Electronic Supplementary Material (ESI) for Physical Chemistry Chemical Physics.

Anisotropy analysis: the MATLAB code

```
function faia(varargin)
%==========================================================================
%FAIA   fluorescence anisotropy image analysis
%   Gui-based. Written for rot-Bodipy project (II 2014).
%   FAIA(II_channel, L_channel)  Parameters (ascii images exported from
%   PicoQuant SymphoTime software) are optional; no warranties whatsoever.
%   For saving images with color bars export_fig needs to be installed.
%   piotr.jurkiewicz@jh-inst.cas.cz
%   (license included at the end of the source file)
%==========================================================================

% other functions included in this file:
% loadSPTascii, angularDependence, sem, roundSP, bin, cmap2rgb

version='0.11';

%   Naming scheme:
%   "I" stands for parallel and "L" for peprendicular channels
%--------------------------------------------------------------------------

%- Initialisation ---------------------------------------------------------

% global variables (names are awkward - I know. But I wanted them short):
IFileName=''; LFileName='';                                             %
file names for I and L channels
currentPath='';                                                         %
path from which last file was loaded
configFileName='faia.cfg';                                              %
for saving parameters
IrealSize=[1,1]; LrealSize=[1,1];                                       %
image sizes in real units
IrealSizeUnits='mm'; LrealSizeUnits='mm';                               %
and names of those units
iIS=[100,100];                                                          %
initial image size [x,y]
Iint=zeros(iIS(2),iIS(1)); Lint=zeros(iIS(2),iIS(1));                   %
intensity images
IintB=Iint;   LintB=Lint;                                               %
binned
IintBt=IintB; LintBt=LintB;                                             %
and thresholded
Imlt=NaN; Lmlt=NaN;                                                     %
mean lifetime images (might not be available)
ImltB=Imlt;   LmltB=Lmlt;                                               %
binned
ImltBt=ImltB; LmltBt=LmltB;                                             %
and and lifetime tunned
tmpint=IintBt; tmpmlt=ImltBt;                                           %
additional variables used in redrawH
A=zeros(iIS(2),iIS(1));                                                 %
anisotropy image
HA=0; HIint=0; HLint=0; HImlt=1; HLmlt=1; HImltw=1; HLmltw=1;           %
angular dependences (anisotropy, intensities, lifetimes, intensity-weighted
lifetimes)

Ihsv=ImltBt-ImltBt; Ihsv(:,:,2)=1; Ihsv(:,:,3)=0;                       %
matrix for I lifetime hsv image to display
Lhsv=LmltBt-LmltBt; Lhsv(:,:,2)=1; Lhsv(:,:,3)=0;                       %
matrix for L lifetime hsv image to display

IintBmin=min(min(IintB)); LintBmin=min(min(LintB));                     %
minima of binned intesity images
```

```
IintBmax=max(max(IintB)); LintBmax=max(max(LintB));                         %
maxima of binned intesity images
ImltBmin=min(min(ImltB)); LmltBmin=min(min(LmltB));                         %
minima of binned lifetime images
ImltBmax=max(max(ImltB)); LmltBmax=max(max(LmltB));                         %
minima of binned lifetime images

IintBtmin=min(min(IintBt)); LintBtmin=min(min(LintBt));                     %
minima of binned intesity images
IintBtmax=max(max(IintBt)); LintBtmax=max(max(LintBt));                     %
maxima of binned intesity images
ImltBtmin=min(min(ImltBt)); LmltBtmin=min(min(LmltBt));                     %
minima of binned lifetime images
ImltBtmax=max(max(ImltBt)); LmltBtmax=max(max(LmltBt));                     %
minima of binned lifetime images

IintBSize=size(IintB); LintBSize=size(LintB); ASize=size(A);                %
image sizes
hIImage=0; hIaxis=0; hLImage=0; hLaxis=0; hAImage=0; hAaxis=0; hHaxes=0;    %
handles of images, their axes and histogram axes

AScale=[-0.2,0.5];                                                          %
min and max anisotropy values for display
mltScale=[3.5,6];                                                           %
min and max lifetime values for display
mask=1;                                                                     %
mask for images (used to get read of unwanted image parts)
dispMask=mask;                                                              %
mask for display (it has a cross at the ceter)
shade=0.2;                                                                  %
for displaying a mask 1-nothing 0-fully transparent

G=1.0;                                                                      %
G-factor for anisotropy calculation, it was 0.925 for our data
LmltShift=0;                                                                %
shift in lifetime data (usually detectors are shifted and Symphotime does not
care!~)
binSize=[1,1];                                                              %
size of bin for image binning (binSize x binSize) (pixel)
histBinWidth=10;                                                            %
width of bin for histogramming angular dependencies (deg)
cmapName='jet';                                                             %
name of matlab colormap for display intensity images
cmap=0;                                                                     %
matlab colormap for display intensity images
grayCmapImage=0;                                                            %
gray image for intensity colorbar for flim display
flimCmapImage=0;                                                            %
hue scale for flim colorbar display

background=[0.33,0.33,0.33];                                                %
main window background color
IColor=[0,0.47,0.94];                                                       %
color for I curve
LColor=[0,0.85,0];                                                          %
color for L curve

set(0,'DefaultFigureMenu','none','DefaultAxesDrawmode',
'fast','DefaultAxesFontSize',8,'DefaultAxesColor',[shade,shade,shade]);    %
defaults
%--------------------------------------------------------------------------

%- Processing the agruments: ----------------------------------------------
if nargin>0
    if nargin>2, error('MATLAB:faia:InvalidArgument','Wrong function call. Too
many arguments'); end
```

```
    if ~ischar(varargin{1}), error('MATLAB:faia:InvalidArgument','Wrong
function call. 1st argument must be a string.');
    else IFileName=varargin{1}; end
    if nargin==2,
        if ~ischar(varargin{2}), error('MATLAB:faia:InvalidArgument','Wrong
function call. 2nd argument must be a string.');
        else, LFileName=varargin{2}; end
    end
end
%--------------------------------------------------------------------------

%- Building GUI -----------------------------------------------------------

% creating main figure window and setting colormaps
x=0; y=0; dx=0;
%needed temporarily, cleared once GUI built
hMainFigure=figure('Position',[500 50 800 800],'Name',['FAIA - Fluorescence
Anisotropy Image Analysis
v',version],'NumberTitle','off','Resize','off','Color',background,'CloseReques
tFcn',@closingFaiaCallback);
grayCmapImage=cmap2rgb(colormap(gray(256))); flimCmapImage=ones(256,1,3);
flimCmapImage(:,1,1)=(255:-1:0)'/255*0.7;
flimCmapImage=hsv2rgb(flimCmapImage); cmap=colormap([cmapName,'(256)']);

%- Loading config ---------------------------------------------------------
if exist(configFileName,'file'), load(configFileName,'-mat'); cd(currentPath);
end
%--------------------------------------------------------------------------

colormap(cmap);

% adding subplots with colorbars:
subplot(2,2,1); hIImage=imagesc(Iint); hIaxis=gca;
set(hIaxis,'Xlim',[0.5, IintBSize(2)+0.5],'Ylim',[0.5,
IintBSize(2)+0.5],'XTick',[1,IintBSize(2)/2,IintBSize(2)],'YTick',[1,IintBSize
(1)/2,IintBSize(1)],'YDir','normal','TickDir','out');
dx=0.005;
hIImageCBintAxis=axes('Position',[0.466+dx,0.77,0.014,0.15]);
hIImageCBint=image(cmap2rgb(cmap)); ylabel('Int','Position',[2.3 126
1],'FontSize',9);   % homemade colorbar (the original one sucks)
set(hIImageCBintAxis,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick'
,[],'YDir','normal','YTick',[0.5,256],'YTickLabel',['min';'max'],'YMinorTick',
'on');
hIImageCBmltAxis=axes('Position',[0.466+dx,0.59,0.014,0.15]);
hIImageCBmlt=image(flimCmapImage,'Visible','off'); ylabel('\tau
(ns)','Position',[2.3 126 1],'FontSize',9); % homemade colorbar
set(hIImageCBmltAxis,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick'
,[],'YDir','normal','YTick',[0.5,256],'YTickLabel',['min';'max'],'YMinorTick',
'on','Visible','off');
subplot(2,2,2); hLImage=imagesc(Lint); hLaxis=gca;
set(hLaxis,'Xlim',[0.5, LintBSize(2)+0.5],'Ylim',[0.5,
LintBSize(2)+0.5],'XTick',[1,LintBSize(2)/2,LintBSize(2)],'YTick',[1,LintBSize
(1)/2,LintBSize(1)],'YDir','normal','TickDir','out');
dx=dx;
hLImageCBintAxis=axes('Position',[0.906+dx,0.77,0.014,0.15]);
hLImageCBint=image(cmap2rgb(cmap)); ylabel('Int','Position',[2.3 126
1],'FontSize',9);   % homemade colorbar (the original one sucks)
set(hLImageCBintAxis,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick'
,[],'YDir','normal','YTick',[0.5,256],'YTickLabel',['min';'max'],'YMinorTick',
'on');
hLImageCBmltAxis=axes('Position',[0.906+dx,0.59,0.014,0.15]);
hLImageCBmlt=image(flimCmapImage,'Visible','off'); ylabel('\tau
(ns)','Position',[2.3 126 1],'FontSize',9); % homemade colorbar
set(hLImageCBmltAxis,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick'
,[],'YDir','normal','YTick',[0.5,256],'YTickLabel',['min';'max'],'YMinorTick',
'on','Visible','off');
```

```
subplot(2,2,3); hAaxis=gca;
set(hAaxis,'Color',[0.90,0.90,0.90],'XTick',[],'YTick',[],'Box','on');
hAImage=imagesc(A,'AlphaData', dispMask);
set(hAaxis,'TickLength',[0,0],'XTick',[],'YTick',[],'Color',[0,0,0]);
errorbarFunction=@(x,y)errorbar(x,y,ones(1,37).*0.1);
hHaxis=subplot(2,2,4); [hHaxes,hHAplot,hHIplot]=plotyy(-
180:10:180,zeros(1,37),-180:10:180,ones(1,37),errorbarFunction);
set(hHaxes(1),'YColor',[0,0,0],'FontSize',9,'Color',[0.83,0.83,0.83],'XAxisLoc
ation','top','XLim',[-200,200],'XTick',-
180:90:180,'YLim',AScale,'YTickMode','auto','Box','off','TickDir','out','XGrid
','on');
set(hHAplot,'Color',[0,0,0]);
set(hHIplot,'Color',IColor); hold(hHaxes(2),'on');
hHLplot=errorbar(-
180:10:180,ones(1,37),ones(1,37).*0.1,'Color',LColor,'Parent',hHaxes(2));
hold(hHaxes(2),'off');
set(hHaxes(2),'YColor',[0.65,0.85,0.75],'FontSize',9,'XTickLabel','','TickDir'
,'out','XLim',[-200,200],'XTick',-
180:90:180,'YLim',mltScale,'YTickMode','auto');
ylabel(hHaxes(1),'R'); ylabel(hHaxes(2),'\tau  (ns)');

% adding uicontrols:
hILabel    =uicontrol('Style','text','Position',[105,765,270,20],'String','II
-
parallel','BackgroundColor',background,'FontSize',11,'ForegroundColor',IColor)
;
hLLabel    =uicontrol('Style','text','Position',[520,765,140,20],'String','L
-
perpendicular','BackgroundColor',background,'FontSize',11,'ForegroundColor',LC
olor);
hIFileLabel=uicontrol('Style','text','Position',[110,745,210,15],'String',IFil
eName,'BackgroundColor',background,'HorizontalAlignment','left','FontSize',9);
hLFileLabel=uicontrol('Style','text','Position',[462,745,210,15],'String',LFil
eName,'BackgroundColor',background,'HorizontalAlignment','left','FontSize',9);
hILoadButton=uicontrol('Style','pushbutton','Position',[325,745,50,20],'Callba
ck',@ILoadButtonCallback,'String','Load');
hLLoadButton=uicontrol('Style','pushbutton','Position',[676,745,50,20],'Callba
ck',@LLoadButtonCallback,'String','Load');
x=20; y=467;
hGLabel=
uicontrol('Style','text','Position',[x,y+151,50,17],'String','G-
factor:','BackgroundColor',background,'HorizontalAlignment','left');
hLmltShiftLabel=uicontrol('Style','text','Position',[x,y+97,50,30],'String',['
L lifetime',char(10),'shift
(ns):'],'BackgroundColor',background,'HorizontalAlignment','left');
hBinLabel=
uicontrol('Style','text','Position',[x,y+56,50,17],'String','Binning:','Backgr
oundColor',background,'HorizontalAlignment','left');
hColormapLabel=
uicontrol('Style','text','Position',[x,y+16,60,17],'String','Color
map:','BackgroundColor',background,'HorizontalAlignment','left');
hGEdit=
uicontrol('Style','edit','Position',[x,y+135,43,17],'Callback',@GEditCallback,
'String',G);
hLmltShiftEdit=uicontrol('Style','edit','Position',[x,y+80,46,17],'Callback',@
LmltShiftEditCallback,'String',LmltShift);
hBinEdit=
uicontrol('Style','edit','Position',[x,y+40,37,17],'Callback',@BinEditCallback
,'String',binSize(1));
hColormapEdit=uicontrol('Style','edit','Position',[x,y,50,17],'Callback',@Colo
rmapEditCallback,'String',cmapName);
x=103; y=430;
hIintLabel  =uicontrol('Style','text','Position',[x+1,y-
1,15,17],'String','Int','HorizontalAlignment','left','BackgroundColor',backgro
und,'FontSize',9);
```

```
hIintSlider
=uicontrol('Style','slider','Position',[x+20,y,172,17],'Min',IintBmin,'Max',Ii
ntBmax+1,'SliderStep',[0.01,0.05],'Callback',@IintSliderCallback);
hIintMinEdit=uicontrol('Style','edit','Position',[x+195,y,36,17],'Callback',@I
intMinEditCallback,'String','min','Enable','on');
hIintMaxEdit=uicontrol('Style','edit','Position',[x+234,y,36,17],'Callback',@I
intMaxEditCallback,'String','max','Enable','on');
x=455;
hLintLabel  =uicontrol('Style','text','Position',[x+1,y-
1,15,17],'String','Int','HorizontalAlignment','left','BackgroundColor',backgro
und,'FontSize',9);
hLintSlider
=uicontrol('Style','slider','Position',[x+20,y,172,17],'Min',LintBmin,'Max',Li
ntBmax+1,'SliderStep',[0.01,0.05],'Callback',@LintSliderCallback);
hLintMinEdit=uicontrol('Style','edit','Position',[x+195,y,36,17],'Callback',@L
intMinEditCallback,'String','min','Enable','on');
hLintMaxEdit=uicontrol('Style','edit','Position',[x+234,y,36,17],'Callback',@L
intMaxEditCallback,'String','max','Enable','on');
x=103; y=406;
hImaskRadio
=uicontrol('Style','radio','Position',[x,y,80,17],'Callback',@ImaskRadioCallba
ck,'String','show mask','BackgroundColor',background,'Value',0);
hIflimRadio
=uicontrol('Style','radio','Position',[x+80,y,45,17],'Callback',@IflimRadioCal
lback,'String','FLIM','BackgroundColor',background,'Value',0,'Enable','off');
hIflimMinEdit=uicontrol('Style','edit','Position',[x+127,y,30,17],'Callback',@
IflimMinEditCallback,'String','min','Enable','off');
hIflimMaxEdit=uicontrol('Style','edit','Position',[x+160,y,30,17],'Callback',@
IflimMaxEditCallback,'String','max','Enable','off');
hIflimLabel
=uicontrol('Style','text','Position',[x+193,y,14,15],'String','ns','Horizontal
Alignment','left','BackgroundColor',background,'Enable','off');
hISaveButton=uicontrol('Style','pushbutton','Position',[x+221,y-
2,50,20],'Callback',@ISaveButtonCallback,'String','Save');
x=455;
hLmaskRadio=
uicontrol('Style','radio','Position',[x,y,80,17],'Callback',@LmaskRadioCallbac
k,'String','show mask','BackgroundColor',background,'Value',0);
hLflimRadio=
uicontrol('Style','radio','Position',[x+80,y,45,17],'Callback',@LflimRadioCall
back,'String','FLIM','BackgroundColor',background,'Value',0,'Enable','off');
hLflimMinEdit=uicontrol('Style','edit','Position',[x+127,y,30,17],'Callback',@
LflimMinEditCallback,'String','min','Enable','off');
hLflimMaxEdit=uicontrol('Style','edit','Position',[x+160,y,30,17],'Callback',@
LflimMaxEditCallback,'String','max','Enable','off');
hLflimLabel=
uicontrol('Style','text','Position',[x+193,y,14,15],'String','ns','HorizontalA
lignment','left','BackgroundColor',background,'Enable','off');
hLSaveButton=uicontrol('Style','pushbutton','Position',[x+221,y-
2,50,20],'Callback',@LSaveButtonCallback,'String','Save');
hALabel   =uicontrol('Style','text','Position',[105,378,270,20],'String','R  -
anisotropy','BackgroundColor',background,'FontSize',11);
hA0Label
=uicontrol('Style','text','Position',[226,365,30,11],'String',['0',char(186)],
'BackgroundColor',background);
hAp90Label=uicontrol('Style','text','Position',[377,222,30,11],'String',['90',
char(186)],'BackgroundColor',background,'HorizontalAlignment','left');
hAm90Label=uicontrol('Style','text','Position',[71,222,30,11],'String',['-
90',char(186)],'BackgroundColor',background,'HorizontalAlignment','right');
hCyLabel   =uicontrol('Style','text','Position',[25,80,30,17],'String','C
y','BackgroundColor',background);
hCySlider=uicontrol('Style','slider','Position',[58,63,17,323],'Min',-
IintBSize(1),'Max',-1,'Value',round(-
IintBSize(1)/2),'SliderStep',[1/IintBSize(1),5/IintBSize(1)],'Callback',@CySli
derCallback);
```

```
hCyEdit
=uicontrol('Style','edit','Position',[25,63,30,17],'Callback',@CyEditCallback,
'String',round(IintBSize(1)/2));
hCxLabel   =uicontrol('Style','text','Position',[105,41,30,17],'String','C
x','BackgroundColor',background);
hCxSlider=uicontrol('Style','slider','Position',[77,63,323,17],'Min',1,'Max',I
intBSize(2),'Value',round(IintBSize(2)/2),'SliderStep',[1/IintBSize(2),5/IintB
Size(2)],'Callback',@CxSliderCallback);
hCxEdit
=uicontrol('Style','edit','Position',[77,43,30,17],'Callback',@CxEditCallback,
'String',round(IintBSize(2)/2));
diagon=ceil(sqrt(sum(IintBSize.^2)));
hrLabel
=uicontrol('Style','text','Position',[165,40,15,17],'String','r','BackgroundCo
lor',background);
hrEdit
=uicontrol('Style','edit','Position',[181,42,30,17],'Callback',@rEditCallback,
'String','0');
hrSlider=uicontrol('Style','slider','Position',[214,42,432,17],'Min',0,'Max',d
iagon,'Value',0,'SliderStep',[1/diagon,5/diagon],'Callback',@rSliderCallback);
hRLabel
=uicontrol('Style','text','Position',[165,20,15,17],'String','R','BackgroundCo
lor',background);
hREdit
=uicontrol('Style','edit','Position',[181,22,30,17],'Callback',@REditCallback,
'String',diagon);
hRSlider=uicontrol('Style','slider','Position',[214,22,432,17],'Min',0,'Max',d
iagon,'Value',ceil(diagon/2),'SliderStep',[1/diagon,5/diagon],'Callback',@RSli
derCallback);
hHAScaleMax=uicontrol('Style','edit','Position',[421,352,30,17],'Callback',@HA
ScaleMaxCallback,'String',AScale(2),'HorizontalAlignment','right');
hHAScaleMin=uicontrol('Style','edit','Position',[421,81,30,17],'Callback',@HAS
caleMinCallback,'String',AScale(1),'HorizontalAlignment','right');
hHmltScaleMax=uicontrol('Style','edit','Position',[730,352,30,17],'Callback',@
HmltScaleMaxCallback,'String',mltScale(2),'HorizontalAlignment','right');
hHmltScaleMin=uicontrol('Style','edit','Position',[730,81,30,17],'Callback',@H
mltScaleMinCallback,'String',mltScale(1),'HorizontalAlignment','right');
x=466; y=66;
hHbinLabel=uicontrol('Style','text','Position',[x,y-
2,120,17],'String',['Interval width
',char(186)],'HorizontalAlignment','left','BackgroundColor',background);
hHbinEdit=
uicontrol('Style','edit','Position',[x+68,y,36,17],'Callback',@HbinEditCallbac
k,'String',num2str(histBinWidth));
x=602; dx=29;
hHARadio=
uicontrol('Style','radio','Position',[x,y,dx,17],'Callback',@HARadioCallback,'
String','R','BackgroundColor',background,'Value',1,'Enable','off');
hHIRadio=
uicontrol('Style','radio','Position',[x+dx,y,dx,17],'Callback',@HIRadioCallbac
k,'String','II','BackgroundColor',background,'Value',0,'ForegroundColor',IColo
r,'Enable','off');
hHLRadio= uicontrol('Style','radio','Position',[x+2*dx-
2,y,dx,17],'Callback',@HLRadioCallback,'String','L','BackgroundColor',backgrou
nd,'Value',0,'ForegroundColor',LColor,'Enable','off');
hASaveRadio=uicontrol('Style','radio','Position',[666,23,32,17],'String','all'
,'BackgroundColor',background,'Value',0,'Enable','on');
hHSaveButton=uicontrol('Style','pushbutton','Position',[700,45,80,22],'Callbac
k',@HSaveButtonCallback,'String','Save Plot','Enable','on'); % tmp
hASaveButton=uicontrol('Style','pushbutton','Position',[700,20,80,22],'Callbac
k',@ASaveButtonCallback,'String','Save Image','Enable','off');
clear x y dx diagon;
%--------------------------------------------------------------------------
redrawA; redrawH; if ~isempty(IFileName) || ~isempty(LFileName), newIFile;
newLFile; end
%--------------------------------------------------------------------------
```

```
% Callback functions:

%...ILoadButtonCallback.............................................NESTED
function ILoadButtonCallback(~, ~)
    [FileName,PathName] = uigetfile({'*.dat'; '*.*'}, 'FAIA - Load immage from
parallel channel (SyphoTime ascii)','Multiselect','on');
    if isequal(FileName,0) || isequal(PathName,0), return; end
    currentPath=PathName;
    if iscell(FileName)
        IFileName=fullfile(PathName,FileName{1});
        LFileName=fullfile(PathName,FileName{2});
        newIFile; newLFile;
    else
        IFileName=fullfile(PathName,FileName); newIFile;
    end
end
%.............................................ILoadButtonCallback...

%...LLoadButtonCallback.............................................NESTED
function LLoadButtonCallback(~, ~)
    [FileName,PathName] = uigetfile({'*.dat'; '*.*'}, 'FAIA - Load image from
perpedicular channel (SyphoTime ascii)');
    if isequal(FileName,0) || isequal(PathName,0), return; end
    currentPath=PathName;
    LFileName=fullfile(PathName,FileName); newLFile;
end
%.............................................LLoadButtonCallback...

%...GEditCallback...................................................NESTED
function GEditCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if val>=0, G=val; end
    set(hObject,'String',G); redrawA; redrawH;
end
%.................................................GEditCallback...


%...LmltShiftEditCallback...........................................NESTED
function LmltShiftEditCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if isreal(val), LmltShift=val; end
    set(hObject,'String',LmltShift); newLFile;
end
%.........................................LmltShiftEditCallback...

%...BinEditCallback.................................................NESTED
function BinEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    newBinSize=[val,val];
    if  sum(rem(size(Iint),newBinSize)),
        msgbox(['Image size is not divisible by ',num2str(val)],'FAIA -
Error');
        set(hObject,'String',binSize(1));
        return;
    else binSize=round(newBinSize); end
    set(hObject,'String',binSize(1));
    binIImage;
    IintBtmin=get(hIintSlider,'Value'); % get values from slider
    IintBtmax=get(hIintSlider,'Max');
    IintBtBottom=get(hIintSlider,'Min');
    set(hIintSlider,'Value',IintBtBottom);
    IintBtmin=ceil(IintBtmin*(IintBmax-IintBmin)/(IintBtmax-
IintBtBottom)+IintBmin);
    if IintBtmin>IintBmin, else IintBtmin=IintBmin; end
    IintBtBottom=IintBmin; IintBtmax=floor(IintBmax);
    set(hIintSlider,'Max',IintBtmax); set(hIintSlider,'Value',IintBtmax);
    set(hIintSlider,'Min',IintBtBottom); set(hIintSlider,'Value',IintBtmin);
```

```
    set(hIintMinEdit,'String',IintBtmin);
set(hIintMaxEdit,'String',IintBtmax);
    thresholdIImage;
    if ImltBtmin>ImltBmin, else ImltBtmin=ImltBmin; end
    if ImltBtmax<ImltBmax, else ImltBtmax=ImltBmax; end
    set(hIflimMinEdit,'String',ImltBtmin);
    set(hIflimMaxEdit,'String',ImltBtmax);
    lifetimeTuneIImage;
    redrawI;
    binLImage;
    LintBtmin=get(hLintSlider,'Value'); % get values from slider
    LintBtmax=get(hLintSlider,'Max');
    LintBtBottom=get(hLintSlider,'Min');
    set(hLintSlider,'Value',LintBtBottom);
    LintBtmin=ceil(LintBtmin*(LintBmax-LintBmin)/(LintBtmax-
LintBtBottom)+LintBmin);
    if LintBtmin>LintBmin, else LintBtmin=LintBmin; end
    LintBtBottom=LintBmin; LintBtmax=floor(LintBmax);
    set(hLintSlider,'Max',LintBtmax); set(hLintSlider,'Value',LintBtmax);
    set(hLintSlider,'Min',LintBtBottom); set(hLintSlider,'Value',LintBtmin);
    set(hLintMinEdit,'String',LintBtmin);
set(hLintMaxEdit,'String',LintBtmax);
    thresholdLImage;
    if LmltBtmin>LmltBmin, else LmltBtmin=LmltBmin; end
    if LmltBtmax<LmltBmax, else LmltBtmax=LmltBmax; end
    set(hLflimMinEdit,'String',LmltBtmin);
    set(hLflimMaxEdit,'String',LmltBtmax);
    lifetimeTuneLImage; redrawL; redrawA; redrawH;
end
%.....................................................BinEditCallback...

%...ColormapEditCallback.............................................NESTED
function ColormapEditCallback(hObject, ~)
    val=get(hObject,'String');
    if isequal(val,'pit')
        newCmap=jet(256); newCmap(1,:)=[0,0,0.3]; colormap(newCmap);
    else
        try newCmap=colormap([val,'(256)']);catch, val=cmapName; newCmap=cmap;
end
    end
    cmap=newCmap; cmapName=val;
    if ~get(hIflimRadio,'Value'), set(hIImageCBint,'CData',cmap2rgb(cmap));
end;
    if ~get(hLflimRadio,'Value'), set(hLImageCBint,'CData',cmap2rgb(cmap));
end;
    set(hObject,'String',cmapName);
end
%.................................................ColormapEditCallback...

%...IintSliderCallback...............................................NESTED
function IintSliderCallback(hObject, ~)
    IintBtmin=round(get(hObject,'Value'));
    set(hIintMinEdit,'String',IintBtmin); set(hObject,'Value',IintBtmin);
    thresholdIImage; redrawI; redrawA; redrawH;
end
%.................................................IintSliderCallback...

%...LintSliderCallback...............................................NESTED
function LintSliderCallback(hObject, ~)
    LintBtmin=round(get(hObject,'Value'));
    set(hLintMinEdit,'String',LintBtmin); set(hObject,'Value',LintBtmin);
    thresholdLImage; redrawL; redrawA; redrawH;
end
%.................................................LintSliderCallback...

%...IintMinEditCallback..............................................NESTED
```

```
function IintMinEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val<IintBmin, IintBtmin=IintBmin;
    elseif val>IintBtmax,IintBtmin=IintBtmax; else IintBtmin=val; end
    set(hObject,'String',IintBtmin);
    set(hIintSlider,'Value',IintBtmin);
    thresholdIImage; redrawI; redrawA; redrawH;
end
%.....................................................IintMinEditCallback...

%...LintMinEditCallback.....................................................NESTED
function LintMinEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val<LintBmin, LintBtmin=LintBmin;
    elseif val>LintBtmax, LintBtmin=LintBtmax; else LintBtmin=val; end
    set(hObject,'String',LintBtmin);
    set(hLintSlider,'Value',LintBtmin);
    thresholdLImage; redrawL; redrawA; redrawH;
end
%.....................................................LintMinEditCallback...

%...IintMaxEditCallback.....................................................NESTED
function IintMaxEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val>IintBmax, IintBtmax=IintBmax;
    elseif val<IintBtmin, IintBtmax=IintBtmin; else IintBtmax=val; end
    set(hObject,'String',IintBtmax);
    set(hIintSlider,'Max',IintBtmax);
    thresholdIImage; redrawI; redrawA; redrawH;
end
%.....................................................LintMaxEditCallback...

%...LintMaxEditCallback.....................................................NESTED
function LintMaxEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val>LintBmax, LintBtmax=LintBmax;
    elseif val<LintBtmin, LintBtmax=LintBtmin; else LintBtmax=val; end
    set(hObject,'String',LintBtmax);
    set(hLintSlider,'Max',LintBtmax);
    thresholdLImage; redrawL; redrawA; redrawH;
end
%.....................................................LintMaxEditCallback...

%...ImaskRadioCallback.....................................................NESTED
function ImaskRadioCallback(hObject, ~)
    redrawI;
end
%.....................................................ImaskRadioCallback...

%...LmaskRadioCallback.....................................................NESTED
function LmaskRadioCallback(hObject, ~)
    redrawL;
end
%.....................................................LmaskRadioCallback...

%...IflimRadioCallback.....................................................NESTED
function IflimRadioCallback(hObject, ~)
    val=get(hObject,'Value');
    if val, set(hIImageCBint,'CData',grayCmapImage); else
set(hIImageCBint,'CData',cmap2rgb(cmap)); end
    redrawI;
end
%.....................................................IflimRadioCallback...

%...LflimRadioCallback.....................................................NESTED
function LflimRadioCallback(hObject, ~)
```

```
    val=get(hObject,'Value');
    if val, set(hLImageCBint,'CData',grayCmapImage); else
set(hLImageCBint,'CData',cmap2rgb(cmap)); end
    redrawL;
end
%.....................................................LflimRadioCallback...

%...IflimMinEditCallback.....................................................NESTED
function IflimMinEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val<ImltBmin, ImltBtmin=ImltBmin;
    elseif val>ImltBtmax, ImltBtmin=ImltBtmax; else ImltBtmin=val; end
    set(hObject,'String',ImltBtmin);
    lifetimeTuneIImage;
    redrawI; redrawA; redrawH;
end
%.................................................IflimMinEditCallback...

%...LflimMinEditCallback.....................................................NESTED
function LflimMinEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val<LmltBmin, LmltBtmin=LmltBmin;
    elseif val>LmltBtmax, LmltBtmin=LmltBtmax; else LmltBtmin=val; end
    set(hObject,'String',LmltBtmin);
    lifetimeTuneLImage;
    redrawL; redrawA; redrawH;
end
%.................................................LflimMinEditCallback...

%...IflimMaxEditCallback.....................................................NESTED
function IflimMaxEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val>ImltBmax, ImltBtmax=ImltBmax;
    elseif val<ImltBtmin, ImltBtmax=ImltBtmin; else ImltBtmax=val; end
    set(hObject,'String',ImltBtmax);
    lifetimeTuneIImage;
    redrawI; redrawA; redrawH;
end
%.................................................IflimMaxEditCallback...

%...LflimMaxEditCallback.....................................................NESTED
function LflimMaxEditCallback(hObject, ~)
    val=str2double(get(hObject,'String'));
    if isnan(val), elseif val>LmltBmax, LmltBtmax=LmltBmax;
    elseif val<LmltBtmin, LmltBtmax=LmltBtmin; else LmltBtmax=val; end
    set(hObject,'String',LmltBtmax);
    lifetimeTuneLImage;
    redrawL; redrawA; redrawH;
end
%.................................................LflimMaxEditCallback...

%...ISaveButtonCallback.....................................................NESTED
function ISaveButtonCallback(hObject, ~)
    if isequal(get(hIflimRadio,'Enable'),'on')
        [FileName,PathName,format]=uiputfile(...
        {'*.dat','ASCII (*.dat) - intensity only'; '*.dat','ASCII (*.dat) -
intensity only masked'; '*.dat','ASCII (*.dat) - lifetimes only';
'*.dat','ASCII (*.dat) - lifetimes only masked';
        '*.bmp','Bitmap (*.bmp) - intensity false-color'; '*.bmp','Bitmap
(*.bmp) - intensity false-color masked'; '*.bmp','Bitmap (*.bmp) - FLIM';
'*.bmp','Bitmap (*.bmp) - FLIM masked'},'FAIA - Save image',IFileName);
    else
        [FileName,PathName,format] = uiputfile(...
        {'*.dat','ASCII (*.dat) - intensity'; '*.dat','ASCII (*.dat) -
intensity masked'; '*.bmp','Bitmap (*.bmp) - intensity'; '*.bmp','Bitmap
(*.bmp) - intensity masked'},'FAIA - Save image',IFileName);
```

```
        if format>2, format=format+2; end
    end
    maksf=8; if isequal(FileName,0) || isequal(PathName,0) ||
isequal(format,0) || format>maksf, return; end
    if format<5             % ASCII
        switch format, case 1, im2save=IintBt; case 2, im2save=IintBt.*mask;
case 3, im2save=ImltBt; case 4, im2save=ImltBt.*mask; end
        im2save(isnan(im2save))=0;
        dlmwrite(fullfile(PathName,FileName),im2save,'\t');
    else                    % BMP
        if isequal(exist('export_fig'),2)   % export_fig
            if format>6 % flim
                Ihsv=IintBt-IintBt; Ihsv(:,:,1)=flipud(0.7*(ImltBtmax-
ImltBt)./(ImltBtmax-ImltBtmin));
                Ihsv(:,:,2)=1; Ihsv(:,:,3)=flipud((IintBt-
IintBtmin)./(IintBtmax-IintBtmin));
            end
            imSiz=size(Iint); f=figure('Color',[1,1,1],'Visible','off');
hIax=gca; colormap(cmap);
            switch format, case 5, hIIm=imagesc(flipud(IintBt)); case 6,
hIIm=imagesc(flipud(IintBt),'AlphaData',flipud(dispMask));
                case 7, hIIm=imagesc(hsv2rgb(Ihsv)); case 8,
hIIm=imagesc(hsv2rgb(Ihsv),'AlphaData',flipud(dispMask)); end
            fdy=imSiz(1); fdx=imSiz(2);
            bottomy=0; if fdy>246 bottomy=bottomy+fdy+40-246; end;
            set(f,'Position',[33,33,fdx+120,fdy+256]);
            xlabel(IrealSizeUnits);

set(hIax,'Xlim',[0.5,IintBSize(2)+0.5],'Ylim',[0.5,IintBSize(2)+0.5],'XTick',[
1,IintBSize(2)],'XTickLabel',[0,IrealSize(2)],'YTick',[1,IintBSize(1)],'YTickL
abel',[0,IrealSize(1)],'YDir','normal','TickDir','out','Units','pixels','Posit
ion',[50,bottomy+246-fdy,imSiz+1]);

hIcb=axes('Units','pixels','Position',[fdx+70,bottomy+140,10,100]);
            if format<7, hIcbIm=image(cmap2rgb(cmap)); else
hIcbIm=image(grayCmapImage); end
            ylabel('Int','FontSize',9,'Position',[3,132,1]);

set(hIcb,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick',[],'YDir','
normal','YTick',[0.5,256],'YTickLabel',[roundSP(IintBtmin,3),roundSP(IintBtmax
,3)],'YMinorTick','on');
            if format>6 % flim

hIcbmlt=axes('Units','pixels','Position',[fdx+70,bottomy+10,10,100]);
                hIcbmltIm=image(flimCmapImage); ylabel('\tau
(ns)','Position',[3 126 1],'FontSize',9);

set(hIcbmlt,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick',[],'YDir
','normal','YTick',[0.5,256],'YTickLabel',[roundSP(ImltBtmin,3),roundSP(ImltBt
max,3)],'YMinorTick','on');
            end
            export_fig(fullfile(PathName,FileName),'-a1',f); close(f); %'-
zbuffer'
        else    % no export_fig
            if format>6 % flim
                Ihsv=IintBt-IintBt; Ihsv(:,:,1)=0.7*(ImltBtmax-
ImltBt)./(ImltBtmax-ImltBtmin);
                Ihsv(:,:,2)=1; Ihsv(:,:,3)=(IintBt-IintBtmin)./(IintBtmax-
IintBtmin);
            end
            msgbox(['Could not find "export_fig".',char(10),'To show a color
scale export_fig function by Oliver Woodford is needed.',char(10),'Image will
be saved without it.'],'FAIA - Warning');
            switch format, case 5, im2save=IintBt; case 6,
im2save=IintBt.*mask;
```

```
                case 7, im2save=hsv2rgb(Ihsv); case 8,
Ihsv(:,:,3)=Ihsv(:,:,3).*mask; im2save=hsv2rgb(Ihsv); end
            if size(im2save,3)==1
                mini=min(im2save(:)); maxi=max(im2save(:));
                im2save=255*(im2save-mini)/(maxi-mini);

imwrite(uint8(im2save),colormap([cmapName,'(256)']),fullfile(PathName,FileName
),'bmp');
            else
                imwrite(im2save,fullfile(PathName,FileName),'bmp');
            end
        end
    end
end
%.................................................ISaveButtonCallback...

%...LSaveButtonCallback.............................................NESTED
function LSaveButtonCallback(hObject, ~)
    if isequal(get(hLflimRadio,'Enable'),'on')
        [FileName,PathName,format]=uiputfile(...
        {'*.dat','ASCII (*.dat) - intensity only'; '*.dat','ASCII (*.dat) -
intensity only masked'; '*.dat','ASCII (*.dat) - lifetimes only';
'*.dat','ASCII (*.dat) - lifetimes only masked';
        '*.bmp','Bitmap (*.bmp) - intensity false-color'; '*.bmp','Bitmap
(*.bmp) - intensity false-color masked'; '*.bmp','Bitmap (*.bmp) - FLIM';
'*.bmp','Bitmap (*.bmp) - FLIM masked'},'FAIA - Save image',LFileName);
    else
        [FileName,PathName,format] = uiputfile(...
        {'*.dat','ASCII (*.dat) - intensity'; '*.dat','ASCII (*.dat) -
intensity masked'; '*.bmp','Bitmap (*.bmp) - intensity'; '*.bmp','Bitmap
(*.bmp) - intensity masked'},'FAIA - Save image',LFileName);
        if format>2, format=format+2; end
    end
    maksf=8; if isequal(FileName,0) || isequal(PathName,0) ||
isequal(format,0) || format>maksf, return; end
    if rem(format+1,2) && ~isequal(LintBSize,size(mask)), msgbox('Cannot mask
the image - the sizes do not match.','FAIA - Error'); return; end
    if format<5             % ASCII
        switch format, case 1, im2save=LintBt; case 2, im2save=LintBt.*mask;
case 3, im2save=LmltBt; case 4, im2save=LmltBt.*mask; end
        im2save(isnan(im2save))=0;
        dlmwrite(fullfile(PathName,FileName),im2save,'\t');
    else                    % BMP
        if isequal(exist('export_fig'),2)   % export_fig
            if format>6 % flim
                Lhsv=LintBt-LintBt; Lhsv(:,:,1)=flipud(0.7*(LmltBtmax-
LmltBt)./(LmltBtmax-LmltBtmin));
                Lhsv(:,:,2)=1; Lhsv(:,:,3)=flipud((LintBt-
LintBtmin)./(LintBtmax-LintBtmin));
            end
            imSiz=size(Lint); f=figure('Color',[1,1,1],'Visible','off');
hLax=gca; colormap(cmap)
            switch format, case 5, hLIm=imagesc(flipud(LintBt)); case 6,
hLIm=imagesc(flipud(LintBt),'AlphaData',flipud(dispMask));
                case 7, hLIm=imagesc(hsv2rgb(Lhsv)); case 8,
hLIm=imagesc(hsv2rgb(Lhsv),'AlphaData',flipud(dispMask)); end

set(hLax,'Xlim',[0.5,LintBSize(2)+0.5],'Ylim',[0.5,LintBSize(2)+0.5],'XTick',[
1,LintBSize(2)],'XTickLabel',[0,LrealSize(2)],'YTick',[1,LintBSize(1)],'YTickL
abel',[0,LrealSize(1)],'YDir','normal','TickDir','out','Units','pixels','Posit
ion',[50,50,imSiz+1]);
            fdy=imSiz(1); fdx=imSiz(2); if fdy<250, fdy=250; end; if fdx<250,
fdx=250; end;
            set(f,'Position',[33,33,fdx+120,fdy+70]); xlabel(LrealSizeUnits);
            hLcb=axes('Units','pixels','Position',[imSiz(2)+50+10,imSiz(1)+50-
100-10,10,100]);
```

```
            if format<7, hLcbIm=image(cmap2rgb(cmap)); else
hLcbIm=image(grayCmapImage); end
            ylabel('Int','FontSize',9,'Position',[3,132,1]);

set(hLcb,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick',[],'YDir','
normal','YTick',[0.5,256],'YTickLabel',[roundSP(LintBtmin,3),roundSP(LintBtmax
,3)],'YMinorTick','on');
            if format>6 % flim

hLcbmlt=axes('Units','pixels','Position',[imSiz(2)+50+10,imSiz(1)+50-100-
140,10,100]);
                hLcbmltIm=image(flimCmapImage); ylabel('\tau
(ns)','Position',[3 126 1],'FontSize',9);

set(hLcbmlt,'YAxisLocation','right','ticklength',[0.02,0.005],'XTick',[],'YDir
','normal','YTick',[0.5,256],'YTickLabel',[roundSP(LmltBtmin,3),roundSP(LmltBt
max,3)],'YMinorTick','on');
            end
            export_fig(fullfile(PathName,FileName),'-a1',f); close(f);
        else    % no export_fig
            if format>6 % flim
                Lhsv=LintBt-LintBt; Lhsv(:,:,1)=0.7*(LmltBtmax-
LmltBt)./(LmltBtmax-LmltBtmin);
                Lhsv(:,:,2)=1; Lhsv(:,:,3)=(LintBt-LintBtmin)./(LintBtmax-
LintBtmin);
            end
            msgbox(['Could not find "export_fig".',char(10),'To show a color
scale export_fig function by Oliver Woodford is needed.',char(10),'Image will
be saved without it.'],'FAIA - Warning');
            switch format, case 5, im2save=LintBt; case 6,
im2save=LintBt.*mask;
                case 7, im2save=hsv2rgb(Lhsv); case 8,
Lhsv(:,:,3)=Lhsv(:,:,3).*mask; im2save=hsv2rgb(Lhsv); end
            if size(im2save,3)==1
                mini=min(im2save(:)); maxi=max(im2save(:));
                im2save=255*(im2save-mini)/(maxi-mini);

imwrite(uint8(im2save),colormap([cmapName,'(256)']),fullfile(PathName,FileName
),'bmp');
            else
                imwrite(im2save,fullfile(PathName,FileName),'bmp');
            end
        end
    end
end
%.....................................................LSaveButtonCallback...

%...CySliderCallback.....................................................NESTED
function CySliderCallback(hObject, ~)
    val=round(-get(hObject,'Value'));  % This slider is reversed
    set(hCyEdit,'String',val); set(hObject,'Value',-val);
    redrawA; redrawH; reposition90Labels;
end
%.....................................................CySliderCallback...

%...CyEditCallback.....................................................NESTED
function CyEditCallback(hObject, ~)
    val=round(str2double(get(hObject,'String')));
    if isnan(val), set(hObject,'String',-get(hCySlider,'Value'));
    else
        sliderMin=-get(hCySlider,'Max'); sliderMax=-get(hCySlider,'Min');
        if val<sliderMin, val=sliderMin; end; if val>sliderMax, val=sliderMax;
end
        set(hObject,'String',val); set(hCySlider,'Value',-val);
        redrawA; redrawH; reposition90Labels;
    end
```

```
end
%.....................................................CyEditCallback...

%...CxSliderCallback.................................................NESTED
function CxSliderCallback(hObject, ~)
    val=round(get(hObject,'Value'));
    set(hCxEdit,'String',val); set(hObject,'Value',val);
    redrawA; redrawH; reposition0Label;
end
%.....................................................CxSliderCallback...

%...CxEditCallback...................................................NESTED
function CxEditCallback(hObject, ~)
    val=round(str2double(get(hObject,'String')));
    if isnan(val), set(hObject,'String',get(hCxSlider,'Value'));
    else
        sliderMin=get(hCxSlider,'Min'); sliderMax=get(hCxSlider,'Max');
        if val<sliderMin, val=sliderMin; end; if val>sliderMax, val=sliderMax;
end
        set(hObject,'String',val); set(hCxSlider,'Value',val);
        redrawA; redrawH; reposition0Label;
    end
end
%.....................................................CxEditCallback...

%...rSliderCallback...................................................NESTED
function rSliderCallback(hObject, ~)
    val=round(get(hObject,'Value'));
    set(hrEdit,'String',val); set(hObject,'Value',val);
    redrawA; redrawH;
end
%.....................................................rSliderCallback...

%...rEditCallback.....................................................NESTED
function rEditCallback(hObject, ~)
    val=round(str2double(get(hObject,'String')));
    if isnan(val), set(hObject,'String',get(hrSlider,'Value'));
    else
        sliderMin=get(hrSlider,'Min'); sliderMax=get(hrSlider,'Max');
        if val<sliderMin, val=sliderMin; end; if val>sliderMax, val=sliderMax;
end
        set(hObject,'String',val); set(hrSlider,'Value',val);
        redrawA; redrawH;
    end
end
%.....................................................rEditCallback...

%...RSliderCallback...................................................NESTED
function RSliderCallback(hObject, ~)
    val=round(get(hObject,'Value'));
    set(hREdit,'String',val); set(hObject,'Value',val);
    redrawA; redrawH;
end
%.....................................................RSliderCallback...

%...REditCallback.....................................................NESTED
function REditCallback(hObject, ~)
    val=round(str2double(get(hObject,'String')));
    if isnan(val), set(hObject,'String',get(hRSlider,'Value'));
    else
        sliderMin=get(hRSlider,'Min'); sliderMax=get(hRSlider,'Max');
        if val<sliderMin, val=sliderMin; end; if val>sliderMax, val=sliderMax;
end
        set(hObject,'String',val); set(hRSlider,'Value',val);
        redrawA; redrawH;
    end
```

```
end
%.....................................................REditCallback...

%...HAScaleMaxCallback.............................................NESTED
function HAScaleMaxCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if val>AScale(1), AScale(2)=val;
end
    set(hObject,'String',AScale(2)); set(hHaxes(1),'YLim',AScale);
end
%.................................................HAScaleMaxCallback...

%...HAScaleMinCallback.............................................NESTED
function HAScaleMinCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if val<AScale(2), AScale(1)=val;
end
    set(hObject,'String',AScale(1)); set(hHaxes(1),'YLim',AScale);
end
%.................................................HAScaleMinCallback...

%...HmltScaleMaxCallback............................................NESTED
function HmltScaleMaxCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if val>mltScale(1),
mltScale(2)=val; end
    set(hObject,'String',mltScale(2)); set(hHaxes(2),'YLim',mltScale);
end
%................................................HmltScaleMaxCallback...

%...HmltScaleMinCallback............................................NESTED
function HmltScaleMinCallback(hObject, ~)
    val=str2double(get(hObject,'String')); if val<mltScale(2),
mltScale(1)=val; end
    set(hObject,'String',mltScale(1)); set(hHaxes(2),'YLim',mltScale);
end
%................................................HmltScaleMinCallback...

%...HbinEditCallback................................................NESTED
function HbinEditCallback(hObject, ~)
    val=str2double(get(hObject, 'String')); if val>=0.001 histBinWidth=val;
end
    set(hObject,'String',histBinWidth); redrawH;
end
%.....................................................HbinEditCallback...

%...HARadioCallback.................................................NESTED
function HARadioCallback(hObject, ~)
    redrawH;
end
%.....................................................HARadioCallback...

%...HIRadioCallback.................................................NESTED
function HIRadioCallback(hObject, ~)
    redrawH;
end
%.....................................................HIRadioCallback...

%...HLRadioCallback.................................................NESTED
function HLRadioCallback(hObject, ~)
    redrawH;
end
%.....................................................HLRadioCallback...

%...ASaveButtonCallback.............................................NESTED
function ASaveButtonCallback(hObject, ~)                                %
saving anisotropy image as ascii or bitmap
    len=min(length(IFileName),length(LFileName));
```

```
    i=1; while i<=len, if isequal(IFileName(i),LFileName(i)), i=i+1; else
break; end; end; i=i-1;
    FileName=IFileName(1:i);
    if ~get(hASaveRadio,'Value')
        [FileName,PathName,format] = uiputfile( {'*.aim','ASCII (*.aim)';
'*.aim','ASCII (*.aim) - masked image'; '*.bmp','Bitmap (*.bmp)';
'*.bmp','Bitmap (*.bmp) - masked image'; }, 'FAIA - Save anisotropy
image',FileName);
        maksf=4;
        if isequal(FileName,0) || isequal(PathName,0) || isequal(format,0) ||
format>maksf, return; end
        saveAImage(fullfile(PathName,FileName),format);
    else
        [PathName,FileName,~]=fileparts(FileName);
        saveAImage(fullfile(PathName,[FileName,'.aim']),1);
saveAImage(fullfile(PathName,[FileName,'_m.aim']),2);
        saveAImage(fullfile(PathName,[FileName,'.bmp']),3);
saveAImage(fullfile(PathName,[FileName,'_m.bmp']),4);
    end
end
%.................................................ASaveButtonCallback...

%...HSaveButtonCallback.............................................NESTED
function HSaveButtonCallback(hObject, ~)                                %
saving angular dependence plot
    len=min(length(IFileName),length(LFileName));
    i=1; while i<=len, if isequal(IFileName(i),LFileName(i)), i=i+1; else
break; end; end; i=i-1;
    AFileName=IFileName(1:i);
    [FileName,PathName,format] = uiputfile( {'*.apl','ASCII
(*.apl)';'*.bmp','Bitmap (*.bmp)'}, 'FAIA - Save plot',AFileName);
    if isequal(FileName,0) || isequal(PathName,0), return; end;
    if format==2
        if ~isequal(exist('export_fig'),2), msgbox(['Cannot find
"export_fig".',char(10),'export_fig function by Oliver Woodford is needed to
perform this action.',char(10),'Please install export_fig and try
again.'],'FAIA - Error'); return; end;
        f=figure('Color',background,'Visible','off'); axis square;
        [hHax,hHApl,hHIpl]=plotyy(-180:10:180,zeros(1,37),-
180:10:180,ones(1,37));

set(hHax(1),'YColor',[0,0,0],'FontSize',9,'Color',[0.83,0.83,0.83],'XAxisLocat
ion','top','XLim',[-200,200],'XTick',-
180:90:180,'YLim',AScale,'YTickMode','auto','Box','off','TickDir','out','XGrid
','on');
        set(hHApl,'Color',[0,0,0],'Visible','off');
        set(hHIpl,'Color',IColor,'Visible','off'); hold(hHax(2),'on');
hHLpl=plot(-
180:10:180,ones(1,37),'Color',LColor,'Parent',hHax(2),'Visible','off');
hold(hHax(2),'off');

set(hHax(2),'YColor',[0.65,0.85,0.75],'FontSize',9,'XTickLabel','','TickDir','
out','XLim',[-200,200],'XTick',-
180:90:180,'YLim',mltScale,'YTickMode','auto');
        ylabel(hHax(1),'R'); ylabel(hHax(2),'\tau  (ns)');
        if (get(hHARadio,'Value') && isequal(get(hHARadio,'Enable'),'on')),
set(hHApl,'XData',HA(:,1),'YData',HA(:,2),'Visible','on'); end
        if get(hHIRadio,'Value'),
set(hHIpl,'XData',HImltw(:,1),'YData',HImltw(:,2),'Visible','on'); end
        if get(hHLRadio,'Value'),
set(hHLpl,'XData',HLmltw(:,1),'YData',HLmltw(:,2),'Visible','on'); end
        legend([hHApl;hHIpl;hHLpl],'R','II - \tau','L -
\tau','Location','Best');
        export_fig(fullfile(PathName,FileName),'-a1','-nocrop',f); close(f);
return
    end
```

```
    % checking if HA,HIint,HLint,HImltw,HLmltw are available:
    isHA=0; if (isequal(get(hHARadio,'Enable'),'on') &&
get(hHARadio,'Value')), isHA=1; end
    isHI=0; if get(hHIRadio,'Value'), isHI=1; end
    isHL=0; if get(hHLRadio,'Value'), isHL=1; end
    results=nan(size(HIint,1),13); results(:,1)=HIint(:,1);
    if isHA, results(:,2:3)=HA(:,2:3); end % anisotropy
    if isHI, results(:,4:5)=HIint(:,2:3); results(:,8)=HImlt(:,2);
results(:,9)=HImltw(:,2); results(:,10)=HImlt(:,3); end % I lifetimes
    if isHL, results(:,6:7)=HLint(:,2:3); results(:,11)=HLmlt(:,2);
results(:,12)=HLmltw(:,2); results(:,13)=HLmlt(:,3); end % L lifetimes
    fid=fopen(fullfile(PathName,FileName),'w+t');
    if fid==-1, msgbox('Cannot open file for writting.','FAIA - Error');
return; end;

fprintf(fid,'%7s\t%7s\t%6s\t%7s\t%6s\t%7s\t%6s\t%6s\t%5s\t%6s\t%5s\t%6s\t%6s\n
','angle','R','+/-','Int(II)','+/-','Int(L)','+/-','T(II)','Tw(II)','+/-
','T(L)','Tw(L)','+/-');

fprintf(fid,'%7s\t%7s\t%6s\t%7s\t%6s\t%7s\t%6s\t%6s\t%5s\t%6s\t%5s\t%6s\t%6s\n
','(deg)','','','','','','(ns)','(ns)','(ns)','(ns)','(ns)','(ns)');

fprintf(fid,'%7.2f\t%7.4f\t%6.4f\t%7.3f\t%6.3f\t%7.3f\t%6.3f\t%6.3f\t%6.3f\t%5
.3f\t%6.3f\t%6.3f\t%5.3f\n',results');
    fclose(fid);
end
%..................................................HSaveButtonCallback...

%...closingFaiaCallback.............................................NESTED
function closingFaiaCallback(hObject, ~)                                %
saving current parameters
    if isequal(currentPath,''), currentPath=pwd; end

save(configFileName,'G','LmltShift','AScale','mltScale','binSize','histBinWidt
h','cmapName','IFileName','LFileName','currentPath','cmap');
    delete(gcbf);
end
%.................................................closingFaiaCallback...
% Other functions:

%...newIFile........................................................NESTED
function newIFile                                                       %
new I file appeared - lets take care of it
    if ~isequal(exist(IFileName,'file'),2), IFileName=''; end
    [~,nam,ext]=fileparts(IFileName); IFileNameShort=[nam,ext];
    if isempty(IFileName), return; end
    [head,Iint,Imlt]=loadSPTascii(IFileName);
    if isnan(Iint), return; end;
    wStart=strfind(head,'Width: ')+7; head=head(wStart:end);
wEnd=strfind(head,' '); w=head(1:wEnd-1);
    IrealSizeUnits=head(wEnd+1:wEnd+2);
    hStart=strfind(head,'Height: ')+8; head=head(hStart:end);
hEnd=strfind(head,' '); h=head(1:hEnd-1);
    IrealSize=[str2double(h),str2double(w)];
    set(hIFileLabel,'String',IFileNameShort);
    if ~isequal(size(Iint),size(Imlt)), Imlt=NaN; end
    binIImage;
    IintBtmin=get(hIintSlider,'Value'); % get values from slider
    IintBtmax=get(hIintSlider,'Max');
    IintBtBottom=get(hIintSlider,'Min');
    set(hIintSlider,'Value',IintBtBottom);
    IintBtmin=ceil(IintBtmin*(IintBmax-IintBmin)/(IintBtmax-
IintBtBottom)+IintBmin);
    if IintBtmin>IintBmin, else IintBtmin=IintBmin; end
    IintBtBottom=IintBmin; IintBtmax=floor(IintBmax);
    set(hIintSlider,'Max',IintBtmax); set(hIintSlider,'Value',IintBtmax);
```

```
    set(hIintSlider,'Min',IintBtBottom); set(hIintSlider,'Value',IintBtmin);
    set(hIintMinEdit,'String',IintBtmin);
set(hIintMaxEdit,'String',IintBtmax);
    thresholdIImage;
    if isnan(Imlt)  % no lifetimes
        set(hIflimRadio,'Value',0,'Enable','off');
set(hIflimLabel,'Enable','off');
        set(hHIRadio,'Value',0,'Enable','off');
        set(hIflimMinEdit,'String','min','Enable','off');
set(hIflimMaxEdit,'String','max','Enable','off');
    else
        if ImltBtmin>ImltBmin, else ImltBtmin=ImltBmin; end
        if ImltBtmax<ImltBmax, else ImltBtmax=ImltBmax; end
        set(hIflimMinEdit,'String',ImltBtmin,'Enable','on');
        set(hIflimMaxEdit,'String',ImltBtmax,'Enable','on');
        set(hIflimRadio,'Enable','on'); set(hIflimLabel,'Enable','on');
        set(hHIRadio,'Value',1,'Enable','on');
        lifetimeTuneIImage;
    end
    redrawI; redrawA; redrawH;
end
%.....................................................................newIFile...

%...newLFile.....................................................................NESTED
function newLFile                                                                %
new L file appeared - lets take care of it
    if ~isequal(exist(LFileName,'file'),2), LFileName=''; end
    [~,nam,ext]=fileparts(LFileName); LFileNameShort=[nam,ext];
    if isempty(LFileName), return; end
    [head,Lint,Lmlt]=loadSPTascii(LFileName);
    if isnan(Lint), return; end;
    wStart=strfind(head,'Width: ')+7; head=head(wStart:end);
wEnd=strfind(head,' '); w=head(1:wEnd-1);
    LrealSizeUnits=head(wEnd+1:wEnd+2);
    hStart=strfind(head,'Height: ')+8; head=head(hStart:end);
hEnd=strfind(head,' '); h=head(1:hEnd-1);
    LrealSize=[str2double(h),str2double(w)];
    set(hLFileLabel,'String',LFileNameShort);
    if ~isequal(size(Lint),size(Lmlt)), Lmlt=NaN;
    else Lmlt=Lmlt+LmltShift; end;  % correction for symphotime
    binLImage;
    LintBtmin=get(hLintSlider,'Value'); % get values from slider
    LintBtmax=get(hLintSlider,'Max');
    LintBtBottom=get(hLintSlider,'Min');
    set(hLintSlider,'Value',LintBtBottom);
    LintBtmin=ceil(LintBtmin*(LintBmax-LintBmin)/(LintBtmax-
LintBtBottom)+LintBmin);
    if LintBtmin>LintBmin, else LintBtmin=LintBmin; end
    LintBtBottom=LintBmin; LintBtmax=floor(LintBmax);
    set(hLintSlider,'Max',LintBtmax); set(hLintSlider,'Value',LintBtmax);
    set(hLintSlider,'Min',LintBtBottom); set(hLintSlider,'Value',LintBtmin);
    set(hLintMinEdit,'String',LintBtmin);
set(hLintMaxEdit,'String',LintBtmax);
    thresholdLImage;
    if isnan(Lmlt)  % no lifetimes
        set(hLflimRadio,'Value',0,'Enable','off');
set(hLflimLabel,'Enable','off');
        set(hHLRadio,'Value',0,'Enable','off');
        set(hLflimMinEdit,'String','min','Enable','off');
set(hLflimMaxEdit,'String','max','Enable','off');
    else
        if LmltBtmin>LmltBmin, else LmltBtmin=LmltBmin; end
        if LmltBtmax<LmltBmax, else LmltBtmax=LmltBmax; end
        set(hLflimMinEdit,'String',LmltBtmin,'Enable','on');
        set(hLflimMaxEdit,'String',LmltBtmax,'Enable','on');
        set(hLflimRadio,'Enable','on'); set(hLflimLabel,'Enable','on');
```

```
        set(hHLRadio,'Value',1,'Enable','on');
        lifetimeTuneLImage;
    end
    redrawL; redrawA; redrawH;
end
%.....................................................newLFile...

%...binIImage.....................................................NESTED
function binIImage
    if sum(rem(size(Iint),binSize)) %sizes do not match - can't be binned
       IintB=NaN; ImltB=NaN;
    else %binning:
        IintB=bin(Iint,binSize(1),binSize(2));
        if isnan(Imlt), ImltB=NaN; else ImltB=bin(Imlt,binSize(1),binSize(2));
end
    end
    IintBSize=size(IintB);
    IintBmin=min(min(IintB)); IintBmax=max(max(IintB));
    ImltBmin=min(min(ImltB)); ImltBmax=max(max(ImltB));
end
%.....................................................binIImage...

%...binLImage.....................................................NESTED
function binLImage
    if sum(rem(size(Lint),binSize)) %sizes do not match - can't be binned
       LintB=NaN; LmltB=NaN;
    else %binning:
        LintB=bin(Lint,binSize(1),binSize(2));
        if isnan(Lmlt), LmltB=NaN; else LmltB=bin(Lmlt,binSize(1),binSize(2));
end
    end
    LintBSize=size(LintB);
    LintBmin=min(min(LintB)); LintBmax=max(max(LintB));
    LmltBmin=min(min(LmltB)); LmltBmax=max(max(LmltB));
end
%.....................................................binLImage...

%...thresholdIImage.....................................................NESTED
function thresholdIImage
    IintBt=IintB; IintBt(IintBt<IintBtmin)=NaN; IintBt(IintBt>IintBtmax)=NaN;

set(hIImageCBintAxis,'YTickLabel',[roundSP(IintBtmin,3),roundSP(IintBtmax,3)])
;
end
%.....................................................thresholdIImage...

%...thresholdLImage.....................................................NESTED
function thresholdLImage
    LintBt=LintB; LintBt(LintBt<LintBtmin)=NaN; LintBt(LintBt>LintBtmax)=NaN;

set(hLImageCBintAxis,'YTickLabel',[roundSP(LintBtmin,3),roundSP(LintBtmax,3)])
;
end
%.....................................................thresholdLImage...

%...lifetimeTuneIImage.....................................................NESTED
function lifetimeTuneIImage
    ImltBt=ImltB; ImltBt(ImltBt<ImltBtmin)=NaN; ImltBt(ImltBt>ImltBtmax)=NaN;

set(hIImageCBmltAxis,'YTickLabel',[roundSP(ImltBtmin,3),roundSP(ImltBtmax,3)])
;
end
%.....................................................lifetimeTuneIImage...

%...lifetimeTuneLImage.....................................................NESTED
function lifetimeTuneLImage
```

```
    LmltBt=LmltB; LmltBt(LmltBt<LmltBtmin)=NaN; LmltBt(LmltBt>LmltBtmax)=NaN;

set(hLImageCBmltAxis,'YTickLabel',[roundSP(LmltBtmin,3),roundSP(LmltBtmax,3)])
;
end
%.....................................................lifetimeTuneLImage...

%...redrawI.............................................................NESTED
function redrawI
    Ihsv=IintBt-IintBt; Ihsv(:,:,2)=1; Ihsv(:,:,3)=IintBt;
    currentIImageSize=[get(hIImage,'XData');get(hIImage,'YData')];
    currentIImageSize=currentIImageSize(:,2)';
    if ~isequal(currentIImageSize,IintBSize)
        set(hIaxis,'Xlim',[0.5, IintBSize(2)+0.5],'Ylim',[0.5,
IintBSize(2)+0.5],'XTick',[1,IintBSize(2)/2,IintBSize(2)],'YTick',[1,IintBSize
(1)/2,IintBSize(1)],'YDir','normal');
    end
    flim=get(hIflimRadio,'Value');
    if flim
        Ihsv(:,:,1)=flipud(0.7*(ImltBtmax-ImltBt)./(ImltBtmax-ImltBtmin));  %
hue
        Ihsv(:,:,2)=1;                                                      %
saturation
        Ihsv(:,:,3)=flipud((IintBt-IintBtmin)./(IintBtmax-IintBtmin));      %
value
        imageData=hsv2rgb(Ihsv);
        set(hIImageCBmltAxis,'Visible','on');
set(hIImageCBmlt,'Visible','on');
    else
        imageData=flipud(IintBt);
        set(hIImageCBint,'CData',cmap2rgb(cmap));
        set(hIImageCBmltAxis,'Visible','off');
set(hIImageCBmlt,'Visible','off');
    end
    if get(hImaskRadio,'Value') && isequal(size(dispMask),IintBSize)
        alphaD=flipud(dispMask); else alphaD=1; end
    set(hIImage,'AlphaData',alphaD,'CData',imageData); drawnow;
end
%.............................................................redrawI...

%...redrawL.............................................................NESTED
function redrawL
    Lhsv=LintBt-LintBt; Lhsv(:,:,2)=1; Lhsv(:,:,3)=LintBt;
    currentLImageSize=[get(hLImage,'XData');get(hLImage,'YData')];
    currentLImageSize=currentLImageSize(:,2)';
    if ~isequal(currentLImageSize,LintBSize)
        set(hLaxis,'Xlim',[0.5, LintBSize(2)+0.5],'Ylim',[0.5,
LintBSize(2)+0.5],'XTick',[1,LintBSize(2)/2,LintBSize(2)],'YTick',[1,LintBSize
(1)/2,LintBSize(1)],'YDir','normal');
    end
    flim=get(hLflimRadio,'Value');
    if flim
        Lhsv(:,:,1)=flipud(0.7*(LmltBtmax-LmltBt)./(LmltBtmax-LmltBtmin));  %
hue
        Lhsv(:,:,2)=1;                                                      %
saturation
        Lhsv(:,:,3)=flipud((LintBt-LintBtmin)./(LintBtmax-LintBtmin));      %
value
        imageData=hsv2rgb(Lhsv);
        set(hLImageCBmltAxis,'Visible','on');
set(hLImageCBmlt,'Visible','on');
    else
        imageData=flipud(LintBt);
        set(hLImageCBint,'CData',cmap2rgb(cmap));
        set(hLImageCBmltAxis,'Visible','off');
set(hLImageCBmlt,'Visible','off');
```

```
    end
    if get(hLmaskRadio,'Value') && isequal(size(dispMask),LintBSize)
        alphaD=flipud(dispMask); else alphaD=1; end
    set(hLImage,'AlphaData',alphaD,'CData',imageData); drawnow;
end
%.....................................................redrawL...

%...redrawA.....................................................NESTED
function redrawA                                                %
calculates anisotropy image and mask and displays them (displays also masks in
I and L images)
    if ~isequal(size(IintBt),size(LintBt))  % no anisotropy will be calculated
        if ~isequal(sum(sum(LintBt)),0), maskSize=size(LintBt); end
        if ~isequal(sum(sum(IintBt)),0), maskSize=size(IintBt); end
        createMask(maskSize);
        set(hAImage,'Visible','off'); set(hHARadio,'Enable','off');
set(hASaveButton,'Enable','off');
    else    % I and L images are equal in sizes
        A=(IintBt-LintBt*G)./(IintBt+2*G*LintBt);   % calculating anisotropy
        A(IintBt==0)=NaN; A(LintBt==0)=NaN;
        newASize=size(A);
        createMask(newASize);
        if isequal(newASize,ASize)
            set(hAImage,'Cdata',A,'AlphaData',dispMask,'Visible','on');
        else
            axes(hAaxis);
            hAImage=imagesc(A,'AlphaData',dispMask);
            set(hAaxis,'TickLength',[0,0],'XTick',[],'YTick',[]);
            ASize=newASize;
        end
        set(hHARadio,'Enable','on'); set(hASaveButton,'Enable','on');
    end
    if get(hImaskRadio,'Value'), redrawI; end
    if get(hLmaskRadio,'Value'), redrawL; end
    drawnow;
end
%.....................................................redrawA...

%...createMask..................................................NESTED
function createMask(imSize)
    % refresh limits and values of Cy,Cx,r,R sliders/edits:
    CxSliderMax=get(hCxSlider,'Max');
    CySliderMax=-get(hCySlider,'Min');
    if ~isequal([CySliderMax,CxSliderMax],imSize)
        CxSliderVal=get(hCxSlider,'Value'); CySliderVal=-
get(hCySlider,'Value');
        rSliderMax=get(hrSlider,'Max'); RSliderMax=get(hRSlider,'Max');
        rSliderVal=get(hrSlider,'Value'); RSliderVal=get(hRSlider,'Value');
        cx=round(CxSliderVal/CxSliderMax*imSize(2));
cy=round(CySliderVal/CySliderMax*imSize(1));
        diagon=ceil(sqrt(sum((imSize-2).^2)));
        r=round(rSliderVal/rSliderMax*diagon);
R=round(RSliderVal/RSliderMax*diagon);
        set(hCxSlider,'Value',1); set(hCxSlider,'Max',imSize(2));
set(hCxSlider,'Value',cx,'SliderStep',[1/imSize(2),5/imSize(2)]);
set(hCxEdit,'String',cx);
        set(hCySlider,'Value',-1); set(hCySlider,'Min',-imSize(1));
set(hCySlider,'Value',-cx,'SliderStep',[1/imSize(1),5/imSize(1)]);
set(hCyEdit,'String',cx);
        set(hrSlider,'Value',0); set(hrSlider,'Max',diagon);
set(hrSlider,'Value',r,'SliderStep',[1/diagon,5/diagon]);
set(hrEdit,'String',r);
        set(hRSlider,'Value',0); set(hRSlider,'Max',diagon);
set(hRSlider,'Value',R,'SliderStep',[1/diagon,5/diagon]);
set(hREdit,'String',R);
    else
```

```
        cx=get(hCxSlider,'Value'); cy=-get(hCySlider,'Value');
        r=get(hrSlider,'Value'); R=get(hRSlider,'Value');
    end
    % create mask:
    if R==0, mask=ones(imSize);
    else
        mask=ones(imSize)*shade;
        for y=1:imSize(1)
            for x=1:imSize(2)
                ci=round((x-cx)^2+(y-cy)^2);
                if ci>round(r^2) && ci<round(R^2), mask(y,x)=1; end
            end
        end
    end
    dispMask=mask; dispMask(:,cx)=0; dispMask(round(cy),:)=0;
    mask(mask~=1)=NaN;
end
%.....................................................createMask...

%...redrawH.....................................................NESTED
function redrawH
    cx=get(hCxSlider,'Value'); cy=-get(hCySlider,'Value');
    if get(hHARadio,'Value') && isequal(get(hAImage,'Visible'),'on')
        HA=angularDependence(A.*mask,cx,cy,histBinWidth);

set(hHAplot,'XData',HA(:,1),'YData',HA(:,2),'UData',HA(:,3),'LData',HA(:,3),'V
isible','on');
    else
        set(hHAplot,'Visible','off');
    end
    if get(hHIRadio,'Value') && isequal(get(hHIRadio,'Enable'),'on')
        tmpint=IintBt; tmpint(tmpint==0)=NaN; tmpmlt=ImltBt;
tmpmlt(isnan(tmpint))=NaN;
        if isequal(IintBSize,size(mask))
            HImlt=angularDependence(tmpmlt.*mask,cx,cy,histBinWidth);
            HImltw=angularDependence(tmpmlt.*tmpint.*mask,cx,cy,histBinWidth);
            HIint=angularDependence(tmpint.*mask,cx,cy,histBinWidth);
        else
            HImlt=angularDependence(tmpmlt,cx,cy,histBinWidth);
            HImltw=angularDependence(tmpmlt.*tmpint,cx,cy,histBinWidth);
            HIint=angularDependence(tmpint,cx,cy,histBinWidth);
        end
        HImltw(:,2)=HImltw(:,2)./HIint(:,2);

set(hHIplot,'XData',HImltw(:,1),'YData',HImltw(:,2),'UData',HImlt(:,3),'LData'
,HImlt(:,3),'Visible','on');
    else
        set(hHIplot,'Visible','off');
    end
    if get(hHLRadio,'Value') && isequal(get(hHLRadio,'Enable'),'on')
        tmpint=LintBt; tmpint(tmpint==0)=NaN; tmpmlt=LmltBt;
tmpmlt(isnan(tmpint))=NaN;
        if isequal(LintBSize,size(mask))
            HLmlt=angularDependence(tmpmlt.*mask,cx,cy,histBinWidth);
            HLmltw=angularDependence(tmpmlt.*tmpint.*mask,cx,cy,histBinWidth);
            HLint=angularDependence(tmpint.*mask,cx,cy,histBinWidth);
        else
            HLmlt=angularDependence(tmpmlt,cx,cy,histBinWidth);
            HLmltw=angularDependence(tmpmlt.*tmpint,cx,cy,histBinWidth);
            HLint=angularDependence(tmpint,cx,cy,histBinWidth);
        end
        HLmltw(:,2)=HLmltw(:,2)./HLint(:,2);

set(hHLplot,'XData',HLmltw(:,1),'YData',HLmltw(:,2),'UData',HLmlt(:,3),'LData'
,HLmlt(:,3),'Visible','on');
    else
```

```
        set(hHLplot,'Visible','off');
    end
end
%.....................................................................redrawH...

%...reposition0Label.................................................NESTED
function reposition0Label
    sMax=get(hCxSlider,'Max'); sVal=get(hCxSlider,'Value');
    set(hA0Label,'Position',[94+268*(sVal-1)/(sMax-1),365,30,11]);
end
%.....................................................reposition0Label...

%...reposition90Labels................................................NESTED
function reposition90Labels
    sMax=-get(hCySlider,'Min'); sVal=-get(hCySlider,'Value');
    set(hAm90Label,'Position',[73,357-272*(sVal-1)/(sMax-1),30,11]);
    set(hAp90Label,'Position',[376,357-272*(sVal-1)/(sMax-1),30,11]);
end
%.....................................................reposition90Labels...

%...saveAImage........................................................NESTED
function saveAImage(fileName,format)
    if format<3
        if format==1, im2save=A; end; if format==2, im2save=A.*mask; end;
        im2save(isnan(im2save))=0;
        dlmwrite(fileName,im2save,'\t');
    else
        if isequal(exist('export_fig'),2)
            imSiz=size(Iint);
            f=figure('Color',[1,1,1],'Visible','off'); hAax=gca;
colormap(cmap);
            set(hAax,'Box','on');
            if format==3, hAIm=imagesc(A); else
hAIm=imagesc(A,'AlphaData',dispMask); end

set(hAax,'TickLength',[0,0],'XTick',[1,IintBSize(2)],'XTickLabel',[0,IrealSize
(2)],'YTick',[1,IintBSize(1)],'YTickLabel',[0,IrealSize(1)],'Units','pixels','
Position',[50,50,imSiz+1]);
            xlabel(IrealSizeUnits);
            set(f,'Position',[1,1,imSiz(2)+120,imSiz(1)+70]);
            hAcb=axes('Units','pixels','Position',[imSiz(2)+50+10,imSiz(1)+50-
100-10,10,100]);
            hAcbIm=image(cmap2rgb(cmap));
ylabel('Anisotropy','FontSize',10,'Position',[3,132,1]);

set(hAcb,'YAxisLocation','right','ticklength',[0.02,0.005],'YDir','normal','XT
ick',[],'YTick',[0.5,256],'YTickLabel',[roundSP(min(A(:)),2),roundSP(max(A(:))
,2)],'YMinorTick','on');
            export_fig(fileName,'-a1',f); close(f);
        else
            msgbox(['Could not find "export_fig".',char(10),'To show a color
scale export_fig function by Oliver Woodford is needed.',char(10),'Image will
be saved without it.'],'FAIA - Warning');
            if format==3, im2save=A; end; if format==4, im2save=A.*mask; end;
            mini=min(im2save(:)); maxi=max(im2save(:));
            im2save=255*(im2save-mini)/(maxi-mini);

imwrite(uint8(im2save),colormap([cmapName,'(256)']),fileName,'bmp');
        end
    end
end
%.....................................................................saveAimage...


end     %END OF THE MAIN FUNCTION (faia)
%=====================================================================faia===
```

```
%__angularDependence_____________________________________________________
function r=angularDependence(im, center_x, center_y, binSize)
%   r = angularDependence(im, center_x, center_y, binSize)
%   Calculating angular depennce of the im values around the center point
%   (center_x, center_y). Returns bins (of width binSize; except the first
%   and the last one) with the following columns:
%   1) mean angle (deg), 2) mean im value 3) standard error
%   piotr.jurkiewicz@jh-inst.cas.cz
[max_y,max_x]=size(im);
angVals=NaN(max_y*max_y,2);     %initializing matrix for storing the angles
and values
angValLen=0;
for y=1:max_y
    dy=center_y-y;
    for x=1:max_x
        imValue=im(y,x); if isnan(imValue), continue; end
        angValLen=angValLen+1;
        dx=x-center_x;
        angle=atan(dx/dy)*180/pi+(sign(dy)-abs(sign(dy)))*(abs(sign(dx))-
sign(dx)-1)*90;
        angVals(angValLen,:)=[angle, imValue];
    end
end
angVals=sortrows(angVals(1:angValLen,:),1);   %sorting [angle, value] pairs
with ascending angle
if (angValLen~=0 & isnan(angVals(angValLen,1)))   %NaN angle can be there if
the center is exactly at one of image points
    angValLen=angValLen-1;
    angVals=angVals(1:angValLen,:);
end
bins=binSize/2:binSize:180;
bins=sort([-bins,bins]);
if bins(end)~=180, bins=[bins,180]; end
binsLen=size(bins,2);
bins=[bins',nan(binsLen,2)];   %with 3 columns
binContent=zeros(angValLen,1);
currBin=1;
currBinMember=0;
currAngVal=1;
while currAngVal<=angValLen
    if angVals(currAngVal,1)<=bins(currBin) %another bin member -> add it to
binContent
        currBinMember=currBinMember+1;
        binContent(currBinMember)=angVals(currAngVal,2);
        currAngVal=currAngVal+1;
    else    %finish current bin and proceed to the next one
        m=mean(binContent(1:currBinMember));    %calculate mean
        se=sem(binContent(1:currBinMember));    %calculate standard error of
mean
        bins(currBin,:)=[bins(currBin,1), m, se];   %save to bins
        binContent(1:currBinMember)=0;          %clean binContent
        currBinMember=0;
        currBin=currBin+1;
    end
end
if currBinMember~=0
    m=mean(binContent(1:currBinMember));    %calculate mean
    se=sem(binContent(1:currBinMember));    %calculate standard error of mean
    bins(currBin,:)=[bins(currBin,1), m, se];   %save to bins   ..-0.5*binSize
end
%calculating bin centers:
bins(1)=(bins(1)-180)*0.5;
bins(binsLen)=(bins(binsLen-1)+180)*0.5;
bins(2:binsLen-1)=bins(2:binsLen-1)-0.5*binSize;
r=bins;
```

```
end
%_____________________________________________________angularDependence___


%___loadSPTascii_____________________________________________________________
function [head,int,mlt]=loadSPTascii(fileName)
% Reads flim data from ascii file exported from Pico Quant SymphoTime
% [head,int,mlt]=loadSPTascii(fileName), where head is text of the header,
% int is a matrix of intensity values, mlt is a matrix of mean lifetimes
% piotr.jurkiewicz@jh-inst.cas.cz
    fileContent=fileread(fileName);
    intStart=strfind(fileContent,'Intens. :');
    mltStart=strfind(fileContent,'Aver. LT:');
    if isempty(intStart), head=''; int=NaN; mlt=NaN; return; end
    head=fileContent(1:intStart-1);
    s=strfind(head,'Pixels (x/y):');
    imSize=sscanf(head(s+13:end),'%d / %d');
    if isempty(mltStart)
        [int,c]=sscanf(fileContent(intStart+10:end),'%f');
        if isequal(c,prod(imSize)), int=reshape(int,imSize(1),imSize(2))';
        else int=str2num(fileContent(intStart+10:end)); end; mlt=NaN;
    else
        [int,c]=sscanf(fileContent(intStart+10:mltStart-1),'%f');
        if isequal(c,prod(imSize)), int=reshape(int,imSize(1),imSize(2))';
        else int=str2num(fileContent(intStart+10:mltStart-1)); end
        [mlt,c]=sscanf(fileContent(mltStart+10:end),'%f');
        if isequal(c,prod(imSize)), mlt=reshape(mlt,imSize(1),imSize(2))';
        else mlt=str2num(fileContent(mltStart+10:end)); end
    end
end
%_____________________________________________________loadSPTascii___

%___sem_______________________________________________________________________
function y = sem(varargin)
%SEM Standard error of the mean.
%   For vectors, Y = SEM(X) returns the experimental standard deviation of the
%   mean according to the Guide to the Expression of Uncertainty in
Measurement
%   from International Organization for Standardization
%   pit.jurkiewicz@gmail.com
y = std(varargin{:})./sqrt(size(varargin{:},1));
end
%_____________________________________________________________________sem___

%___roundSP___________________________________________________________________
function r=roundSP(x,n)
%r=roundSP(x,n) rounds x to n significant places(digits)
if x==0, r=0; return; end
minus=0;
if x<0, x=-x; minus=1; end
m=floor(log10(x))+1; m=10^(m-n); r=round(x/m)*m;
if minus, r=-r; end;
end
%_____________________________________________________________roundSP___

%___bin_______________________________________________________________________
function r=bin(x,p,q)
%Simple matrix binning
%   r = BIN(x,p,q) averages! the elements of matrix m pxq creating new matrix
r
%   Dimensions of matrix x must divisible by p and q (respectively).
[m,n]=size(x);
if sum([rem(m,p), rem(n,q)])
      error('MATLAB:bin:InvalidArgument',...
        'Matrix dimensions must be divisible with the new ones.'); end
```

```
x=sum( reshape(x,p,[]), 1 ); x=reshape(x,m/p,[]).';
x=sum( reshape(x,q,[]), 1 ); x=reshape(x,n/q,[]).'; r=x./(p*q);
end
%_____________________________________________________________________bin___

%___cmap2rgb_____________________________________________________________
function rgb=cmap2rgb(cmap)
% Changes color map to rgb matrix ready to be displayed with image
s=size(cmap); rgb=zeros(s(1),1,s(2)); rgb(:,1,:)=cmap(:,:);
end
%_____________________________________________________________cmap2rgb___

%{
FREEWARE LICENCE

Copyright (c) 2014, Piotr Jurkiewicz
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
%}
```