Appendix A

MATLAB Code for 3-D Particles Analysis:

Lines marked with the % are comments in the code, and are not required for running the program, but should be left in place in order help users understand the various components. The function depends on a few other MATLAB files, including "getstacks," "timebar" (written and provided by Chad English), and "dlview" (written and provided by David Liebowitz). All the dependent functions are also shown below. In order for the program to run properly, all the functions must be copied into a text editor and then saved as a .m file into a directory on the MATLAB path. The files should be saved according to the function name, such as "Particles3.m" and "dlview.m". This version has been tested on MATLAB version 6, releases 12, 13, and 13.1.

```
function varargout = Particles3(varargin)
% Particles3 - graphical interface for running 3D particle analysis.
%
% Calling Particles3 opens a new window with two axes displayed in 3 dimensions.
% To load images (in TIFF format only), click the "Load 3D Images" button.
% If you need to use some other image format, you can modify the GETSTACKS
% function, which is called for opening the image stacks.  Once images have
% been loaded, isosurfaces of each image are plotted in the axes.  The initial
% isovalue is set to the maximum value of GRAYTHRESH for each image.  You
% can adjust the threshold settings using the sliders beneath each axis,
% or by entering threshold values (between zero and one) in the text boxes
% to the left of each slider.
%
% Left-click and drag on each axis to rotate the image in 3D.
% Right-click and drag on each axis to zoom in and out of the image.
% Middle-click and drag to pan left or right in the image.
%
% The idea is to set a threshold level that will isolate the particles in
% your image data without losing any 'real' information.  Of course what is
% real is very subjective, so you may wish to set several thresholds for any
% given analysis.  You may then run your image processing routine with the
% various threshold settings to see if any effect you observe is due to an
% artifact of your threshold settings, or if it is truly something real in
% your data.  An intensity-coded 2D version of the images will be plotted
% into a new window if the 2D Projections button is pressed.
%
% Once a threshold is set, clicking the "Save and Run" button will complete
% the analysis and plot the 3D render of the overlapping particles in a new
% figure.  Depending on your data, you may want to only plot the particles
% that colocalize, or you may choose to plot all of one image or the other.
% These selections can be made with the Colocalization Plots radio buttons.
% If you do not want to plot the data, uncheck the "Colocalization Only"
% button, which is checked by default.
%
% If you have a slow video card, you can use the Reducepatch command by
```

```
% entering a value less than one in the Reducepatch Ratio text box.  This
% will increase the computation time required to render the surface, but
% will make the real-time 3D rotation much faster.  By default, the ratio
% is set to 1, which renders all surfaces.  Changing the value to something
% less than 1 will reduce the number of surfaces rendered.  For example,
% entering 0.2 will result in rendering 20% of the total number of surfaces
% for the image.
%
% See Also IMREAD, GETSTACKS, ISOSURFACE, REDUCEPATCH, GRAYTHRESH, DLVIEW

% Last Modified by GUIDE v2.5 03-Apr-2004 22:25:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Particles3_OpeningFcn, ...
                   'gui_OutputFcn',  @Particles3_OutputFcn, ...
                   'gui_LayoutFcn',  @Particles3_LayoutFcn, ...
                   'gui_Callback',   []);
if nargin & isstr(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


% --- Executes just before Particles3 is made visible.
function Particles3_OpeningFcn(hObject, eventdata, handles, varargin)
set(gcf,'Color',[.3 .3 .3]);
set(gcf,'toolbar','none');
%s = wgetname(gcf);
%seticon(s,'C:\MATLAB6p5\p3icon.ico');

% Choose default command line output for Particles3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Particles3_OutputFcn(hObject, eventdata, handles)

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

% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
dlview on;
```

```
% --- Plots isosurface of image1
function surf1
handles = guidata(gcbo);
[r c z] = size(handles.img1);
if str2double(get(handles.edit3,'String')) == 1;
   axes(handles.img1plot); cla;
   p1 = patch(isosurface(handles.img1,get(handles.slider1,'Value'))); axis([0 c 0 r 0 z]);
   set(p1,'FaceColor',[.4 .6 .7],'FaceAlpha',1,'EdgeColor',[.4 .6 .7],'EdgeAlpha',.01);
   axis vis3d; light; light; light('Position', [-1 -1 -3]); lighting gouraud;
   set(gca,'Color','black');
else str2double(get(handles.edit3,'String')) ~= 1;
   axes(handles.img1plot); cla;
   p1 = patch(isosurface(handles.img1,get(handles.slider1,'Value'))); axis([0 c 0 r 0 z]);
   reducepatch(p1, str2double(get(handles.edit3,'String')));
   set(p1,'FaceColor',[.4 .6 .7],'FaceAlpha',1,'EdgeColor',[.4 .6 .7],'EdgeAlpha',.01);
   axis vis3d; light; light; light('Position', [-1 -1 -3]); lighting gouraud;
   set(gca,'Color','black');
end

% --- Plots isosurface of image2
function surf2
handles = guidata(gcbo);
[r c z] = size(handles.img2);
if str2double(get(handles.edit3,'String')) == 1;
   axes(handles.img2plot); cla;
   p2 = patch(isosurface(handles.img2,get(handles.slider2,'Value'))); axis([0 c 0 r 0 z]);
   set(p2,'FaceColor',[.7 .2 .4],'FaceAlpha',1,'EdgeColor',[.7 .2 .4],'EdgeAlpha',.01);
   axis vis3d; light; light('Position', [-1 -1 -3]); lighting gouraud;
else str2double(get(handles.edit3,'String')) ~= 1;
   axes(handles.img2plot); cla;
   p2 = patch(isosurface(handles.img2,get(handles.slider2,'Value'))); axis([0 c 0 r 0 z]);
   reducepatch(p2, str2double(get(handles.edit3,'String')));
   set(p2,'FaceColor',[.7 .2 .4],'FaceAlpha',1,'EdgeColor',[.7 .2 .4],'EdgeAlpha',.01);
   axis vis3d; light; light('Position', [-1 -1 -3]); lighting gouraud;
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
[img1,img2] = getstacks;
handles.img1 = img1;
handles.img2 = img2;
guidata(hObject,handles);
set(handles.slider1, 'Min', min(min(min(img1))));
set(handles.slider1, 'Max', max(max(max(img1))));
set(handles.slider2, 'Min', min(min(min(img2))));
set(handles.slider2, 'Max', max(max(max(img2))));
set(handles.slider1, 'Value', max(graythresh(img1)));
set(handles.slider2, 'Value', max(graythresh(img2)));
set(handles.edit1, 'String', num2str(get(handles.slider1,'Value')));
set(handles.edit2, 'String', num2str(get(handles.slider2,'Value')));
surf1;
surf2;
```

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
figure, subplot(211), imagesc(flipud(max(handles.img1,[],3))), axis image;
      subplot(212), imagesc(flipud(max(handles.img2,[],3))), axis image;
      set(gcf,'Color','Black');

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
axes(handles.img1plot); view(3);
axes(handles.img2plot); view(3);

% --- Executes on slider1 movement.
function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'String',num2str(get(handles.slider1,'Value')));
surf1;

% --- Executes when user enters values in edit1 textbox
function edit1_Callback(hObject, eventdata, handles)
set(handles.slider1,'Value',str2double(get(handles.edit1,'String')));
surf1;

% --- Executes on slider2 movement.
function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'String',num2str(get(handles.slider2,'Value')));
surf2;

% --- Executes when user enters values in edit2 textbox
function edit2_Callback(hObject, eventdata, handles)
img2 = handles.img2;
set(handles.slider2,'Value',str2double(get(handles.edit2,'String')));
surf2;

% --- Executes when user enters values in edit3 textbox
function edit3_Callback(hObject, eventdata, handles)
[r c z] = size(handles.img1);
if isfield(handles,'img1') == 1;
   surf1;
   surf2;
else isfield(handles,'img1') == 0;
   str2num(get(hObject,'String'));
end

% --- Executes on radiobutton1 activation.
function radiobutton1_Callback(hObject, eventdata, handles)
if get(handles.radiobutton1,'Value') == 1;
   set(handles.radiobutton2,'Value',0);
   set(handles.radiobutton3,'Value',0);
end

% --- Executes on radiobutton2 activation.
function radiobutton2_Callback(hObject, eventdata, handles)
if get(handles.radiobutton2,'Value') == 1;
   set(handles.radiobutton1,'Value',0);
   set(handles.radiobutton3,'Value',0);
```

```
end

% --- Executes on radiobutton3 activation.
function radiobutton3_Callback(hObject, eventdata, handles)
if get(handles.radiobutton3,'Value') == 1;
   set(handles.radiobutton1,'Value',0);
   set(handles.radiobutton2,'Value',0);
end

% --- Executes on button press in pushbutton2.
% This code does most of the work (lines 195-318)
function pushbutton2_Callback(hObject, eventdata, handles)

% Make Image1 Binary
[r c z] = size(handles.img1);
for i = 1:z;
   [img1th(:,:,i)] = im2bw([handles.img1(:,:,i)],get(handles.slider1, 'Value'));
end
img1th = bwareaopen(img1th,4,26);
img1thlabel = bwlabeln(img1th,26);

% Make Image2 Binary
for i = 1:z;
   [img2th(:,:,i)] = im2bw([handles.img2(:,:,i)],get(handles.slider2, 'Value'));
end
img2th = bwareaopen (img2th,4,26);
img2thlabel = bwlabeln (img2th,26);

handles.img1th = img1th; handles.img1thlabel = img1thlabel;
handles.img2th = img2th; handles.img2thlabel = img2thlabel;

% Find Overlap
ol = (img1th&img2th);
ollabel = bwlabeln (ol,26);

% Find particle volumes in each image
props1 = regionprops (img1thlabel,'Area','PixelList');
vols1 = [props1.Area];
props2 = regionprops (img2thlabel,'Area','PixelList');
vols2 = [props2.Area];
propsol = regionprops (ollabel,'Area','PixelList');
volsol = [propsol.Area];

% Find sum, mean, max, and min pixel intensity values
% for image1
pixlist1 = {props1.PixelList};
h = timebar('Finding Image1 Pixel Properties','Computation Progress');
%seticon(h,'C:\MATLAB6p5\p3icon.ico');
for i = 1:length(vols1)
   for j = 1:length(pixlist1{i}(:,1))
      [mpval(j)] = im2uint8(handles.img1(pixlist1{i}(j,2),pixlist1{i}(j,1),pixlist1{i}(j,3)));
   end
   [sumpixvals1(i)] = sum(mpval);
   [meanpixvals1(i)] = mean(mpval);
```

```
   [maxpixvals1(i)] = max(mpval);
   [minpixvals1(i)] = min(mpval);
   timebar(h,i/length(vols1));
   clear mpval;
end
close(h);
clear i j mpval;
p1pixmean = mean(meanpixvals1);
p1pixtot = sum(sumpixvals1);

% Find sum, mean, max, and min pixel intensity values
% for image2
pixlist2 = {props2.PixelList};
h = timebar('Finding Image2 Pixel Properties','Computation Progress');
%seticon(h,'C:\MATLAB6p5\p3icon.ico');
for i = 1:length(vols2)
   for j = 1:length(pixlist2{i}(:,1))
      [mpval(j)] = im2uint8(handles.img2(pixlist2{i}(j,2),pixlist2{i}(j,1),pixlist2{i}(j,3)));
   end
   [sumpixvals2(i)] = sum(mpval);
   [meanpixvals2(i)] = mean(mpval);
   [maxpixvals2(i)] = max(mpval);
   [minpixvals2(i)] = min(mpval);
   timebar(h,i/length(vols2));
   clear mpval;
end
close(h);
clear i j mpval;
p2pixmean = mean(meanpixvals2);
p2pixtot = sum(sumpixvals2);

% Compute total volume of particles in each image (and the overlaps)
vp1tot = length(find(img1th));
vp2tot = length(find(img2th));
voltot = length(find(ol));

% Compute number and mean volume of particles in each image
nump1 = length(vols1);              % Number of particles in img1
meanp1 = mean(vols1);               % Mean volume of particles in img1
nump2 = length(vols2);              % Number of particles in img2
meanp2 = mean(vols2);               % Mean volume of particles in img2
numol = length(volsol);             % Number of particles in overlap image
meanols = mean(volsol);             % Mean volume of particles in overlap image

% Compute fraction of overlaping volumes (for the whole image)
volsolp1 = sum(volsol)/sum(vols1);    % Overlaps as fraction of total img1 particle volumes
volsolp2 = sum(volsol)/sum(vols2);    % Overlaps as fraction of total img2 particle volumes

% Compute fraction of particles (number) that overlap (for the whole image)
numolp1 = numol/nump1;              % Number of overlap particles/number of particles in img1
numolp2 = numol/nump2;              % Number of overlap particles/number of particles in img2

% Compute fraction of EACH particle that contributes to overlap
olp = zeros (length(volsol),3);
```

```
for i = 1:length(volsol)
   olp(i,:) = [propsol(i).PixelList(1,:)];
end

set(0,'Units','Pixels');

% Sub-Volume Calculations (finding properties of particles iff they
% have any overlap)
volsp1sub = zeros(size(volsol));
volsp2sub = zeros(size(volsol));
h = timebar('Finding Sub-Volumes','Computation Progress');
%seticon(h,'C:\MATLAB6p5\p3icon.ico');
for i = 1:length(volsol)
   volsp1sub(i) = length(find(img1thlabel == img1thlabel((olp(i,2)),(olp(i,1)),(olp(i,3)))));
   volsp2sub(i) = length(find(img2thlabel == img2thlabel((olp(i,2)),(olp(i,1)),(olp(i,3)))));
   volsp1subidx(i) = img1thlabel((olp(i,2)),(olp(i,1)),(olp(i,3)));
   volsp2subidx(i) = img2thlabel((olp(i,2)),(olp(i,1)),(olp(i,3)));
   for j=1:length(propsol(i).PixelList(:,1))
      [pixol1(j)] = im2uint8(handles.img1((propsol(i).PixelList(j,2)), (propsol(i).PixelList(j,1)),
(propsol(i).PixelList(j,3))));
      [pixol2(j)] = im2uint8(handles.img2((propsol(i).PixelList(j,2)), (propsol(i).PixelList(j,1)),
(propsol(i).PixelList(j,3))));
   end
   [spixol1(i)] = sum(pixol1);
   [spixol2(i)] = sum(pixol2);
   clear pixol1;
   clear pixol2;
   timebar(h,i/length(volsol));
end
close(h);
p1fract = volsol./volsp1sub;
p2fract = volsol./volsp2sub;
sumpixol1 = sum(spixol1);
sumpixol2 = sum(spixol2);
meanpixol1 = mean((spixol1./volsol));
meanpixol2 = mean((spixol2./volsol));

for i = 1:z
   [img1thlabelsub(:,:,i)] = ismember(img1thlabel(:,:,i), volsp1subidx);
   [img2thlabelsub(:,:,i)] = ismember(img2thlabel(:,:,i), volsp2subidx);
end

varids = {'TH1','TH2','MeanV1','TotalV1','MeanPix1','TotalPix1','NumP1','MeanP1frac'...
      'MeanV2','TotalV2','MeanPix2','TotalPix2','NumP2','MeanP2frac'...
      'MeanVOL','TotalVOL','NumPOL','TotalPixOL1', 'MeanPixOL1','NumOLdivTot1','VOLdivTotV1',...
      'TotalPixOL2','MeanPixOL2','NumOLdivTot2','VOLdivTotV2'};
% Make data row vector
[data(1,:)] = [get(handles.slider1,'Value'), get(handles.slider2,'Value'),...
      meanp1, vp1tot, p1pixmean, p1pixtot, nump1, mean(p1fract),...
      meanp2, vp2tot, p2pixmean, p2pixtot, nump2, mean(p2fract),...
      meanols, voltot, numol,...
      sumpixol1, meanpixol1, numolp1, volsolp1,...
      sumpixol2, meanpixol2, numolp2, volsolp2];
% dlmwrite('rowdat.xls',data,'\t');
```

```
if isempty(dir('rowdat.txt')) == 1;
   fid = fopen('rowdat.txt','at+');
   fprintf(fid,'%s\t',varids{:});
   fprintf(fid,'\r');
   fprintf (fid,'%g\t',data);
   fclose(fid);
else isempty(dir('rowdat.txt')) == 0;
   fid = fopen('rowdat.txt','at+');
   fseek(fid,0,'eof');
   fprintf(fid,'\r');
   fprintf (fid,'%g\t',data);
   fclose (fid);
end

save alldat;

% Plot Colocalization isosurfaces
if (get(handles.radiobutton1,'Value'))+(get(handles.radiobutton2,'Value'))+...
  (get(handles.radiobutton3,'Value')) == 1;
   if get(handles.radiobutton1,'Value') == 1;
      fig2 = figure('numbertitle','off','name','Colocalization Plot');
      set(gcf,'InvertHardCopy', 'off','PaperPositionMode','Auto','Renderer','OpenGL');
      m = uimenu('Label','File');
         uimenu(m,'Label','Export...','Callback','print -dtiff -r300 Coloc');
      %seticon(gcf,'C:\MATLAB6p5\p3icon.ico');
      figure(fig2);
      p1 = patch(isosurface(img1thlabelsub,0));
      set(p1,'FaceColor','green','EdgeColor','none','FaceAlpha',1), lighting gouraud;
      hold on;
      p2 = patch(isosurface(img2thlabelsub,0));
      set(p2,'FaceColor','red','EdgeColor','none','FaceAlpha',0.4),
      light; light; light('Position', [-1 -1 -3]); lighting gouraud;
      set (gcf,'menubar','none','Color','Black','Units','Normalized','Position',[.1 .05 .8 .85]);
      view(3), axis([0 c 0 r 0 z]), grid off, box off; axis off;axis vis3d;
      set(gca,'DataAspectRatio',[1,1,1]);
      dlview on;
   elseif get(handles.radiobutton2,'Value') == 1;
      fig2 = figure('numbertitle','off','name','Colocalization Plot');
      set(gcf,'InvertHardCopy', 'off','PaperPositionMode','Auto','Renderer','OpenGL');
      m = uimenu('Label','File');
         uimenu(m,'Label','Export...','Callback','print -dtiff -r300 Coloc');
      %seticon(gcf,'C:\MATLAB6p5\p3icon.ico');
      figure(fig2);
      p1 = patch(isosurface(img1thlabel,0));
      set(p1,'FaceColor',[.3,.4,.6],'EdgeColor','none','FaceAlpha',0.3), lighting gouraud;
      hold on;
      p2 = patch(isosurface(img2thlabelsub,0));
      set(p2,'FaceColor','green','EdgeColor','none','FaceAlpha',1),
      light; light; light('Position', [-1 -1 -3]); lighting gouraud;
      set (gcf,'menubar','none','Color','Black','Units','Normalized','Position',[.1 .05 .8 .85]);
      view(3), axis([0 c 0 r 0 z]), grid off, box off; axis off;axis vis3d;
      set(gca,'DataAspectRatio',[1,1,1]);
      dlview on;
   else get(handles.radiobutton3,'Value') == 1;
```

```
        fig2 = figure('numbertitle','off','name','Colocalization Plot');
        set(gcf,'InvertHardCopy', 'off','PaperPositionMode','Auto','Renderer','OpenGL');
        m = uimenu('Label','File');
           uimenu(m,'Label','Export...','Callback','print -dtiff -r300 Coloc');
        %seticon(gcf,'C:\MATLAB6p5\p3icon.ico');
        figure(fig2);
        p1 = patch(isosurface(img1thlabelsub,0));
        set(p1,'FaceColor','green','EdgeColor','none','FaceAlpha',1), lighting gouraud;
        hold on;
        p2 = patch(isosurface(img2thlabel,0));
        set(p2,'FaceColor',[.3,.4,.6],'EdgeColor','none','FaceAlpha',0.3),
        light; light; light('Position', [-1 -1 -3]); lighting gouraud;
        set (gcf,'menubar','none','Color','Black','Units','Normalized','Position',[.1 .05 .8 .85]);
        view(3), axis([0 c 0 r 0 z]), grid off, box off; axis off;axis vis3d;
        set(gca,'DataAspectRatio',[1,1,1]);
        dlview on;
    end
else (get(handles.radiobutton1,'Value'))+(get(handles.radiobutton2,'Value'))+...
     (get(handles.radiobutton3,'Value')) == 0;
end

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
delete(hObject);

% --- Creates and returns a handle to the GUI figure.
function h1 = Particles3_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.

persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
    h1 = hsingleton;
    return;
end

h1 = figure(...
'Units','characters',...
'BackingStore','off',...
'CloseRequestFcn','Particles3("figure1_CloseRequestFcn",gcf,[],guidata(gcf))',...
'Color',[0.87843137254902 0.874509803921569 0.890196078431373],...
'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0 0.8125;0 0 0.875;0 0 0.9375;0 0 1;0 0.0625 1;0
0.125 1;0 0.1875 1;0 0.25 1;0 0.3125 1;0 0.375 1;0 0.4375 1;0 0.5 1;0 0.5625 1;0 0.625 1;0 0.6875 1;0 0.75
1;0 0.8125 1;0 0.875 1;0 0.9375 1;0 1 1;0.0625 1 1;0.125 1 0.9375;0.1875 1 0.875;0.25 1 0.8125;0.3125 1
0.75;0.375 1 0.6875;0.4375 1 0.625;0.5 1 0.5625;0.5625 1 0.5;0.625 1 0.4375;0.6875 1 0.375;0.75 1
0.3125;0.8125 1 0.25;0.875 1 0.1875;0.9375 1 0.125;1 1 0.0625;1 1 0;1 0.9375 0;1 0.875 0;1 0.8125 0;1 0.75
0;1 0.6875 0;1 0.625 0;1 0.5625 0;1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;1 0.25 0;1 0.1875 0;1 0.125 0;1
0.0625 0;1 0 0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;0.6875 0 0;0.625 0 0;0.5625 0 0],...
'DoubleBuffer','on',...
'IntegerHandle','off',...
'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
'MenuBar','none',...
'ToolBar','figure',...
'Name','Particles3',...
'NextPlot','replace',...
```

```
'NumberTitle','off',...
'PaperPosition',get(0,'defaultfigurePaperPosition'),...
'Position',[17.8 5.23076923076923 209 67.6153846153846],...
'Renderer','OpenGL',...
'RendererMode','manual',...
'WindowButtonDownFcn','dlview("down")',...
'WindowButtonUpFcn','dlview("up")',...
'CreateFcn','Particles3("figure1_CreateFcn",gcbo,[],guidata(gcbo))',...
'HandleVisibility','callback',...
'Tag','figure1',...
'UserData',[]);

setappdata(h1, 'GUIDEOptions',struct(...
'active_h', 141.000366210938, ...
'taginfo', struct(...
'figure', 2, ...
'axes', 3, ...
'slider', 3, ...
'pushbutton', 7, ...
'text', 6, ...
'edit', 4, ...
'radiobutton', 4, ...
'frame', 3), ...
'override', 1, ...
'release', 13, ...
'resize', 'simple', ...
'accessibility', 'callback', ...
'mfile', 1, ...
'callbacks', 1, ...
'singleton', 1, ...
'syscolorfig', 1, ...
'lastSavedFile', 'C:\MATLAB6p5\work\3dparticles\Particles3.m'));


h2 = axes(...
'Parent',h1,...
'View',[-37.5 30],...
'Box','on',...
'CameraPosition',[-4.47536790582704 -5.98402569409362 4.27491721763537],...
'CameraPositionMode',get(0,'defaultaxesCameraPositionMode'),...
'Color',[0 0 0],...
'ColorOrder',get(0,'defaultaxesColorOrder'),...
'DataAspectRatio',[1 1 0.8],...
'DataAspectRatioMode','manual',...
'DrawMode','fast',...
'GridLineStyle','--',...
'Position',[0.189473684210526 0.562002275312856 0.71866028708134 0.399317406143345],...
'XColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'XGrid','on',...
'XTick',[],...
'XTickMode','manual',...
'YColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'YGrid','on',...
'YTick',[],...
```

```
'YTickMode','manual',...
'ZColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'ZGrid','on',...
'ZTick',[],...
'ZTickMode','manual',...
'Tag','img1plot');


h3 = get(h2,'title');

set(h3,...
'Parent',h2,...
'Color',[0 0 0],...
'HorizontalAlignment','center',...
'Position',[0.292601242303295 0.238059871974735 1.5086149558603],...
'VerticalAlignment','bottom',...
'HandleVisibility','off');

h4 = get(h2,'xlabel');

set(h4,...
'Parent',h2,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','right',...
'Position',[0.0810430988934279 -0.838978817227471 0.14616899281302],...
'HandleVisibility','off');

h5 = get(h2,'ylabel');

set(h5,...
'Parent',h2,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','center',...
'Position',[-0.794096904613886 -0.151447703705669 0.163771653834303],...
'HandleVisibility','off');

h6 = get(h2,'zlabel');

set(h6,...
'Parent',h2,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','center',...
'Position',[-0.557043331997765 0.332777628167659 0.896042352319665],...
'Rotation',90,...
'VerticalAlignment','bottom',...
'HandleVisibility','off');

h7 = axes(...
'Parent',h1,...
'View',[-37.5 30],...
'Box','on',...
'CameraPosition',[-4.47536790582704 -5.98402569409362 4.27491721763537],...
'CameraPositionMode',get(0,'defaultaxesCameraPositionMode'),...
'Color',[0 0 0],...
```

```
'ColorOrder',get(0,'defaultaxesColorOrder'),...
'DataAspectRatio',[1 1 0.8],...
'DataAspectRatioMode','manual',...
'DrawMode','fast',...
'Position',[0.189473684210526 0.0705346985210466 0.71866028708134 0.399317406143345],...
'XColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'XTick',[],...
'XTickMode','manual',...
'YColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'YTick',[],...
'YTickMode','manual',...
'ZColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'ZTick',[],...
'ZTickMode','manual',...
'Tag','img2plot');


h8 = get(h7,'title');

set(h8,...
'Parent',h7,...
'Color',[0 0 0],...
'HorizontalAlignment','center',...
'Position',[0.292601242303296 0.238059871974735 1.5086149558603],...
'VerticalAlignment','bottom',...
'HandleVisibility','off');

h9 = get(h7,'xlabel');

set(h9,...
'Parent',h7,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','right',...
'Position',[0.0810430988934288 -0.838978817227471 0.14616899281302],...
'HandleVisibility','off');

h10 = get(h7,'ylabel');

set(h10,...
'Parent',h7,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','center',...
'Position',[-0.794096904613886 -0.151447703705669 0.163771653834303],...
'HandleVisibility','off');

h11 = get(h7,'zlabel');

set(h11,...
'Parent',h7,...
'Color',[0.501960784313725 0.501960784313725 0.501960784313725],...
'HorizontalAlignment','center',...
'Position',[-0.557043331997765 0.332777628167659 0.896042352319665],...
'Rotation',90,...
'VerticalAlignment','bottom',...
```

```
'HandleVisibility','off');

h12 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'CData',[],...
'ForegroundColor',[0 0 1],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.622298065984073 0.126315789473684 0.103526734926052],...
'String',{  '' },...
'Style','frame',...
'Tag','frame1',...
'UserData',[]);


h13 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'Callback','Particles3("slider1_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.286124401913876 0.517633674630262 0.62200956937799 0.0261660978384528],...
'String','slider1',...
'Style','slider',...
'SliderStep',[0.01 0.01],...
'TooltipString','move to adjust image1 threshold',...
'Tag','slider1');


h14 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'Callback','Particles3("slider2_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.286124401913876 0.0307167235494881 0.62200956937799 0.0261660978384528],...
'String',{  '' },...
'Style','slider',...
'SliderStep',[0.004 0.004],...
'TooltipString','move to adjust image2 threshold',...
'Tag','slider2');


h15 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'Callback','Particles3("pushbutton2_Callback",gcbo,[],guidata(gcbo))',...
'FontWeight','bold',...
'ForegroundColor',[0.901960784313726 0.901960784313726 0.901960784313726],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.569965870307167 0.126315789473684 0.0420932878270762],...
'String','Save and Run',...
'Tag','pushbutton2');
```

```
h16 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'Callback','Particles3("pushbutton3_Callback",gcbo,[],guidata(gcbo))',...
'FontWeight','bold',...
'ForegroundColor',[0.901960784313726 0.901960784313726 0.901960784313726],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.899886234357224 0.126315789473684 0.0420932878270762],...
'String','Load 3D Images',...
'Tag','pushbutton3');


h17 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[1 1 1],...
'Callback','Particles3("edit1_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.190430622009569 0.517633674630262 0.0698564593301435 0.025028441410694],...
'String','0',...
'Style','edit',...
'Tag','edit1');


h18 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[1 1 1],...
'Callback','Particles3("edit2_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.19043062200957 0.0352673492605233 0.0679425837320574 0.025028441410694],...
'String','0',...
'Style','edit',...
'Tag','edit2');


h19 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'Callback','Particles3("pushbutton4_Callback",gcbo,[],guidata(gcbo))',...
'FontWeight','bold',...
'ForegroundColor',[0.901960784313726 0.901960784313726 0.901960784313726],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.791808873720137 0.126315789473684 0.0420932878270762],...
'String','2D Projections',...
'Tag','pushbutton4');


h20 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'Callback','Particles3("radiobutton1_Callback",gcbo,[],guidata(gcbo))',...
```

```
'ListboxTop',0,...
'Position',[0.0258373205741627 0.667804323094426 0.109090909090909 0.0182025028441411],...
'String','Colocalization Only',...
'Style','radiobutton',...
'Value',1,...
'Tag','radiobutton1');


h21 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'Callback','Particles3("radiobutton2_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.0430622009569378 0.648464163822526 0.0755980861244019 0.0170648464163823],...
'String','All Image 1',...
'Style','radiobutton',...
'Tag','radiobutton2');


h22 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'Callback','Particles3("radiobutton3_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.0430622009569378 0.625711035267349 0.0755980861244019 0.0170648464163823],...
'String','All Image 2',...
'Style','radiobutton',...
'Tag','radiobutton3');


h23 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[0.0181818181818182 0.688282138794084 0.123444976076555 0.0182025028441411],...
'String','Colocalization Plots',...
'Style','text',...
'Tag','text4');


h24 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[0.501960784313725 0.501960784313725 0.501960784313725],...
'Callback','Particles3("pushbutton5_Callback",gcbo,[],guidata(gcbo))',...
'FontWeight','bold',...
'ForegroundColor',[0.901960784313726 0.901960784313726 0.901960784313726],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.731513083048919 0.126315789473684 0.0420932878270762],...
'String','Reset Axes',...
'Tag','pushbutton5');
```

```
h25 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'ForegroundColor',[0 0 1],...
'ListboxTop',0,...
'Position',[0.0172248803827751 0.844141069397042 0.126315789473684 0.0500568828213879],...
'String',{  '' },...
'Style','frame',...
'Tag','frame2');


h26 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'FontWeight','bold',...
'ListboxTop',0,...
'Position',[0.0181818181818182 0.845278725824801 0.0861244019138756 0.037542662116041],...
'String',{  'Reducepatch'; 'Ratio' },...
'Style','text',...
'Tag','text5');


h27 = uicontrol(...
'Parent',h1,...
'Units','normalized',...
'BackgroundColor',[1 1 1],...
'Callback','Particles3("edit3_Callback",gcbo,[],guidata(gcbo))',...
'ListboxTop',0,...
'Position',[0.105263157894737 0.857792946530148 0.0239234449760766 0.0273037542662116],...
'String','1',...
'Style','edit',...
'Tag','edit3');



hsingleton = h1;


% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)




gui_StateFields =  {'gui_Name'
                    'gui_Singleton'
                    'gui_OpeningFcn'
                    'gui_OutputFcn'
                    'gui_LayoutFcn'
                    'gui_Callback'};
gui_Mfile = '';
for i=1:length(gui_StateFields)
   if ~isfield(gui_State, gui_StateFields{i})
      error('Could not find field %s in the gui_State struct in GUI M-file %s', gui_StateFields{i}, gui_Mfile);
```

```
    elseif isequal(gui_StateFields{i}, 'gui_Name')
        gui_Mfile = [getfield(gui_State, gui_StateFields{i}), '.m'];
    end
end

numargin = length(varargin);

if numargin == 0
    % PARTICLES3
    % create the GUI
    gui_Create = 1;
elseif numargin > 3 & ischar(varargin{1}) & ishandle(varargin{2})
    % PARTICLES3('CALLBACK',hObject,eventData,handles,...)
    gui_Create = 0;
else
    % PARTICLES3(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = 1;
end

if gui_Create == 0
    varargin{1} = gui_State.gui_Callback;
    if nargout
        [varargout{1:nargout}] = feval(varargin{:});
    else
        feval(varargin{:});
    end
else
    if gui_State.gui_Singleton
        gui_SingletonOpt = 'reuse';
    else
        gui_SingletonOpt = 'new';
    end

    % Open fig file with stored settings.  Note: This executes all component
    % specific CreateFunctions with an empty HANDLES structure.

    % Do feval on layout code in m-file if it exists
    if ~isempty(gui_State.gui_LayoutFcn)
        gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);
    else
        gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        % If the figure has InGUIInitialization it was not completely created
        % on the last pass.  Delete this handle and try again.
        if isappdata(gui_hFigure, 'InGUIInitialization')
            delete(gui_hFigure);
            gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
        end
    end

    % Set flag to indicate starting GUI initialization
    setappdata(gui_hFigure,'InGUIInitialization',1);

    % Fetch GUIDE Application options
```

```
gui_Options = getappdata(gui_hFigure,'GUIDEOptions');

if ~isappdata(gui_hFigure,'GUIOnScreen')
   % Adjust background color
   if gui_Options.syscolorfig
      set(gui_hFigure,'Color', get(0,'DefaultUicontrolBackgroundColor'));
   end

   % Generate HANDLES structure and store with GUIDATA
   guidata(gui_hFigure, guihandles(gui_hFigure));
end

% If user specified 'Visible','off' in p/v pairs, don't make the figure
% visible.
gui_MakeVisible = 1;
for ind=1:2:length(varargin)
   if length(varargin) == ind
      break;
   end
   len1 = min(length('visible'),length(varargin{ind}));
   len2 = min(length('off'),length(varargin{ind+1}));
   if ischar(varargin{ind}) & ischar(varargin{ind+1}) & ...
         strncmpi(varargin{ind},'visible',len1) & len2 > 1
      if strncmpi(varargin{ind+1},'off',len2)
         gui_MakeVisible = 0;
      elseif strncmpi(varargin{ind+1},'on',len2)
         gui_MakeVisible = 1;
      end
   end
end

% Check for figure param value pairs
for index=1:2:length(varargin)
   if length(varargin) == index
      break;
   end
   try, set(gui_hFigure, varargin{index}, varargin{index+1}), catch, break, end
end

% If handle visibility is set to 'callback', turn it on until finished
% with OpeningFcn
gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
if strcmp(gui_HandleVisibility, 'callback')
   set(gui_hFigure,'HandleVisibility', 'on');
end

feval(gui_State.gui_OpeningFcn, gui_hFigure, [], guidata(gui_hFigure), varargin{:});

if ishandle(gui_hFigure)
   % Update handle visibility
   set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);

   % Make figure visible
   if gui_MakeVisible
```

```
            set(gui_hFigure, 'Visible', 'on')
            if gui_Options.singleton
               setappdata(gui_hFigure,'GUIOnScreen', 1);
            end
         end

         % Done with GUI initialization
         rmappdata(gui_hFigure,'InGUIInitialization');
      end

      % If handle visibility is set to 'callback', turn it on until finished with
      % OutputFcn
      if ishandle(gui_hFigure)
         gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
         if strcmp(gui_HandleVisibility, 'callback')
            set(gui_hFigure,'HandleVisibility', 'on');
         end
         gui_Handles = guidata(gui_hFigure);
      else
         gui_Handles = [];
      end

      if nargout
         [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
      else
         feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
      end

      if ishandle(gui_hFigure)
         set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);
      end
   end

   function gui_hFigure = local_openfig(name, singleton)
   try
      gui_hFigure = openfig(name, singleton, 'auto');
   catch
      % OPENFIG did not accept 3rd input argument until R13,
      % toggle default figure visible to prevent the figure
      % from showing up too soon.
      gui_OldDefaultVisible = get(0,'defaultFigureVisible');
      set(0,'defaultFigureVisible','off');
      gui_hFigure = openfig(name, singleton);
      set(0,'defaultFigureVisible',gui_OldDefaultVisible);
   end
```

```
function [img1,img2,img1sum,img2sum,r,c,z] = getstacks;
% GETSTACKS User interface for interactively loading 3D image stacks into
% the MATALAB workspace
%
% [img1,img2,img1sum,img2sum,r,c,z] = getstacks;
%
% Calling GETSTACKS with all the output arguments will generate 7 variables
% in the MATLAB workspace: the two images (img1 and img2), 2D projections
% of each image (img1sum, img2sum), and the coordinate dimensions of the
% images (r,c,z).  By default, the script only opens TIFF images, but any
% image type supported by MATLAB will work.  You can simply delete the
% '*.tif' option call from lines 19 and 38, and MATLAB should be able to
% determine all supported image types.
%
% NOTE that the function converts the images into DOUBLE format, even if
% the original data are only 8-bit.
%
% See Also UIGETFILE, IMREAD, IM2DOUBLE, IMFINFO

% Load First Image Stack

[stack1,direct1] = uigetfile('*.tif', 'Select First Stack');
cd (direct1)
info1 = imfinfo (stack1);
r1 = max([info1.Height]);
c1 = max([info1.Width]);
stacksize1 = length([info1.FileSize]);
img1 = zeros (r1,c1,stacksize1);

for i = 1:stacksize1;
   [img1(:,:,i)] = flipud(im2double(imread(stack1,i)));
end

img1sum = sum(img1,3);

% define image dimensions
[r c z] = size (img1);
clear i;

% Load Second Image Stack
[stack2,direct2] = uigetfile('*.tif', 'Select Second Stack');
cd (direct2)
info2 = imfinfo (stack2);
r2 = max([info2.Height]);
c2 = max([info2.Width]);
stacksize2 = length([info2.FileSize]);
img2 = zeros (r2,c2,stacksize2);

for i = 1:stacksize2;
   [img2(:,:,i)] = flipud(im2double(imread(stack2,i)));
end

img2sum = sum(img2,3);
```

```
function dlview(inp)
persistent last_pt

switch inp
	case 'on'
		last_pt = [];
		set(gcf, 'windowbuttondownfcn', 'dlview("down")')
		set(gcf, 'windowbuttonupfcn',  'dlview("up")')
		set(gcf, 'windowbuttonmotionfcn', '')
	case 'off'
		set(gcf, 'windowbuttondownfcn', '');
		set(gcf, 'windowbuttonupfcn',  '');
		set(gcf, 'windowbuttonmotionfcn', '');
	case 'down'
		set(gcf, 'windowbuttonmotionfcn', 'dlview("motion")');
		last_pt = get_pixel_pt(gcf);
	case 'up'
		set(gcf, 'windowbuttonmotionfcn', '');
		last_pt = [];
	case 'motion'
		new_pt = get_pixel_pt(gcf);
		d = new_pt - last_pt;
		last_pt = new_pt;
		switch lower(get(gcf,'SelectionType'))
			case 'normal'
				camorbit(-d(1), -d(2), 'camera')
			case 'alt'
				q = max(-.9, min(.9, sum(d)/70));
				camzoom(1+q);
			case 'extend'
				pan_d = d*camva(gca)/500;
				campan(-pan_d(1), -pan_d(2), 'camera');
			otherwise
		end
end

function pt = get_pixel_pt(figh)
p_units = get(figh, 'Units');
set(figh, 'Units', 'Pixels');
pt = get(figh, 'CurrentPoint');
pt = pt(1,1:2);
set(figh, 'Units', p_units);
```

```
function h = timebar(message,name,update_rate)

% TIMEBAR Progress bar with estimated time remaining
%   H = TIMEBAR('message','name') creates and displays a progress
%   bar with alphanumeric progress percentage and estimated time
%   time remaining.  A user input message ('message') is displayed,
%   typically to distinguish what process is being monitored, and
%   'name' is an optional figure name.  The figure handle H is
%   returned.
%
%   TIMEBAR(H,X) will update the length of the progress bar, the
%   percentage complete, and the estimated time remaining, where
%   X is a fractional progress between 0 (initial) and 1 (complete).
%   (Note that the order of H,X is opposite to waitbar.)
%
%   TIMEBAR(H,X,RATE) will update the progress bar and information
%   at a rate given by RATE in seconds.  Default is 0.1 seconds.
%
%   The estimated time remaining is linear using the initial time
%   (when TIMEBAR is first opened), the current time, and the percent
%   complete.
%
%   TIMEBAR is typically used inside a FOR loop or during numerical
%   simulation.  A sample for loop is shown below:
%
%     h = timebar('Loop counter','Progress')
%     for i = 1:100
%       % computation here %
%       timebar(h,1/100)
%     end
%     close(h)
%
%   A sample for numerical integration is shown below:
%
%     % script file
%     t0 = 0;
%     tf = 60;
%     h = timebar('Simulation integration','Progress')
%     [tt,xx] = ode45('states.m',[t0 tf],initial_conditions);
%     close(h)
%
%     % states.m
%     function xdot = states(t,x)
%     xdot(1) = ...;
%     xdot(2) = ...;
%     ...
%     timebar(h,(t-t0)/(tf-t0))
%
%   Version: 2.0
%   Version History:
%   1.0   2002-01-18   Initial release
%   2.0   2002-01-21   Added update rate option
%
%   Copyright 2002, Chad English
```

```
%    cenglish@myrealbox.com

if nargin < 3                                   % If update rate is not input
   update_rate = 0.1;                           %    set it to 0.1 seconds
end

if ~ishandle(message)                           % If first input is not a timebar handle,
                                                %    treat as new timebar

   % SET WINDOW SIZE AND POSITION
   winwidth = 300;                              % Width of timebar window
   winheight = 75;                              % Height of timebar window
   screensize = get(0,'screensize');            % User's screen size [1 1 width height]
   screenwidth = screensize(3);                 % User's screen width
   screenheight = screensize(4);                % User's screen height
   winpos = [0.5*(screenwidth-winwidth), ...
     0.5*(screenheight-winheight), winwidth, winheight];  % Position of timebar window origin

   % END SET WINDOW SIZE AND POSITION

   % OPEN FIGURE AND SET PROPERTIES
   if nargin < 2
      name = '';                                % If timebar name not input, set blank
   end

   wincolor = 0.75*[1 1 1];                     % Define window color
   est_text = 'Estimated time remaining: ';     % Set static estimated time text

   h = figure('menubar','none',...              % Turn figure menu display off
      'numbertitle','off',...                   % Turn figure numbering off
      'name',name,...                           % Set the figure name to input name
      'position',winpos,...                     % Set the position of the figure as above
      'color',wincolor,...                      % Set the figure color
      'resize','off',...                        % Turn of figure resizing
      'tag','timebar');                         % Tag the figure for later checking

   userdata.text(1) = uicontrol(h,'style','text',...   % Prepare message text (set the style to text)
      'pos',[10 winheight-30 winwidth-20 20],...       % Set the textbox position and size
      'hor','center',...                        % Center the text in the textbox
      'backgroundcolor',wincolor,...            % Set the textbox background color
      'foregroundcolor',0*[1 1 1],...           % Set the text color
      'string',message);                        % Set the text to the input message

   userdata.text(2) = uicontrol(h,'style','text',...   % Prepare static estimated time text
      'pos',[10 5 winwidth-20 20],...           % Set the textbox position and size
      'hor','left',...                          % Left align the text in the textbox
      'backgroundcolor',wincolor,...            % Set the textbox background color
      'foregroundcolor',0*[1 1 1],...           % Set the text color
      'string',est_text);                       % Set the static text for estimated time

   userdata.text(3) = uicontrol(h,'style','text',...   % Prepare estimated time
      'pos',[135 5 winwidth-145 20],...         % Set the textbox position and size
      'hor','left',...                          % Left align the text in the textbox
      'backgroundcolor',wincolor,...            % Set the textbox background color
```

```
    'foregroundcolor',0*[1 1 1],...                % Set the text color
    'string','');                                  % Initialize the estimated time as blank

  userdata.text(4) = uicontrol(h,'style','text',...  % Prepare the percentage progress
    'pos',[winwidth-35 winheight-50 25 20],...     % Set the textbox position and size
    'hor','right',...                              % Left align the text in the textbox
    'backgroundcolor',wincolor,...                 % Set the textbox background color
    'foregroundcolor',0*[1 1 1],...                % Set the textbox foreground color
    'string','');                                  % Initialize the progress text as blank

  userdata.axes = axes('parent',h,...              % Set the progress bar parent to the figure
    'units','pixels',...                           % Provide axes units in pixels
    'pos',[10 winheight-45 winwidth-50 15],...     % Set the progress bar position and size
    'xlim',[0 1],...                               % Set the range from 0 to 1
    'box','on',...                                 % Turn on axes box (to see where 100% is)
    'color',[1 1 1],...                            % Set plot background color to white
    'xtick',[],'ytick',[]);                        % Turn off axes tick marks and labels

  userdata.bar = patch([0 0 0 0 0],[0 1 1 1 0 0],'r');  % Initialize progress bar to zero area
  userdata.time = clock;                           % Record the current time
  userdata.inc = clock;                            % Set incremental clock to current time
  set(h, 'userdata', userdata)                     % Allow access to the text and axes settings
                                                   %    by including them with the timebar data
  % END OPEN FIGURE AND SET PROPERTIES

else                                               % If first input is a timebar handle, update
                                                   %    the window
  % GET HANDLE AND PROGRESS
  pause(10e-100)                                   % Message, bar, and static text won't display
                                                   %    without arbitrary pause (don't know why)
  h = message;                                     % Set handle to first input
  progress = name;                                 % Set progress to second input

  if ~strcmp(get(h,'tag'), 'timebar')              % Check object tag to see if it is a timebar
    error('Handle is not to a timebar window')     % If not a timebar, report error and stop
  end
  % END GET HANGLE AND PROGRESS

  % CALCULATE ESTIMATED TIME REMAINING
  userdata = get(h,'userdata');                    % Get the userdata included with the timebar
  inc = clock-userdata.inc;                        % Calculate time increment since last update
  inc_secs = inc(3)*3600*24 + inc(4)*3600 + ...
    inc(5)*60 + inc(6);                            % Convert the increment to seconds

  if [inc_secs > update_rate] | [progress == 1]    % Only update at update rate or 100% complete
    userdata.inc = clock;                          % If updating, reset the increment clock
    set(h,'userdata',userdata)                     % Update userdata with the new clock setting
    tpast = clock-userdata.time;                   % Calculate time since timebar initialized
    seconds_past = tpast(3)*3600*24 + tpast(4)*3600 + ...
      tpast(5)*60 + tpast(6);                      % Transform passed time into seconds
    estimated_seconds = seconds_past*(1/progress-1);  % Estimate the time remaining in seconds
    hours = floor(estimated_seconds/3600);         % Calculate integer hours of estimated time
    minutes = floor((estimated_seconds-3600*hours)/60); % Calculate integer minutes of estimated time
    seconds = floor(estimated_seconds-3600*hours- ...
```

```
      60*minutes);                                % Calculate integer seconds of estimated time
    tenths = floor(10*(estimated_seconds - ...
      floor(estimated_seconds)));                 % Calculate tenths of seconds (as integer)
    % END CALCULATE ESTIMATED TIME REMAINING

    % UPDATE ESTIMATED TIME AND PROGRESS TO TIMEBAR
    if progress > 1                               % Check if input progress is > 1
      time_message = ' Error!  Progress > 1!';    % If >1, print error to estimated time
      time_color = 'r';                           %    in red
    else
      if hours < 10; h0 = '0'; else h0 = '';end   % Put leading zero on hours if < 10
      if minutes < 10; m0 = '0'; else m0 = '';end % Put leading zero on minutes if < 10
      if seconds < 10; s0 = '0'; else s0 = '';end % Put leading zero on seconds if < 10
      time_message = strcat(h0,num2str(hours),':',m0,...
        num2str(minutes),':',s0,num2str(seconds),...
        '.',num2str(tenths),' (hh:mm:ss.t)');     % Format estimated time as hh:mm:ss.t
      time_color = 'k';                           % Format estimated time text as black
    end

    set(userdata.bar,'xdata',[0 0 progress progress 0]) % Update progress bar
    set(userdata.text(3),'string',time_message,...
      'foregroundcolor',time_color);              % Update estimated time
    set(userdata.text(4),'string',...
      strcat(num2str(floor(100*progress)),'%'));  % Update progress percentage
  end
  % END UPDATE ESTIMATED TIME AND PROGRESS TO TIMEBAR
end

% TIMEBAR HANDLE
if nargout == 0                                   % If handle not asked for
  clear h                                         %    do not output it
end
% END TIMEBAR HANDLE
```

Appendix B

MATLAB Code for 2-D Correlation Analysis:

The function for normalized 2-D correlation is included below. The output variables are identified by comments in the code. Three dependent functions, rnda, rndb, and fastxcor2 which randomize the particles for each image, and calculate the correlation, are included at the end of this code. This script also requires the 'timebar' function included in Appendix A.

```
function [NcorDat, NcorRnd, coloc] = xc2d

% 2DXcorr script for calculating normalized
% 2D correlation between two images.  The output
% variables are defined as:
%
% NcorDat: normalized correlation between images (a) and (b), computed as
%
% NcorDat = (corwin-meanrndcor)/corNNwin
%
% where corwin is the raw 2D correlation in a defined lag window,
% meanrndcorr is the mean of 1024 correlations generated from random
% versions of the real image data, and corNNwin is the autocorrelation
% of the image with the greatest amount of signal.  For example,
% if length(find(a)) > length(find(b)), corNNwin == coraawin.
%
%
% NcorRnd: normalized 95th percentile correlation from randomized data.
%
% coloc: colocalization value, determined as the zero-lag correlation,
% divided by the autocorrelation of the image with the greatest amount
% of signal.
%
% note that, by default, these are the only three output variables
% generated by this function.  If desired, all the output data will
% remain in the workspace by simply comenting out the 'function' line
% above and running the routine as a script.
%
% Last modified by W. Bryan Smith, Caltech, March 28, 2004.

[a1,direct1] = uigetfile('*.tif', 'Select First Image');
cd (direct1)
info1 = imfinfo (a1);
r = max([info1.Height]);
c = max([info1.Width]);
a = zeros (r,c);
a = flipud(im2double(imread(a1)));

[b1,direct2] = uigetfile('*.tif', 'Select Second Image');
cd (direct2)
```

```
info2 = imfinfo (b1);
b = zeros (r,c);
b = flipud(im2double(imread(b1)));

abw = im2bw(a, graythresh(a));
bbw = im2bw(b, graythresh(b));

% Compute real data cross-correlation
cor = fastxcor2(a,b);
[szcory,szcorx] = size(cor);

% Find properties of data particles
alabel = bwlabeln (abw,8);
blabel = bwlabeln (bbw,8);
aprops = regionprops (alabel, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength');
bprops = regionprops (blabel, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength');

clear *label

%% Extract properties of image 'a' particles
aarea = mean ([aprops(find([aprops.Area]>2)).Area]);
aep = std([aprops.MajorAxisLength])/2;
acentr = [aprops.Centroid];
acentrx = mean(acentr(find(diff(acentr(1:2:length(acentr)-1))  >= aep )));
amajor = round(mean([aprops.MajorAxisLength]));
aminor = round(mean([aprops.MinorAxisLength]));

%% Extract properties of image 'b' particles
barea = mean ([bprops(find([bprops.Area]>2)).Area]);
bep = std([bprops.MajorAxisLength])/2;
bcentr = [bprops.Centroid];
bcentrx = mean(bcentr(find(diff(bcentr(1:2:length(bcentr)-1))  >= bep )));
bmajor = round(mean([bprops.MajorAxisLength]));
bminor = round(mean([bprops.MinorAxisLength]));

clear m n i *props


if aminor > bminor
   winy = aminor;
elseif aminor < bminor
   winy = bminor;
else aminor = bminor
   winy = aminor;
end
if amajor > bmajor
   winx = bmajor;
elseif amajor < bmajor
   winx = amajor;
else amajor = bmajor
   winx = amajor;
end
```

```
%% Define limit of x-lag for peak search
%% based on the max mean distance between data particles
if acentrx == bcentrx
   xlim = acentrx;
elseif acentrx < bcentrx
   xlim = bcentrx;
else acentrx > bcentrx
   xlim = acentrx;
end

% Make xlim smaller, based on half the major axis length
% of the largest particle set because we used centroids
% to calculate the distance between particles
xlim = round ((xlim/2) - (0.5*winx));

% Randomize particles in each image 32 times
h = timebar('Generating Randomized Images','Progress');
rnda = zeros(r,c,32);
rndb = zeros(r,c,32);
for i = 1:32
   rnda(:,:,i) = rnda2(a);
   rndb(:,:,i) = rndb2(b);
   timebar(h,i/32);
end
close (h);

% Compute 1024 random cross-correlations
rndcor = zeros(szcory,(2*xlim)+1,1024);
h = timebar('Computing Random Correlations','Progress');
i = 0;
for n = 1:32
   for m = 1:32
      i = i+1;
      randcor = fastxcor2(rnda(:,:,n),rndb(:,:,m));
      rndcor(:,:,i) = randcor(:,c-xlim:c+xlim);
      timebar(h,i/1024);
   end
end
close(h);

% compute mean of random correlations,
meanrnd = mean(rndcor,3);

% Find 95th percentile random correlation
sumrnds = squeeze(sum(sum(rndcor)));
[sumrnds, sortidx] = sort(sumrnds);
rndcor95 = (rndcor(:,:,find(sortidx == 973)))-meanrnd;

% calculate normalized correlation window
corlim = cor(:,c-xlim:c+xlim);
corwin = corlim-meanrnd;
[szcorwiny,szcorwinx] = size(corwin);

filla = length(find(abw))/(r*c);
```

```
fillb = length(find(bbw))/(r*c);

%% Compute Properties of Autocorrelation of image a
coraa = fastxcor2(abw,abw);
coraawin = [coraa(:,c-xlim:c+xlim)] - meanrnd;
coraaPwin = coraawin(floor(r-(0.5*(winy))):ceil(r+(0.5*(winy))),...
   (floor(0.5*(szcorwinx-winx)):ceil(0.5*(szcorwinx+winx))));
coraamax = sum(sum(coraaPwin));

%% Compute Properties of Autocorrelation of image b
corbb = fastxcor2(bbw,bbw);
corbbwin = [corbb(:,c-xlim:c+xlim)] - meanrnd;
corbbPwin = corbbwin(floor(r-(0.5*(winy))):ceil(r+(0.5*(winy))),...
   (floor(0.5*(szcorwinx-winx)):ceil(0.5*(szcorwinx+winx))));
corbbmax = sum(sum(corbbwin));

%% Compute Properties of Peaks p1 and p2

% Find peak at -ylag
p1 = max(max(corwin(1:r,:)));
p1r = max(max(rndcor95(1:r,:)));
% Find peak at +ylag
p2 = max(max(corwin((r+1):szcorwiny,:)));
p2r = max(max(rndcor95((r+1):szcorwiny,:)));

[p1y,p1x] = find (corwin(1:r,:) == p1);
[p1ry,p1rx] = find(rndcor95(1:r,:) == p1r);
[p2y,p2x] = find (corwin((r+1):szcorwiny,:) == p2);
[p2ry,p2rx] = find (rndcor95((r+1):szcorwiny,:) == p2r);
p2y = p2y + r;
p2ry = p2ry +r;

if (p1x-(round(0.5*(winx)))) < 0;
   p1win = corwin((p1y-(round(0.5*(winy))):p1y+(round(0.5*(winy)))),...
      (1:p1x+(round(0.5*(winx)))));
elseif (p1x+(round(0.5*(winx)))) > szcorwinx;
   p1win = corwin((p1y-(round(0.5*(winy))):p1y+(round(0.5*(winy)))),...
      (p1x-(round(0.5*(winx))):(szcorwinx)));
else (p1x-(round(0.5*(winx)))) >= 0;
   if (p1x-(round(0.5*(winx)))) == 0
      p1win = corwin((p1y-(round(0.5*(winy))):p1y+(round(0.5*(winy)))),...
         (1+(p1x-(round(0.5*(winx)))):p1x+(round(0.5*(winx)))));
   elseif (p1x-(round(0.5*(winx)))) > 0
      p1win = corwin((p1y-(round(0.5*(winy))):p1y+(round(0.5*(winy)))),...
         (p1x-(round(0.5*(winx))):p1x+(round(0.5*(winx)))));
   end
end

if (p1rx-(round(0.5*(winx)))) < 0;
   p1rwin = corwin((p1ry-(round(0.5*(winy))):p1ry+(round(0.5*(winy)))),...
      (1:p1rx+(round(0.5*(winx)))));
elseif (p1rx+(round(0.5*(winx)))) > szcorwinx;
   p1rwin = corwin((p1ry-(round(0.5*(winy))):p1ry+(round(0.5*(winy)))),...
      (p1rx-(round(0.5*(winx))):(szcorwinx)));
```

```
else (p1rx-(round(0.5*(winx)))) >= 0;
   if (p1rx-(round(0.5*(winx)))) == 0
      p1rwin = corwin((p1ry-(round(0.5*(winy))):p1ry+(round(0.5*(winy)))),...
         (1+(p1rx-(round(0.5*(winx)))):p1rx+(round(0.5*(winx))));
   elseif (p1rx-(round(0.5*(winx)))) > 0
      p1rwin = corwin((p1ry-(round(0.5*(winy))):p1ry+(round(0.5*(winy)))),...
         (p1rx-(round(0.5*(winx))):p1rx+(round(0.5*(winx))));
   end
end

if (p2x-(round(0.5*(winx)))) < 0;
   p2win = corwin((p2y-(round(0.5*(winy))):p2y+(round(0.5*(winy)))),...
      (1:p2x+(round(0.5*(winx))));
elseif (p2x+(round(0.5*(winx)))) > szcorwinx;
   p2win = corwin((p2y-(round(0.5*(winy))):p2y+(round(0.5*(winy)))),...
      (p2x-(round(0.5*(winx))):(szcorwinx));
else (p2x-(round(0.5*(winx)))) >= 0;
   if (p2x-(round(0.5*(winx)))) == 0
      p2win = corwin((p2y-(round(0.5*(winy))):p2y+(round(0.5*(winy)))),...
         (1+(p2x-(round(0.5*(winx)))):p2x+(round(0.5*(winx))));
   elseif (p2x-(round(0.5*(winx)))) > 0
      p2win = corwin((p2y-(round(0.5*(winy))):p2y+(round(0.5*(winy)))),...
         (p2x-(round(0.5*(winx))):p2x+(round(0.5*(winx))));
   end
end

if (p2rx-(round(0.5*(winx)))) < 0;
   p2rwin = corwin((p2ry-(round(0.5*(winy))):p2ry+(round(0.5*(winy)))),...
      (1:p2rx+(round(0.5*(winx))));
elseif (p2rx+(round(0.5*(winx)))) > szcorwinx;
   p2rwin = corwin((p2ry-(round(0.5*(winy))):p2ry+(round(0.5*(winy)))),...
      (p2rx-(round(0.5*(winx))):(szcorwinx));
else (p2rx-(round(0.5*(winx)))) >= 0;
   if (p2rx-(round(0.5*(winx)))) == 0
      p2rwin = corwin((p2ry-(round(0.5*(winy))):p2ry+(round(0.5*(winy)))),...
         (1+(p2rx-(round(0.5*(winx)))):p2rx+(round(0.5*(winx))));
   elseif (p2rx-(round(0.5*(winx)))) > 0
      p2rwin = corwin((p2ry-(round(0.5*(winy))):p2ry+(round(0.5*(winy)))),...
         (p2rx-(round(0.5*(winx))):p2rx+(round(0.5*(winx))));
   end
end


% Integrate +/- ylag peaks
% Add them together to get the total correlation
p1sum = sum(sum(p1win));
p1rsum = sum(sum(p1rwin));

p2sum = sum(sum(p2win));
p2rsum = sum(sum(p2rwin));

corsum = p1sum+p2sum;
corsumr = p1rsum+p2rsum;
```

```
if filla == fillb
   cormax = max(max(coraa));
   NcorDat = corsum/coraamax;
   NcorRnd = corsumr/coraamax;
elseif filla > fillb;
   cormax = max(max(coraa));
   NcorDat = corsum/coraamax;
   NcorRnd = corsumr/coraamax;
else fillb > filla;
   cormax = max(max(corbb));
   NcorDat = corsum/corbbmax;
   NcorRnd = corsumr/corbbmax;
end
disp('Normalized corr score = '), disp(NcorDat)
disp('95th percentile randcor score = '), disp(NcorRnd)

% Express Colocalization as fraction of overlapping pixels
coloc = (max(cor(r,c)))/cormax;


% Plot the data
cmin = 0;
cmax = max(max(corwin));
r2 = 2^nextpow2(r);
c2 = 2^nextpow2(c);
subplot (411), imagesc (a),axis image, title(a1(1:length(a1)-4),'fontname','Times New Roman',...
   'fontsize',14,'VerticalAlignment','baseline');
subplot (412), imagesc (b),axis image, title(b1(1:length(b1)-4),'fontname','Times New Roman',...
   'fontsize',14,'VerticalAlignment','baseline');
subplot (413), imagesc (cor), set(gca,'XTickLabel',{'X0','c2'}), set(gca,'XTick',c2),...
   set(gca,'YTickLabel',{'Y0','r2'}), set(gca,'YTick',r2), caxis([cmin cmax]),axis tight,...
   title('cor','fontname','Times New Roman','fontsize',14,'VerticalAlignment','baseline');
subplot (414), imagesc (corwin),axis square,...
   set(gca,'XTick',[]),...
   set(gca,'YTickLabel',{'Y0','r2'}), set(gca,'YTick',r2),caxis([cmin cmax]),...
   title('corwin','fontname','Times New Roman','fontsize',14,'VerticalAlignment','baseline');
   text(p2x,p2y,'p2','color','white','FontSize',12,'HorizontalAlignment','center','fontname','Times New Roman');
   text(p1x,p1y,'p1','color','white','FontSize',12,'HorizontalAlignment','center','fontname','Times New Roman');
colorbar('horiz');

figure, subplot (221), surf (p1win), zlim([min(min(p1win)),max(max(corwin))]), caxis([cmin cmax]),...
   title('p1win','fontname','Times New Roman','fontsize',14);
subplot (222), surf (p2win), zlim([min(min(p2win)),max(max(corwin))]), caxis([cmin cmax]),...
   title('p2win','fontname','Times New Roman','fontsize',14);
subplot (223), contourf (p1win),caxis([cmin cmax]), colorbar('horiz'), axis image;
subplot (224), contourf (p2win),caxis([cmin cmax]), colorbar('horiz'), axis image;

% % clear s* w* xlim fill* h i* p1x p1y p2x p2y d* f* coraa corbb ans c c2 cmax cmin m n r r2
% % clear aarea acentr acentrx aep amajor aminor barea bcentr bcentrx bep bmajor bminor
```

```
function rnda = rnda2(a)
rand ('state',sum(100*clock));
rnda = zeros(size(a));
abw = im2bw(a,graythresh(a));
[albl,aobj] = bwlabeln (abw,8);

for objnum = 1:aobj
   [r,c]=find(albl==objnum);
   validated = 0;
   while validated == 0
      newloc = ([ceil(rand*size(abw,1)), ceil(rand*size(abw,2))]);
      rshift = newloc(1) - r(1);
      cshift = newloc(2) - c(1);
      newrows = r + rshift;
      newcols = c + cshift;
      if ~isempty(find(newrows < 1))
         newrows = newrows - min(newrows) + 1;
      end
      if ~isempty(find(newcols < 1))
         newcols = newcols - min(newcols) + 1;
      end
      if ~isempty(find(newrows > size(abw,1)))
         newrows = newrows-(max(newrows)-size(abw,1));
      end
      if ~isempty(find(newcols > size(abw,2)))
         newcols = newcols-(max(newcols)-size(abw,2));
      end
      val = [];
      for pix = 1:length(r)
         val = [val rnda(newrows(pix),newcols(pix))];
         if rnda(newrows(pix),newcols(pix))~=0
            break
         end
      end
      if length(find(val)) == 0
         validated = 1;
      end
   end
   for pix = 1:length(r)
      rnda(newrows(pix),newcols(pix)) = abw(r(pix),c(pix));
   end
end
```

```
function rndb = rndb2(b)
rand ('state',sum(100*clock));
rndb = zeros(size(b));
bbw= im2bw(b,graythresh(b));
[blbl,bobj] = bwlabeln (bbw,8);

for objnum = 1:bobj
   [r,c]=find(blbl==objnum);
   validated = 0;
   while validated == 0
      newloc = ([ceil(rand*size(bbw,1)), ceil(rand*size(bbw,2))]);
      rshift = newloc(1) - r(1);
      cshift = newloc(2) - c(1);
      newrows = r + rshift;
      newcols = c + cshift;
      if ~isempty(find(newrows < 1))
         newrows = newrows - min(newrows) + 1;
      end
      if ~isempty(find(newcols < 1))
         newcols = newcols - min(newcols) + 1;
      end
      if ~isempty(find(newrows > size(bbw,1)))
         newrows = newrows-(max(newrows)-size(bbw,1));
      end
      if ~isempty(find(newcols > size(bbw,2)))
         newcols = newcols-(max(newcols)-size(bbw,2));
      end
      val = [];
      for pix = 1:length(r)
         val = [val rndb(newrows(pix),newcols(pix))];
         if rndb(newrows(pix),newcols(pix))~=0
            break
         end
      end
      if length(find(val)) == 0
         validated = 1;
      end
   end
   for pix = 1:length(r)
      rndb(newrows(pix),newcols(pix)) = bbw(r(pix),c(pix));
   end
end


function cor = fastxcor2(a,b);

T_size = size(a);
A_size = size(b);
outsize = A_size + T_size - 1;
outsize = [2^nextpow2(outsize(1)),2^nextpow2(outsize(2))];
Fa = fft2(rot90(a,2),outsize(1),outsize(2));
Fb = fft2(b, outsize(1),outsize(2));
cor = real(ifft2(Fa .* Fb));
```