Emission Anisotropy (hFRET) Analysis

Thomas S. van Zanten

05/05/2021

1 Analysis Aims

The proximity of two like-fluorophores can be measured using *homo*-FRET or emission polarisation anisotropy. The basic premise is that here energy transfer from one fluorophore to another cannot be detected spectrally (like for *hetero*-FRET). Instead here energy transfer has to be measured by measuring the depolarisation of the excitation. Depolarisation of photo-selected fluorophores occurs due to tumbling (the dipole of the fluorophore rotates before emitting a photon) or energy transfer (the orientation dipole of the other fluorophore is distinct). The aim of this analysis software is to obtain align and correct both the Parallel and Perpendicular polarisation images obtained from an emission anisotropy or *homo*-FRET experiment. Quantitative analysis of the data can be extracted by collecting the photons from squared Regions Of Interest (ROIs, typically 10x10 or 20x20 pixels) which are subsequently plotted in a Photon Count versus Anisotropy plot. The corrected images themselves will be converted in *Total Intensity* and *Anisotropy* maps. Due to a high error associated with low photon count per pixel the raw images will also be binned in order to provide a low-error *Anisotropy* map.

2 Program Design

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

- main_anisotropy.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_anisotropy.m

2.1 Main code: main_anisotropy.m

The following code allows the user to generate anisotropy (or FRET) maps as well as quantify the measurements from ROIs. Proper imaging is imperative for correct data so please ensure that all parameters are covered. To be able to compare data from different dates, labs and people please ensure to align multiple cameras properly (if using more than one), to include excitation conditions (polarisation extinction coefficient and actual laser power) and detection conditions (polarisation extinction coefficient, filters used, and camera integration time). Also ensure to have taken a proper Gfactor image using the same conditions as how the data was collected. A Gfactor image allows for the (local) correction of any system related reduction in detection of one of the two polarization channels. 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 fast rotation ensures that even if only one orientation is photo-selected during excitation the emission is completely depolarized. In other words both detectors should receive equal amount of photons.

The main code is divided in 5 main sections:

- Section 0: Initialisation
- Section 1: Variable loading, image loading, image conversion, image alignment and background/GFactor correction
- Section 2: Indicating the ROIs for quantitative analysis, binning the images and creating the maps
- Section 3: Extracing relevant information from ROIs

- Section 4: Data representation

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).

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

The second section sets the variables (line 10-12), camera details (line 19 & 20), the filename series (line 14 & 16) and the places for background and Gfactor (line 24-25 & 30-31). 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. In line 27 the background of the PA channel is already converted to photons because its needed for background correcting that particular channel in line 38 (see below). The other images of background and GFactor are converted while being assigned to the dual channel matrices.

```
9     %% Section 1: Variable and image loading and conversion %%%%%%%%%%%%%%%%%%%
10    ws=10;%width of the measurement ROIs
11    bin=3;%data binning pixel-anisotropy determination (should be an odd number)
12    blur=3;%anisotropy image Gaussian blur for visualisation purposes
13
14    temp_PA=tiffread2b([pathname 'PA.tif'],1,40);
15    for i=1:length(temp_PA), PA(i,:,:)=temp_PA(i).data; end
16    temp_PE=tiffread2b([pathname 'PE.tif'],1,40);
17    for i=1:length(temp_PE), PE(i,:,:)=temp_PE(i).data; end
18
19    cam.mode=0; cam.serialNo='0000'; cam.tInt=2000;%camera settings for more info
20    cam.ROI=[0 0 size(PA,2) size(PA,3)];%see PhotoConvertIm.m
21    [PA, gain_PA, offset_PA] = PhotoConvertIm (PA, cam);
22    [PE, gain_PE, offset_PE] = PhotoConvertIm (PE, cam);
23
24    PA_BG=tiffread2([pathname 'BG/BG_PA.tif']);PA_BG=double(PA_BG.data);
25    PE_BG=tiffread2([pathname 'BG/BG_PE.tif']);PE_BG=double(PE_BG.data);
26    %BE AWARE TO REMOVE gain and offset IF CAMERA IS NOT EMCCD
27    PA_BG = PhotoConvertIm (PA_BG, cam, gain_PA, offset_PA);
28    BG(1,:,:) = PA_BG;
29    BG(2,:,:) = PhotoConvertIm (PE_BG, cam, gain_PE, offset_PE);
30    PA_GF=tiffread2([pathname 'BG/GF_PA.tif']);PA_GF=double(PA_GF.data);
31    PE_GF=tiffread2([pathname 'BG/GF_PE.tif']);PE_GF=double(PE_GF.data);
32    %BE AWARE TO REMOVE gain and offset IF CAMERA IS NOT EMCCD
33    GF(1,:,:) = PhotoConvertIm (PA_GF, cam, gain_PA, offset_PA);
34    GF(2,:,:) = PhotoConvertIm (PE_GF, cam, gain_PE, offset_PE);
```

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. There are several ways to run the DualCh_align and ImageCorrection functions together. In the below example the alignment is performed one-by-one on both the GFactor-image and the Data-image in the ImageCorrection function. One could also determine an average transform function and convert the entire Data-image stack outside of the loop (line 36-42) using PE as input and output in the ImageCorrection function.

```
36    for i=1:size(PE,1)
37        tim1(:,:)=PA(i,:,:);tim2(:,:)=PE(i,:,:);
38        PA(i,:,:)=ImageCorrection(tim1,PA_BG);
39        [timcorr tFORM(i)] = DualCh_align (tim1, tim2, 'affine', 1);
40        [PE(i,:,:) GFactor]=ImageCorrection(tim2, BG, 'Ani', GF, tFORM(i));
41        clear tim1 tim2 timcorr
42    end
```

Section 2 allows the user to pinpoint the squared regions (with width *ws*) in each image (j) of the image-stack. In each image first the user should identify the background in an image by selecting a portion of the image that does not contain signal. Next each cell should be identified. The selection can be stopped by double-clicking on an empty.

```
%% Section 2: Indicating the square ROIs on cells to be analysed per image
c=1; ROI=[];
for j=1:size(PE,1)

    chPA(:,:)=PA(j,:,:);
    chPE(:,:)=PE(j,:,:);

    No=size(ROI,2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(9)
    set(gcf, 'Position',  [400 50 950 900])
    imagesc(chPA+2*chPE);
        title(['Image ' num2str(j) ' of ' num2str(size(PE,1)) ' images: PLEASE DRAW BACKGROUND'])

imroi = imfreehand%%%Draw a single region that will become the image-associated background
setColor(imroi,'black');
BW=createMask(imroi);
BG(1)=sum(sum(BW.*chPA))/sum(sum(BW));
BG(2)=sum(sum(BW.*chPE))/sum(sum(BW));
cellBW=zeros(size(chPA));
title(['Image ' num2str(j) ' of ' num2str(size(PE,1))...
' images: PLEASE DRAW THE CELLS (double-click in empty space when done)'])
    while sum(sum(BW))>100
        imroi = imfreehand%%%Draw the cells in the image
        setColor(imroi,'green');
        BW=createMask(imroi);
        cellBW=cellBW + c*BW; c=c+1;
    end
c=c-1; delete(imroi);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

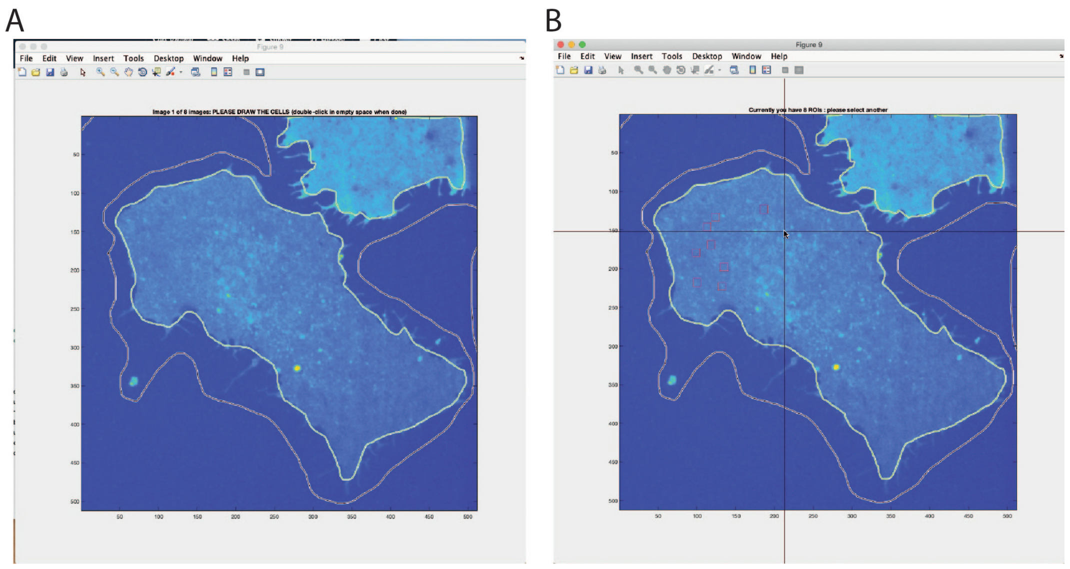

Image 1 of 8 images: PLEASE DRAW THE CELLS (double-click in empty space when done)

Currently you have 8 ROIs : please select another

Figure 1: **Selection of background, cells and ROIs.** **A** Selection of the background (black region) and the individual cells (green regions). **B** Alter the regions have been indicated the squared ROIs (red) can be placed at the places of interest. Ideally the regions for quantification should be intensity-flat in nature.

On confirming *Do you want to select another single ROI* the user can continue to select ROIs. The user can use the space bar to confirm and select the region when the cursor in on the correct spot. Once *NO* is selected the next image will be offered until the stack has been worked through. At each image the data from the indicated ROIs will be processed and the images as well. The entire ROI of any indicated cells will get a unique number associated to its pixels and will be saved as an image (line 87). To increase the pixel-wise accuracy of the Anisotropy map the data from the Parallel and Perpendicular data images will be binned.

```
79          fin=0;
80          while fin==0
81             [ROI, fin] = ROIselect(chPA+chPE, ws, 'square', ROI);
82          end
83
84          for i=No+1:size(ROI,2)
85              ROI(i).BG=BG;
86              ROI(i).frame=j;
87              ROI(i).cell=cellBW(ROI(i).area(3)+ws/2,ROI(i).area(1)+ws/2);
88              ROI(i).PA=chPA(ROI(i).area(3):ROI(i).area(4),ROI(i).area(1):ROI(i).area(2));
89              ROI(i).PE=chPE(ROI(i).area(3):ROI(i).area(4),ROI(i).area(1):ROI(i).area(2));
90          end
91
92          im(j).cells=cellBW;
93          im(j).tot=chPA+2*chPE;
94          im(j).an=(chPA-chPE)./im(j).tot;
95      %% binning the images%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
96          fun = @(x) sum(x(:));
97          chPA = nlfilter(chPA,[bin bin],fun);
98              chPA=chPA(1+((bin-1)/2):bin:end-((bin-1)/2),1+((bin-1)/2):bin:end-((bin-1)/2));
99          chPE = nlfilter(chPE,[bin bin],fun);
100             chPE=chPE(1+((bin-1)/2):bin:end-((bin-1)/2),1+((bin-1)/2):bin:end-((bin-1)/2));
101     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
102         im(j).BinTot=chPA+2*chPE;
103         im(j).BinAn=(chPA-chPE)./im(j).BinTot;
```

Section 3 analyses the data from all the collected ROIs in terms of number of photons, anisotropy and also the theoretical error in the anisotropy calculation.

```
112     %% Section 3: Analyzing all the ROIs and extracting relevant data %%%%%%%%
113     j=1;
114     for i=1:length(ROI)
115         if length(ROI(i).PA)>0
116         ROI(i).totInt=sum(sum(ROI(i).PA))+2*sum(sum(ROI(i).PE));
117         ROI(i).stdInt=std2(ROI(i).PA+2*ROI(i).PE);
118         ROI(i).meanInt=mean2(ROI(i).PA+2*ROI(i).PE);
119         ROI(i).CountRate=(1000*ROI(i).meanInt)/cam.tInt;
120         ROI(i).AN=(sum(sum(ROI(i).PA))-sum(sum(ROI(i).PE)))/ROI(i).totInt;
121     %error in anisotropy from photon statistics error propagation: Lidke et al 2005
122         ROI(i).Err=((1-ROI(i).AN)*(1+2*ROI(i).AN)*(1-ROI(i).AN+mean(mean(GFactor))*...
123         (1+2*ROI(i).AN)))/(3*ROI(i).totInt);
124         IntAn(i,:)=[ROI(i).meanInt ROI(i).AN ROI(i).Err/ROI(i).AN];
125         end
126     end
```

At the end of each of the above sections a .mat file is saved at the indicated path. These are named: *corrected_images.mat*, *images_ROIs.mat* (only the binned images and ROIs are saved in this file), and *Analysed_final.mat*. The last section (Section 4) allows the representation of the maps of one of the images in the stack as well as all the data from the collected ROIs.

```
135     %% Section 4: Data representation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136     i=2; figure(i)
137     set(gcf, 'Position', [0 50 1500 1000]);
```

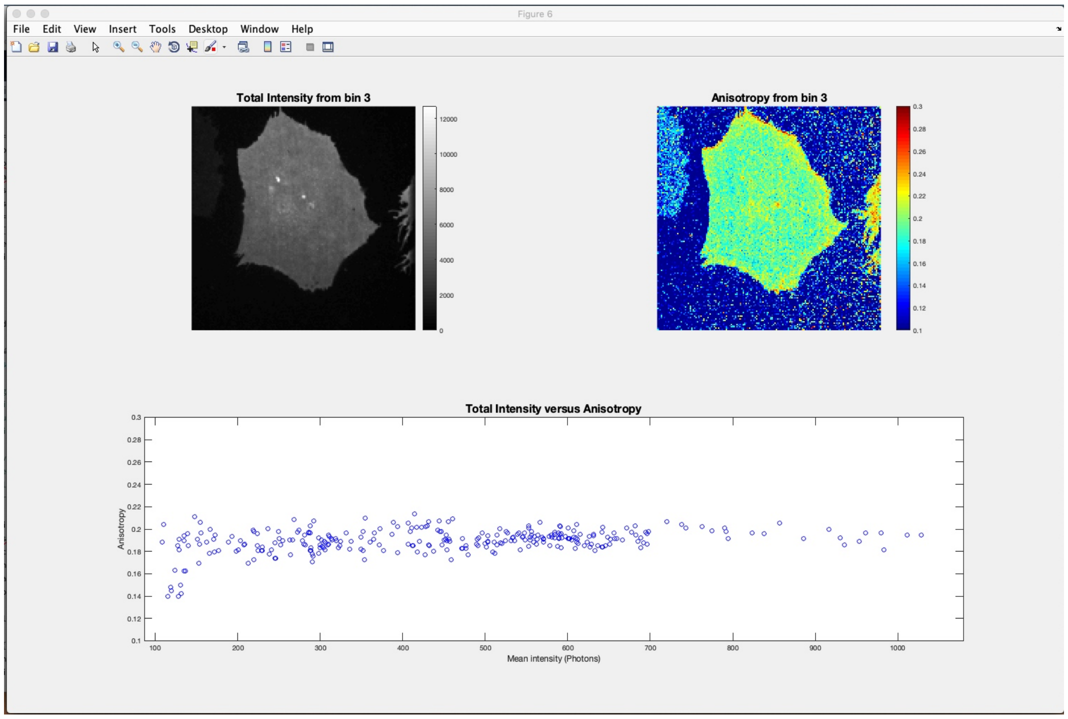
Figure 6
File Edit View Insert Tools Desktop Window Help
Total Intensity from bin 3
Anisotropy from bin 3
Total Intensity versus Anisotropy
Anisotropy
Mean intensity (Photons)

Figure 2: **Final representation of the data. top left** Total Intensity image where the grayscale is directly associated to the number of photons. **top right** Anisotropy map. **bottom** All the ROI data that was collected plotted as mean intensity per ROI versus the calculated anisotropy.

2.2 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.

2.3 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).

```
function [image_photons, gain, offset] = PhotoConvertIm (im, cam, gain, offset)

addpath('path where tiffread2 is located')
cameraPath='path associated to all sCMOS calibration images';

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)]*).

```
%% EMCCD %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<4 && mode==0
    addpath('path associated to the Single Shot Gain program (PCFO)')
    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']);
61     gain=tg.data(ROI(2)+1:ROI(2)+ROI(4),ROI(1)+1:ROI(1)+ROI(3)); gain=double(gain)/10000;
62             for i=1:frames
63                 waitbar(i/frames,wb);
64                 if frames==1
65                     image_photons=(double(im)-offset)./gain;
66                 elseif frames>1
67                     temp_im(:,:)=double(im(i,:,:));
68                     image_photons(i,:,:)=(temp_im-offset)./gain;
69                 end
70             end
71 gain=nanmean(nanmean(gain));
72 offset=nanmean(nanmean(offset));
```

The images required to correct the sCMOS camera at the specified mode can be acquired with CameraGain-Calibration 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*.

2.4 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.

```
19 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.

2.5 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.

```
15    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.

```
35    if nargin==4
36        tFORM=affine2d([1 0 0; 0 1 0; 0 0 1]);
37    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.

2.6 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.

```
10    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));
```