```
function varargout = Raman_process_v5(varargin)
% RAMAN_PROCESS_V5 MATLAB code for Raman_process_v5.fig
%      RAMAN_PROCESS_V5, by itself, creates a new RAMAN_PROCESS_V5 or raises
%      the existing singleton*.
%
%      H = RAMAN_PROCESS_V5 returns the handle to a new RAMAN_PROCESS_V5 or the
%      handle to the existing singleton*.
%
%      RAMAN_PROCESS_V5('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RAMAN_PROCESS_V5.M with the given input arguments.
%
%      RAMAN_PROCESS_V5('Property','Value',...) creates a new RAMAN_PROCESS_V5 or
%      raises the existing singleton*.  Starting from the left, property value pairs
%      are applied to the GUI before Raman_process_v5_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Raman_process_v5_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Raman_process_v5

% Last Modified by GUIDE v2.5 11-Jan-2018 10:51:53

% Author(s): A. Fisher, K. Meissner
% Centre of NanoHealth, College of Science,
% Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

% Begin initialization code - DO NOT EDIT


    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @Raman_process_v5_OpeningFcn, ...
                       'gui_OutputFcn',  @Raman_process_v5_OutputFcn, ...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
% End initialization code - DO NOT EDIT
end


% --- Executes just before Raman_process_v5 is made visible.
function Raman_process_v5_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Raman_process_v5 (see VARARGIN)
```

```
    % Choose default command line output for Raman_process_v5
    handles.output = hObject;

    % Update handles structure
    guidata(hObject, handles);
    xlabel('Raman Shift (cm^{-1})');
    ylabel('Signal');
    title('Input Data');
    handles.lt_bl = [.871,.922,.98];
    handles.dk_bl = [.729,.831,.957];
    handles.lt_rd = [.961,.922,.922];
    handles.dk_rd = [.925,.839,.839];
    guidata(hObject,handles);

    % UIWAIT makes Raman_process_v5 wait for user response (see UIRESUME)
    % uiwait(handles.figure1);
end


% --- Outputs from this function are returned to the command line.
function varargout = Raman_process_v5_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    % Get default command line output from handles structure
    varargout{1} = handles.output;
end


% --- Executes on button press in export.
function export_Callback(hObject, eventdata, handles)
% hObject    handle to export (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    if get(handles.exp_fitparams,'Value') == 1
        assignin('base','fit_param',handles.fit_array);
    end
    if get(handles.exp_fitgood,'Value') == 1
        assignin('base','fit_goodness',handles.goodness_array);
    end
    if get(handles.exp_fluor_fitparams,'Value') == 1
        assignin('base','fit_param',handles.fluor_fit_array);
    end
    if get(handles.exp_fluor_fitgood,'Value') == 1
        assignin('base','fit_goodness',handles.fluor_goodness_array);
    end
    if get(handles.exp_mag,'Value') == 1
        assignin('base','image_mag',handles.image_mag);
    end
    if get(handles.exp_ctr,'Value') == 1
        assignin('base','image_ctr',handles.image_ctr);
    end
    if get(handles.exp_width,'Value') == 1
        assignin('base','image_width',handles.image_width);
    end
    if get(handles.exp_r2,'Value') == 1
        assignin('base','image_r2',handles.image_r2);
```

```
    end
    if get(handles.exp_fluor_mag,'Value') == 1
        assignin('base','image_mag',handles.fluor_mag);
    end
    if get(handles.exp_fluor_asym,'Value') == 1
        assignin('base','image_ctr',handles.fluor_asym);
    end
    if get(handles.exp_fluor_width,'Value') == 1
        assignin('base','image_width',handles.fluor_width);
    end
    if get(handles.exp_fluor_r2,'Value') == 1
        assignin('base','image_r2',handles.fluor_r2);
    end
    if get(handles.exp_raw,'Value') == 1
        assignin('base','data',handles.data);
        assignin('base','shift_axis',handles.shift_axis);
    end
    if get(handles.exp_raw_dec,'Value') == 1
        assignin('base','data_dec',handles.data_dec);
    end
    if get(handles.exp_f_dec,'Value') == 1
        assignin('base','data_f_dec',handles.data_f_dec);
    end
    if get(handles.exp_f_b_dec,'Value') == 1
        assignin('base','data_f_b_dec',handles.data_f_b_dec);
    end
    if (get(handles.exp_raw_dec,'Value') == 1) || (get(handles.exp_f_b_dec,'Value')...
            == 1) || (get(handles.exp_f_b_dec,'Value') == 1)
        assignin('base','shift_axis_dec',handles.shift_axis_dec);
    end
end


% --- Executes on button press in clear.
function clear_Callback(hObject, eventdata, handles)
% hObject    handle to clear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    cla(handles.input_plot);
    handles.data=[];handles.shift_axis=[];
    handles.data_dec=[];handles.data_f_dec=[];handles.data_f_b_dec=[];...
        handles.shift_axis_dec=[];
    handles.image_mag=[];handles.image_ctr=[];handles.image_width=[];...
        handles.image_r2=[];
    handles.rows=0;handles.cols=0;handles.spec=0;
    axis auto;
    xlabel('Raman Shift (cm^{-1})');
    ylabel('Signal');
    title('Input Data');
    s = sprintf('Rows: %.0f',handles.rows);
    set(handles.row_label,'String',s);
    s = sprintf('Cols: %.0f',handles.cols);
    set(handles.col_label,'String',s);
    set(handles.file_info,'String','No File');
    set(handles.decimate,'BackgroundColor',handles.dk_rd);
    set(handles.average,'BackgroundColor',handles.dk_rd);
    set(handles.process,'BackgroundColor',handles.dk_rd);
    set(handles.file_load,'BackgroundColor',handles.dk_bl);
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    set(handles.sav_raw,'Value',0);
```

```
    set(handles.exp_raw,'Value',0);
    set(handles.sav_raw_dec,'Value',0);
    set(handles.exp_raw_dec,'Value',0);
    set(handles.sav_f_dec,'Value',0);
    set(handles.exp_f_dec,'Value',0);
    set(handles.sav_f_b_dec,'Value',0);
    set(handles.exp_f_b_dec,'Value',0);
    set(handles.sav_mag,'Value',0);
    set(handles.sav_ctr,'Value',0);
    set(handles.sav_width,'Value',0);
    set(handles.sav_r2,'Value',0);
    set(handles.exp_mag,'Value',0);
    set(handles.exp_ctr,'Value',0);
    set(handles.exp_width,'Value',0);
    set(handles.exp_r2,'Value',0);
    set(handles.exp_fitgood,'Value',0);
    set(handles.exp_fitparams,'Value',0);
    set(handles.exp_fluor_fitgood,'Value',0);
    set(handles.exp_fluor_fitparams,'Value',0);
    set(handles.exp_fluor_mag,'Value',0);
    set(handles.exp_fluor_asym,'Value',0);
    set(handles.exp_fluor_width,'Value',0);
    set(handles.exp_fluor_r2,'Value',0);
    set(handles.sav_fluor_mag,'Value',0);
    set(handles.sav_fluor_asym,'Value',0);
    set(handles.sav_fluor_width,'Value',0);
    set(handles.sav_fluor_r2,'Value',0);
    pause(.25);
    set(handles.file_info,'String','No File');
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
    guidata(hObject,handles);
end

% --- Executes on button press in exp_mag.
function exp_mag_Callback(hObject, eventdata, handles)
% hObject    handle to exp_mag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_ctr.
function exp_ctr_Callback(hObject, eventdata, handles)
% hObject    handle to exp_ctr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_width.
function exp_width_Callback(hObject, eventdata, handles)
% hObject    handle to exp_width (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_r2.
function exp_r2_Callback(hObject, eventdata, handles)
% hObject    handle to exp_r2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_raw.
function exp_raw_Callback(hObject, eventdata, handles)
% hObject    handle to exp_raw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_f_dec.
function exp_filt_Callback(hObject, eventdata, handles)
% hObject    handle to exp_f_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)
% hObject    handle to save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %%%%%Store data as files%%%%%
    %Get filename and path
    [FileName,PathName] = uiputfile('*.txt');
    %Write image cubes to 2D arrays
    [rows cols spec] = size(handles.data);
    tmpraw = zeros(rows*cols+1,spec+2);
    tmpraw(1,1)=0;
    tmpraw(1,2)=0;
    tmpraw(1,3:end)=handles.shift_axis(1,:);
    cntc = 0;
    cntr = 1;
    for r = 1:rows
        for c = 1:cols
            cntc = cntc+1;
            cntr = cntr+1;
            tmpraw(cntr,1) = r;
            tmpraw(cntr,2) = cntc;
            for sp = 1:spec
                tmpraw(cntr,sp+2) = handles.data(r,c,sp);
            end
        end
        cntc = 0;
    end

    %run the filtered data separately since spatial filtering may have altered
    %dimensions of array
    [rows cols spec] = size(handles.data_f_dec);
    tmpfilt_dec = zeros(rows*cols+1,spec+2);
    tmpfiltb_dec = zeros(rows*cols+1,spec+2);
    tmpfilt_dec(1,1)=0;
    tmpfilt_dec(1,2)=0;
    tmpfilt_dec(1,3:end)=handles.shift_axis_dec(1,:);
    tmpfiltb_dec(1,1)=0;
```

```
tmpfiltb_dec(1,2)=0;
tmpfiltb_dec(1,3:end)=handles.shift_axis_dec(1,:);
cntc = 0;
cntr = 1;
for r = 1:rows
    for c = 1:cols
        cntc = cntc+1;
        cntr = cntr+1;
        tmpfilt_dec(cntr,1) = r;
        tmpfilt_dec(cntr,2) = cntc;
        tmpfiltb_dec(cntr,1) = r;
        tmpfiltb_dec(cntr,2) = cntc;
        for sp = 1:spec
            tmpfilt_dec(cntr,sp+2) = handles.data_f_dec(r,c,sp);
            tmpfiltb_dec(cntr,sp+2) = handles.data_f_b_dec(r,c,sp);
        end
    end
    cntc = 0;
end

[rows cols spec] = size(handles.data_dec);
tmpraw_dec = zeros(rows*cols+1,spec+2);
cntc = 0;
cntr = 1;
for r = 1:rows
    for c = 1:cols
        cntc = cntc+1;
        cntr = cntr+1;
        tmpraw_dec(cntr,1) = r;
        tmpraw_dec(cntr,2) = cntc;
        for sp = 1:spec
            tmpraw_dec(cntr,sp+2) = handles.data_dec(r,c,sp);
        end
    end
    cntc = 0;
end

%Write files to path/filename with extension
if get(handles.sav_raw,'Value') == 1
    s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
        '_Data_Raw','.txt');
    save(s,'tmpraw','-ascii');
end
if get(handles.sav_raw_dec,'Value') == 1
    s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
        '_Data_Raw_Dec','.txt');
    save(s,'tmpraw_dec','-ascii');
end
if get(handles.sav_f_dec,'Value') == 1
    s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
        '_Data_Filt_Dec','.txt');
    save(s,'tmpfilt_dec','-ascii');
end
if get(handles.sav_f_b_dec,'Value') == 1
    s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
        '_Data_Filt_Back_Dec','.txt');
    save(s,'tmpfiltb_dec','-ascii');
end
if get(handles.sav_mag,'Value') == 1
    s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
```

```
            '_Raman_Image_Magnitude','.txt');
        tmp = handles.image_mag;
        save(s,'tmp','-ascii');
    end
    if get(handles.sav_ctr,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Raman_Image_Center','.txt');
        tmp = handles.image_ctr;
        save(s,'tmp','-ascii');
    end
    if get(handles.sav_width,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Raman_Image_Width','.txt');
        tmp = handles.image_width;
        save(s,'tmp','-ascii');
    end
    if get(handles.sav_r2,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Raman_Image_RSquared','.txt');
        tmp = handles.image_r2;
        save(s,'tmp','-ascii');
    end
    if get(handles.exp_fluor_mag,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Fluorescence_Image_Magnitude','.txt');
        tmp = handles.fluor_mag;
        save(s,'tmp','-ascii');
    end
    if get(handles.exp_fluor_asym,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Fluorescence_Image_Asymmetry','.txt');
        tmp = handles.fluor_mag;
        save(s,'tmp','-ascii');
    end
    if get(handles.exp_fluor_width,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Fluorescence_Image_Width','.txt');
        tmp = handles.fluor_width;
        save(s,'tmp','-ascii');
    end
    if get(handles.exp_fluor_r2,'Value') == 1
        s = sprintf('%s%s%s%s',PathName,FileName(1:end-4),...
            '_Fluorescence_Image_RSquared','.txt');
        tmp = handles.fluor_r2;
        save(s,'tmp','-ascii');
    end
end


% --- Executes on button press in sav_mag.
function sav_mag_Callback(hObject, eventdata, handles)
% hObject    handle to sav_mag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_ctr.
function sav_ctr_Callback(hObject, eventdata, handles)
% hObject    handle to sav_ctr (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_width.
function sav_width_Callback(hObject, eventdata, handles)
% hObject    handle to sav_width (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_r2.
function sav_r2_Callback(hObject, eventdata, handles)
% hObject    handle to sav_r2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_raw.
function sav_raw_Callback(hObject, eventdata, handles)
% hObject    handle to sav_raw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_f_dec.
function sav_filt_Callback(hObject, eventdata, handles)
% hObject    handle to sav_f_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in decimate.
function decimate_Callback(hObject, eventdata, handles)
% hObject    handle to decimate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    start_shift = str2num(get(handles.lam_start,'String'));
    stop_shift = str2num(get(handles.lam_stop,'String'));
    bkg_size = str2num(get(handles.bkg_pts,'String'));

    %determine indices of start and stop
    tmp = abs(handles.shift_axis-start_shift);
    [min_val min_index] = min(tmp); %value and index of closest value
    start = min_index; %start index
    clear tmp;
    tmp = abs(handles.shift_axis-stop_shift);
    [min_val min_index] = min(tmp); %value and index of closest value
    stop = min_index; %stop index
    %add the background on either side of the fit to remove
    handles.bkg1 = start - bkg_size;
    handles.bkg2 = stop + bkg_size;

    handles.data_dec = handles.data(:,:,handles.bkg1:handles.bkg2);
    handles.shift_axis_dec = handles.shift_axis(1,handles.bkg1:handles.bkg2);
```

```
    set(handles.decimate,'BackgroundColor',handles.lt_rd);
    s = sprintf('Spec: %.0f',handles.bkg2-handles.bkg1+1);
    set(handles.spec_label,'String',s);
    set(handles.sav_raw_dec,'Value',1);
    set(handles.exp_raw_dec,'Value',1);
    guidata(hObject,handles);
end

% --- Executes on button press in average.
function average_Callback(hObject, eventdata, handles)
% hObject    handle to average (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    %Remove noise
    if get(handles.ave_none,'Value')==1
        handles.label_spectral = 'None';
        handles.data_f = handles.data;
        handles.data_f_dec = handles.data_dec;
    elseif get(handles.ave_box,'Value')==1
        handles.label_spectral = 'Boxcar';
        box_size = str2num(get(handles.box_win,'String'));
        handles.data_f = filter(ones(1,box_size)/box_size,1,handles.data,[],3);
        handles.data_f_dec = handles.data_f(:,:,handles.bkg1:handles.bkg2);
    elseif get(handles.ave_sg,'Value')==1
        handles.label_spectral = sprintf('Sovitzky-Golay Ord: %s Win: %s',...
            get(handles.sg_order,'String'),get(handles.sg_win,'String'));
        order = str2num(get(handles.sg_order,'String'));
        box_size = str2num(get(handles.sg_win,'String'));
        handles.data_f = sgolayfilt_3D(handles.data,order,box_size,[],3);
        handles.data_f_dec = handles.data_f(:,:,handles.bkg1:handles.bkg2);
    end
    if get(handles.ave_spatial,'Value')==1
        handles.data_f_dec = spatial_filter(handles.data_f_dec,handles.spatial_array);
        handles.data_f = spatial_filter(handles.data_f,handles.spatial_array);
    else
        handles.label_spatial = 'No Spatial';
    end
    set(handles.average,'BackgroundColor',handles.lt_rd);
    set(handles.sav_f_dec,'Value',1);
    set(handles.exp_f_dec,'Value',1);
    guidata(hObject,handles);
end


% --- Executes on button press in process.
function process_Callback(hObject, eventdata, handles)
% hObject    handle to process (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Set limits for determining poor fit
    mag_lim = 0;
    width_lim = 1;
    r2_lim = 0.2;

    %Remove background and fit the data
    if get(handles.bkg_detrend,'Value')==1
        handles.label_bkg = 'Detrend';
        [handles.image_mag,handles.image_ctr,handles.image_width,handles.image_r2,...
            handles.data_f_b_dec,handles.fit_array,handles.goodness_array] =...
```

```
        process_detrend(handles.data_f_dec,handles.shift_axis_dec',...
        str2num(get(handles.bkg_pts,'String')),mag_lim,width_lim,r2_lim,...
        get(handles.gauss_fit,'Value'));

else get(handles.bkg_poly,'Value')==1
    handles.label_bkg = 'Poly';
    if get(handles.poly_fit,'Value')==1
        poly = 'poly1';
    elseif get(handles.poly_fit,'Value')==2
        poly = 'poly2';
    end
    [handles.image_mag,handles.image_ctr,handles.image_width,handles.image_r2,...
        handles.data_f_b_dec,handles.fit_array,handles.goodness_array] =...
        process_poly(handles.data_f_dec,handles.shift_axis_dec',...
        str2num(get(handles.bkg_pts,'String')),mag_lim,width_lim,r2_lim,poly,...
        get(handles.gauss_fit,'Value'));
end
if get(handles.gauss_fit,'Value')==1
    handles.label_fit = 'Gaussian';
else
    handles.label_fit = 'Lorentzian';
end
figure('Name',get(handles.file_display,'String'));
subplot(2,2,1);
X = -5:5/8:5;
Y = 5:-5/8:-5;
imagesc(X, Y, handles.image_mag/max(max(handles.image_mag)));
colormap('jet');
caxis([0,1]);
colorbar;
title('Normalized Magnitude');
xlabel('X (\mum)');
ylabel('Y (\mum)');
text(min(X)-3.5, min(Y),'(b)','FontWeight','bold','FontSize',16,'Color','k',...
    'BackgroundColor','w');
fprintf('Avg Mag: %.3f\n', mean(handles.image_mag(:)));
subplot(2,2,2);
imagesc(X, Y, handles.image_ctr);
caxis([str2num(get(handles.lam_start,'String')),...
    str2num(get(handles.lam_stop,'String'))]);
colorbar;
title('Center');
xlabel('X (\mum)');
ylabel('Y (\mum)');
text(min(X)-3.5, min(Y),'(d)','FontWeight','bold','FontSize',16,'Color','k',...
    'BackgroundColor','w');
fprintf('Avg Ctr: %.3f\n', mean(handles.image_ctr(:)));
subplot(2,2,3);
imagesc(X, Y, handles.image_width);
caxis([0,(str2num(get(handles.lam_stop,'String'))-...
    str2num(get(handles.lam_start,'String')))/2]);
colorbar;
title('Width');
xlabel('X (\mum)');
ylabel('Y (\mum)');
text(min(X)-3.5, min(Y),'(c)','FontWeight','bold','FontSize',16,'Color','k',...
    'BackgroundColor','w');
fprintf('Avg Width: %.3f\n', mean(handles.image_width(:)));
subplot(2,2,4);
imagesc(X, Y, handles.image_r2);
```

```
    caxis([0,1]);
    colorbar;
    title('R^{2}');
    xlabel('X (\mum)');
    ylabel('Y (\mum)');
    text(min(X)-3.5, min(Y),'(e)','FontWeight','bold','FontSize',16,'Color','k',...
        'BackgroundColor','w');
    fprintf('Avg R-Squared: %.3f\n', mean(handles.image_r2(:)));
    suptitle(sprintf(['Raman Data: Spectral Ave: %s\nSpatial Ave: %s,',...
        ' Background: %s, Fit: %s, Range: %.0f - %.0f cm^{-1}'],...
        handles.label_spectral,handles.label_spatial,handles.label_bkg,...
        handles.label_fit,str2num(get(handles.lam_start,'String')),...
        str2num(get(handles.lam_stop,'String'))));
    set(handles.process,'BackgroundColor',handles.lt_rd);
    set(handles.sav_f_b_dec,'Value',1);
    set(handles.exp_f_b_dec,'Value',1);
    set(handles.sav_mag,'Value',1);
    set(handles.sav_ctr,'Value',1);
    set(handles.sav_width,'Value',1);
    set(handles.sav_r2,'Value',1);
    set(handles.exp_mag,'Value',1);
    set(handles.exp_ctr,'Value',1);
    set(handles.exp_width,'Value',1);
    set(handles.exp_r2,'Value',1);
    set(handles.exp_fitgood,'Value',1);
    set(handles.exp_fitparams,'Value',1);
    guidata(hObject,handles);
end

% --- Executes on button press in fluor_fit.
function fluor_fit_Callback(hObject, eventdata, handles)
% hObject    handle to fluor_fit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Set limits for determining poor fit
    mag_lim = 0;
    r2_lim = 0.2;

    [handles.fluor_mag,handles.fluor_asym,handles.fluor_width,handles.fluor_r2,...
        handles.fluor_fit_array,handles.fluor_goodness_array] =...
        process_fluor(handles.data_f,handles.shift_axis,mag_lim,r2_lim);
    figure('Name',get(handles.file_display,'String'));
    X = -5:5/8:5;
    Y = 5:-5/8:-5;
    colormap('jet');
    subplot(2,2,1);
    imagesc(X, Y, handles.fluor_mag/max(max(handles.fluor_mag)));
    caxis([0,1]);
    colorbar;
    title('Normalized Fluorescence');
    xlabel('X (\mum)');
    ylabel('Y (\mum)');
    subplot(2,2,2);
    imagesc(X, Y, handles.fluor_asym);
    colorbar;
    title('Asymmetry');
    xlabel('X (\mum)');
    ylabel('Y (\mum)');
    subplot(2,2,3);
```

```
    imagesc(X, Y, handles.fluor_width);
    colorbar;
    title('Width');
    xlabel('X (\mum)');
    ylabel('Y (\mum)');
    subplot(2,2,4);
    imagesc(X, Y, handles.fluor_r2);
    caxis([0,1]);
    colorbar;
    title('R^{2}');
    xlabel('X (\mum)');
    ylabel('Y (\mum)');
    suptitle(sprintf(['Fluorescence Data: Spectral Ave: %s\nSpatial Ave: %s,',...
        ' Fit: Lognormal'],handles.label_spectral,handles.label_spatial));
    set(handles.process,'BackgroundColor',handles.lt_rd);
    set(handles.exp_fluor_fitgood,'Value',1);
    set(handles.exp_fluor_fitparams,'Value',1);
    set(handles.exp_fluor_mag,'Value',1);
    set(handles.exp_fluor_asym,'Value',1);
    set(handles.exp_fluor_width,'Value',1);
    set(handles.exp_fluor_r2,'Value',1);
    set(handles.sav_fluor_mag,'Value',1);
    set(handles.sav_fluor_asym,'Value',1);
    set(handles.sav_fluor_width,'Value',1);
    set(handles.sav_fluor_r2,'Value',1);
    guidata(hObject,handles);
end


function file_display_Callback(hObject, eventdata, handles)
% hObject    handle to file_display (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function file_display_CreateFcn(hObject, eventdata, handles)
% hObject    handle to file_display (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end


% --- Executes on button press in file_load.
function file_load_Callback(hObject, eventdata, handles)
% hObject    handle to file_load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    [FileName,PathName,FilterIndex] = uigetfile('*.txt');
    set(handles.file_display,'String',FileName);
    handles.file = sprintf('%s%s',PathName,FileName);
    A = importdata(handles.file,'\t');
    %Determine the number of row_label and columns - assume row 1 is Raman shift
    % row 2 begins data, col 1 is row_label and col 2 is columns
```

```
    handles.cols = length(find(A(:,1)==A(2,1))); %find # cols in first row
    Adim = size(A);
    handles.rows = (Adim(1)-1)/handles.cols; %ignore first row and calc # of rows
    % put data in an image cube with (row,column,spectrum)
    handles.data =...
        zeros(handles.rows,handles.cols,Adim(2)-2); %spectra ignore first two cols
    handles.shift_axis = A(1,3:end); %establish the Raman shift axis
    for r = 1:handles.rows
        for c = 1:handles.cols
            handles.data(r,c,:) = A((r-1)*handles.cols+c+1,3:end);
        end
    end
    s = sprintf('Rows: %.0f',handles.rows);
    set(handles.row_label,'String',s);
    s = sprintf('Cols: %.0f',handles.cols);
    set(handles.col_label,'String',s);
    s = sprintf('Spec: %.0f',Adim(2)-2);
    set(handles.spec_label,'String',s);
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'String','File Loaded');
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
    set(handles.file_load,'BackgroundColor',handles.lt_bl);
    handles.input_plot;
    for r = 1:handles.rows
        for c = 1:handles.cols
            plot(handles.shift_axis(1,:),squeeze(handles.data(r,c,:)));
            hold on;
        end
    end
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
    set(handles.file_info,'String','File Loaded');
    set(handles.file_info,'String','Raw Data');
    set(handles.file_plot,'BackgroundColor',handles.lt_bl);
    set(handles.sav_raw,'Value',1);
    set(handles.exp_raw,'Value',1);
    guidata(hObject,handles);
end


% --- Executes on button press in file_plot.
function file_plot_Callback(hObject, eventdata, handles)
% hObject    handle to file_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    handles.input_plot;
    for r = 1:handles.rows
        for c = 1:handles.cols
            plot(handles.shift_axis(1,:),squeeze(handles.data(r,c,:)));
            hold on;
        end
    end
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
    set(handles.file_info,'String','File Loaded');
    set(handles.file_info,'String','Raw Data');
end
```

```
function lam_start_Callback(hObject, eventdata, handles)
% hObject    handle to lam_start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function lam_start_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lam_start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end


function lam_stop_Callback(hObject, eventdata, handles)
% hObject    handle to lam_stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function lam_stop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lam_stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end



function bkg_pts_Callback(hObject, eventdata, handles)
% hObject    handle to bkg_pts (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function bkg_pts_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bkg_pts (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

```
end


% --- Executes on selection change in poly_fit.
function poly_fit_Callback(hObject, eventdata, handles)
% hObject    handle to poly_fit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function poly_fit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to poly_fit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end


function sg_order_Callback(hObject, eventdata, handles)
% hObject    handle to sg_order (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function sg_order_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sg_order (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end


function sg_win_Callback(hObject, eventdata, handles)
% hObject    handle to sg_win (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function sg_win_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sg_win (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
```

```
    end
end


function box_win_Callback(hObject, eventdata, handles)
% hObject    handle to box_win (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function box_win_CreateFcn(hObject, eventdata, handles)
% hObject    handle to box_win (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end


% --- Executes on button press in exp_raw_dec.
function exp_raw_dec_Callback(hObject, eventdata, handles)
% hObject    handle to exp_raw_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_f_dec.
function exp_f_dec_Callback(hObject, eventdata, handles)
% hObject    handle to exp_f_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_f_b_dec.
function exp_f_b_dec_Callback(hObject, eventdata, handles)
% hObject    handle to exp_f_b_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_raw_dec.
function sav_raw_dec_Callback(hObject, eventdata, handles)
% hObject    handle to sav_raw_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_f_dec.
function sav_f_dec_Callback(hObject, eventdata, handles)
% hObject    handle to sav_f_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_f_b_dec.
function sav_f_b_dec_Callback(hObject, eventdata, handles)
% hObject    handle to sav_f_b_dec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in roi_plot.
function roi_plot_Callback(hObject, eventdata, handles)
% hObject    handle to roi_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    cla(handles.input_plot);
    axis auto;
    xlabel('Raman Shift (cm^{-1})');
    ylabel('Signal');
    title('Input Data');
    [rows cols spec] = size(handles.data_dec);
    for r = 1:rows
        for c = 1:cols
            plot(handles.shift_axis_dec(1,:),squeeze(handles.data_dec(r,c,:)));
            hold on;
        end
    end
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'String','ROI');
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
end


% --- Executes on button press in proc_plot.
function proc_plot_Callback(hObject, eventdata, handles)
% hObject    handle to proc_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    cla(handles.input_plot);
    axis auto;
    xlabel('Raman Shift (cm^{-1})');
    ylabel('Signal');
    title('Input Data');
    [rows cols spec] = size(handles.data_f_b_dec);
    for r = 1:rows
        for c = 1:cols
            plot(handles.shift_axis_dec(1,:),squeeze(handles.data_f_b_dec(r,c,:)));
            hold on;
        end
    end
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'String','Processed');
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
end
```

```
% --- Executes on button press in ave_plot.
function ave_plot_Callback(hObject, eventdata, handles)
% hObject    handle to ave_plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    cla(handles.input_plot);
    axis auto;
    xlabel('Raman Shift (cm^{-1})');
    ylabel('Signal');
    title('Input Data');
    [rows cols spec] = size(handles.data_f_dec);
    for r = 1:rows
        for c = 1:cols
            plot(handles.shift_axis_dec(1,:),squeeze(handles.data_f_dec(r,c,:)));
            hold on;
        end
    end
    set(handles.file_info,'BackgroundColor',[.94,.94,.94]);
    pause(.25);
    set(handles.file_info,'String','Averaged');
    set(handles.file_info,'BackgroundColor',[.8,.8,.8]);
end


% --- Executes on button press in ave_spatial.
function ave_spatial_Callback(hObject, eventdata, handles)
% hObject    handle to ave_spatial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_fitparams.
function exp_fitparams_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fitparams (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_fitgood.
function exp_fitgood_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fitgood (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on selection change in ave_spatial_array.
function ave_spatial_array_Callback(hObject, eventdata, handles)
% hObject    handle to ave_spatial_array (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    switch(get(handles.ave_spatial_array,'Value'))
        case 1
            handles.spatial_array = [0,.2,0;.2,.2,.2;0,.2,0];
            handles.label_spatial = 'NN Uniform';
        case 2
            handles.spatial_array = [0,.15,0;.15,.4,.15;0,.15,0];
```

```
            handles.label_spatial = 'NN .40, 4x.15';
        case 3
            handles.spatial_array = [0,.10,0;.10,.6,.10;0,.10,0];
            handles.label_spatial = 'NN .60, 4x.10';
        case 4
            handles.spatial_array = [0,.05,0;.05,.8,.05;0,.05,0];
            handles.label_spatial = 'NN .80, 4x.05';
        case 5
            handles.spatial_array = (1/9)*ones(3,3);
            handles.label_spatial = 'NNN Uniform';
        case 6
            handles.spatial_array = [.1,.1,.1;.1,.2,.1;.1,.1,.1];
            handles.label_spatial = 'NNN .20, 8x.10';
        case 7
            handles.spatial_array = [.05,.15,.05;.15,.2,.15;.05,.15,.05];
            handles.label_spatial = 'NNN .20, 4x.15, 4x.05';
        case 8
            handles.spatial_array = [.05,.10,.05;.10,.4,.10;.05,.10,.05];
            handles.label_spatial = 'NNN .40, 4x.10, 4x.05';
        case 9
            handles.spatial_array = [.05,.05,.05;.05,.6,.05;.05,.05,.05];
            handles.label_spatial = 'NNN .60, 8x.05';
        case 10
            handles.spatial_array = [.025,.075,.025;.075,.6,.075;.025,.075,.025];
            handles.label_spatial = 'NNN .60, 4x.075, 4x.025';
        case 11
            handles.spatial_array = [.025,.025,.025;.025,.8,.025;.025,.025,.025];
            handles.label_spatial = 'NNN .80, 8x.025';
        otherwise
            handles.spatial_array = zeros(3,3);
            handles.label_spatial = 'Error: all zeros';
    end
    guidata(hObject,handles);
end


% --- Executes during object creation, after setting all properties.
function ave_spatial_array_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ave_spatial_array (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'), ...
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    handles.spatial_array = [0,.2,0;.2,.2,.2;0,.2,0];
    handles.label_spatial = 'NN Uniform';
    guidata(hObject,handles);
end


% --- Executes on button press in exp_fluor_fitparams.
function exp_fluor_fitparams_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_fitparams (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end
```

```
% --- Executes on button press in exp_fluor_fitgood.
function exp_fluor_fitgood_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_fitgood (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function average_CreateFcn(hObject, eventdata, handles)
% hObject    handle to average (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes on button press in exp_fluor_mag.
function exp_fluor_mag_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_mag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_fluor_asym.
function exp_fluor_asym_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_asym (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_fluor_width.
function exp_fluor_width_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_width (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in exp_fluor_r2.
function exp_fluor_r2_Callback(hObject, eventdata, handles)
% hObject    handle to exp_fluor_r2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_fluor_mag.
function sav_fluor_mag_Callback(hObject, eventdata, handles)
% hObject    handle to sav_fluor_mag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_fluor_asym.
function sav_fluor_asym_Callback(hObject, eventdata, handles)
% hObject    handle to sav_fluor_asym (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_fluor_width.
function sav_fluor_width_Callback(hObject, eventdata, handles)
% hObject    handle to sav_fluor_width (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes on button press in sav_fluor_r2.
function sav_fluor_r2_Callback(hObject, eventdata, handles)
% hObject    handle to sav_fluor_r2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


function [ arrayout ] = spatial_filter( arrayin,filter_template )
%SPATIAL_FILTER Spatial averages an array by Nearest Neighbor (NN) or Next-
%Nearest Neighbor (NNN)
%   filter in is a 3x3 array showing the weight for each plane. Each plane
%   in arrayin should be multipled by filter and summed.
%   output array will be reduced by 2 rows and 2 columns

%   Author(s): A. Fisher, K. Meissner
%   Centre of NanoHealth, College of Science,
%   Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

    [rows,cols,spec] = size(arrayin);
    arrayout = zeros(rows-2,cols-2,spec);
    filtmat = repmat(filter_template,1,1,spec);
    for r = 2:rows-1
        for c = 2:cols-1
            tmp = arrayin(r-1:r+1,c-1:c+1,:).*filtmat;
            arrayout(r-1,c-1,:) = sum(sum(tmp,1),2);
        end
    end
end


function y=sgolayfilt_3D(x,order,framelen,weights,dim)
%SGOLAYFILT Savitzky-Golay Filtering.
%   SGOLAYFILT(X,ORDER,FRAMELEN) smoothes the signal X using a
%   Savitzky-Golay (polynomial) smoothing filter.  The polynomial order,
%   ORDER, must be less than the frame length, FRAMELEN, and FRAMELEN must
%   be odd.  The length of the input X must be >= FRAMELEN.  If X is a
%   matrix, the filtering is done on the columns of X.
%
%   Note that if the polynomial order ORDER equals FRAMELEN-1, no smoothing
%   will occur.
%
%   SGOLAYFILT(X,ORDER,FRAMELEN,WEIGHTS) specifies a weighting vector
%   WEIGHTS with length FRAMELEN containing real, positive valued weights
%   employed during the least-squares minimization. If not specified, or if
%   specified as empty, WEIGHTS defaults to an identity matrix.
%
```

```
%   SGOLAYFILT(X,ORDER,FRAMELEN,[],DIM) and
%   SGOLAYFILT(X,ORDER,FRAMELEN,WEIGHTS,DIM) operate along the dimension DIM.
%
%   % Example:
%   %   Smooth the mtlb signal by applying a cubic Savitzky-Golay filter
%   %   to data frames of length 41.
%   load mtlb                       % Load data
%   smtlb = sgolayfilt(mtlb,3,41);   % Apply 3rd-order filter
%   plot([mtlb smtlb]);
%   legend('Original Data','Filtered Data');
%
%   See also SGOLAY, MEDFILT1, FILTER

%   References:
%     [1] Sophocles J. Orfanidis, INTRODUCTION TO SIGNAL PROCESSING,
%              Prentice-Hall, 1995, Chapter 8.

%   Author(s): R. Losada
%   Copyright 1988-2016 The MathWorks, Inc.

    narginchk(3,5);

    % Check if the input arguments are valid
    if round(framelen) ~= framelen
        error(message('signal:sgolayfilt:MustBeIntegerFrameLength'))
    end
    if rem(framelen,2) ~= 1
        error(message('signal:sgolayfilt:SignalErr'))
    end
    if round(order) ~= order
        error(message('signal:sgolayfilt:MustBeIntegerPolyDegree'))
    end
    if order > framelen-1
        error(message('signal:sgolayfilt:InvalidRangeDegree'))
    end

    if nargin < 4
       weights = [];
    elseif ~isempty(weights)
       % Check for right length of WEIGHTS
       if length(weights) ~= framelen
           error(message('signal:sgolayfilt:InvalidDimensionsWeight'))
       end
       % Check to see if all elements are positive
       if min(weights) <= 0
           error(message('signal:sgolayfilt:InvalidRangeWeight'))
       end
    end

    if nargin < 5, dim = []; end

    % Check the input data type. Single precision is not supported.
    %chkinputdatatype(x,order,framelen,weights,dim);

    % Compute the projection matrix B
    B = sgolay(order,framelen,weights);

    if ~isempty(dim) && dim > ndims(x)
        error(message('signal:sgolayfilt:InvalidDimensionsInput', 'X'))
    end
```

```
    % Reshape X into the right dimension.
    if isempty(dim)
        % Work along the first non-singleton dimension
        [x, nshifts] = shiftdim(x);
    else
        % Put DIM in the first dimension (this matches the order
        % that the built-in filter function uses)
        perm = [dim,1:dim-1,dim+1:ndims(x)];
        x = permute(x,perm);
    end

    if size(x,1) < framelen
        error(message('signal:sgolayfilt:InvalidDimensionsTooSmall'))
    end
    for i = 1:length(x(1,1,:))
            % Compute the transient on
            ybegin = B(end:-1:(framelen-1)/2+2,:) * x(framelen:-1:1,:,i);

            % Compute the steady state output
            ycenter = filter(B((framelen-1)./2+1,:), 1, x(:,:,i));

            % Compute the transient off
            yend = B((framelen-1)/2:-1:1,:) * x(end:-1:end-(framelen-1),:,i);

            % Concatenate
            y(:,:,i) = [ybegin; ycenter(framelen:end,:); yend];
    end

    % Convert Y to the original shape of X
    if isempty(dim)
        y = shiftdim(y, -nshifts);
    else
        y = ipermute(y,perm);
    end
end


function [imag,ictr,iwidth,ir2,dout,foutf,foutg ] = ...
    process_detrend( d,x,b_size,ml,wl,r2l,f)
%DATA_DETREND remove any background drift and fit Raman peaks
%   background drift removed with MATLAB's detrend function.
%   raman peaks fit to user's choice of gaussian or lorentzian functions.

%   Author(s): A. Fisher, K. Meissner
%   Centre of NanoHealth, College of Science,
%   Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

    [rmax, cmax, spec] = size(d);
    shift = x;
    dout = zeros(rmax,cmax,spec);
    foutf = cell(rmax,cmax);
    foutg = cell(rmax,cmax);
    if f ~= 1 %1 means Gaussian fit, otherwise Lorentzian
        lzt = fittype( @(a1,b1,c1,x)...
            (a1)*((c1^2)./(((x-b1).^2)+c1^2 ))); % create Lorentzian fit
    end
    wb = waitbar(0,'Processing Spectra...');
    for r = 1:rmax
        for c = 1:cmax
```

```
            data_sh = squeeze(d(r,c,:));

            %use detrend to remove background
            data_sh_detr = detrend(data_sh);
            data_sh_detr = data_sh_detr - mean(data_sh_detr(1:b_size));
            dout(r,c,:) = data_sh_detr;
            if f == 1
                [F1,G1] = fit(shift,data_sh_detr,'Gauss1');
            else
                [F1,G1] = fit(shift,data_sh_detr,lzt,'StartPoint',...
                    [max(data_sh_detr)/2,mean(shift),shift(2)-shift(1)]);
            end
            foutf{r,c} = F1;
            foutg{r,c} = G1;
            if F1.a1 <= ml
                imag(r,c) = 0; %if mag < 0, make all values 0 = bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            elseif F1.c1 < wl
                imag(r,c) = 0; %if width too small, bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            elseif G1.rsquare < r2l
                imag(r,c) = 0; %if rsquared too small, bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            else
                imag(r,c) = F1.a1; %make an image from the magnitude of fit
                ictr(r,c) = F1.b1; %make an image from the center of fit
                iwidth(r,c) = F1.c1; %make an image from the width of fit
                ir2(r,c) = G1.rsquare; %make an image from the r^2 of fit
            end

        end
        waitbar(r/rmax);
    end
    close(wb);
end


function [ imag,ictr,iwidth,ir2,dout,foutf,foutg  ] = ...
    process_poly(d,x,b_size,ml,wl,r2l,p,f )
%DATA_POLY remove any background drift and fit Raman peaks
%   background drift removed by subtracting a polynomial fit of the first
%   b_size points.
%   raman peaks fit to user's choice of gaussian or lorentzian functions.

%   Author(s): A. Fisher, K. Meissner
%   Centre of NanoHealth, College of Science,
%   Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

    [rmax, cmax, spec] = size(d);
    shift = x;
    dout = zeros(rmax,cmax,spec);
    foutf = cell(rmax,cmax);
    foutg = cell(rmax,cmax);
    if f ~= 1 %1 means Gaussian fit, otherwise Lorentzian
```

```
        lzt = fittype( @(a1,b1,c1,x) ...
            (a1)*((c1^2)./(((x-b1).^2)+c1^2 ))); % create Lorentzian fit
    end
    wb = waitbar(0,'Processing Spectra...');
    for r = 1:rmax
        for c = 1:cmax
            data_sh = squeeze(d(r,c,:));

            %use polynomial to remove background
            num = b_size*2; %number of background points
            data_bkgnd = zeros(num,2);
            data_bkgnd(:,1) = [shift(1:b_size);shift(end-b_size+1:end)];
            data_bkgnd(:,2) = [data_sh(1:b_size);data_sh(end-b_size+1:end)];
            [F2,~] = fit(data_bkgnd(:,1),data_bkgnd(:,2),p);
            data_sh_poly(:,1) = data_sh;
            background(:,1) = F2(shift);
            data_sh_poly(:,2) = data_sh_poly(:,1) - background(:,1);
            dout(r,c,:) = data_sh_poly(:,2);
            tmp = sort(d,'descend');
            fit_max = tmp(3)*2;
            if f == 1
                [F3,G3] = fit(shift,data_sh_poly(:,2),'Gauss1',...
                    'Lower',[0,min(shift),eps],...
                    'Upper',[fit_max,max(shift),max(shift)-min(shift)]);
            else
                [F3,G3] = fit(shift,data_sh_poly(:,2),lzt,...
                    'StartPoint',[max(data_sh_poly(:,2))/2,mean(shift),...
                    shift(2)-shift(1)],'Lower',[0,min(shift),eps],'Upper',...
                    [fit_max,max(shift),max(shift)-min(shift)]);
            end
            data_sh_poly(:,3) = F3(shift);
            foutf{r,c} = F3;
            foutg{r,c} = G3;
            if F3.a1 <= ml
                imag(r,c) = 0; %if mag < 0, make all values 0 = bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            elseif F3.c1 < wl
                imag(r,c) = 0; %if width too large, make all values 0 = bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            elseif G3.rsquare < r2l
                imag(r,c) = 0; %if r^2 too small, make all values 0 = bad fit
                ictr(r,c) = 0;
                iwidth(r,c) = 0;
                ir2(r,c) = 0;
            else
                imag(r,c) = F3.a1; %make an image from the magnitude of fit
                ictr(r,c) = F3.b1; %make an image from the center of fit
                iwidth(r,c) = F3.c1; %make an image from the width of fit
                ir2(r,c) = G3.rsquare; %make an image from the r^2 of fit
            end

        end
        waitbar(r/rmax);
    end
    close(wb);
end
```

```
function [ imag,iasym,iwidth,ir2,foutf,foutg ] = process_fluor(d,x,ml,r2l)
%PROCESS_FLUOR Fits the fluorescence data
%   fits to a lognormal function

%   Author(s): A. Fisher, K. Meissner
%   Centre of NanoHealth, College of Science,
%   Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

    lognorm2 = fittype(@(a,b,c,I0,offset,x) offset+(I0*b./(x-a)).*exp(-c^2).* ...
        exp(-(1/(2*c^2))*(log((x-a)./b)).^2));
    [rmax, cmax, ~] = size(d);
    foutf = cell(rmax,cmax);
    foutg = cell(rmax,cmax);
    wb = waitbar(0,'Processing Spectra...');
        for r = 1:rmax
            for c = 1:cmax
                [F, G] = fit(x',squeeze(d(r,c,:)),lognorm2,'StartPoint',...
                    [mean([min(x)-eps -max(x)]),500,.5,...
                    max(d(r,c,:))-mean(d(r,c,end-20:end)),mean(d(r,c,end-20:end))],...
                    'Upper',[min(x)-eps,2000,10,max(d(r,c,:)),max(d(r,c,:))],...
                    'Lower',[-max(x),eps,eps,0,0]);

                foutf{r,c} = F;
                foutg{r,c} = G;

                if F.I0 <= ml
                    imag(r,c) = 0; %if mag < 0, make all values 0 = bad fit
                    iasym(r,c) = 0;
                    iwidth(r,c) = 0;
                    ir2(r,c) = 0;
                elseif G.rsquare < r2l
                    imag(r,c) = 0; %if r^2 too small, make all values 0 = bad fit
                    iasym(r,c) = 0;
                    iwidth(r,c) = 0;
                    ir2(r,c) = 0;
                else
                    imag(r,c) = F.I0; %make an image from the magnitude of fit
                    iasym(r,c) = F.c; %make an image from the asymmetry of fit
                    iwidth(r,c) = F.b; %make an image from the width of fit
                    ir2(r,c) = G.rsquare;
                end
            end
            waitbar(r/rmax);
        end
    close(wb)
    % figure;
    %
    % subplot(2,2,1);
    % imagesc(imag/max(max(imag)));
    % caxis([0,1]);
    % colorbar;
    % title('Normalized Magnitude');
    % subplot(2,2,2);
    % imagesc(iasym);
    % colorbar;
    % title('Assymetry');
    % subplot(2,2,3);
```

```
    % imagesc(iwidth);
    % colorbar;
    % title('Width');
    % subplot(2,2,4);
    % imagesc(ir2);
    % caxis([0,1]);
    % colorbar;
    % title('R Squared');
end
```