folders. Make sure to check the background and Gfactor image names and paths. This also has to be made sure for the camera gain and dark images.

- The alignment of the GPMVs is not working properly.

Please change the method of alignment in line 108 (possibilities are: 'translation', 'rigid' , 'similarity' and 'affine'). Another reason for challenges in the alignment stem from intense fluorescence from places other that the object of choice. To solve or improve on that last issue you could decrease the window size.

- Circle detection is not detecting any circles in my ROI or circle detection is detecting an overload of circles in my ROI.

The are two possibilities here. 1) your starting values for circle-size estimations are not correct: Please change those respective values in line 10-16. 2) the Hough transform estimation is having trouble with the images. Please change the 'Sensitivity' and/or 'EdgeThreshold' values in line 106-107. The given values determine how well you can identify circles in your experiment and are extremely sensitive. I suggest you first play with these values in your images to find the optimal parameters. Finally, before circle detection the image is blurred to get rid of noise, check the size of the blur.