```
function GUI_MR_v11
%
========================================================================
======
% GUI-Based MRI Image Analysis Program
%
========================================================================
======

% define some constants (colors -- maybe used, maybe not)
yellow   = [1 1 0.75];

% define the main window of the GUI and add control objects to it
fig = figure('numbertitle','off','name','MRI',...
   'menubar','none','color',yellow,'position', [60 80 1700 900]);

% build the File menu with Open and Exit items
file_menu = uimenu('label','&File','accelerator','f');
uimenu(file_menu,'label','&Open','accelerator','o','tag','fopen');
uimenu(file_menu,'label','E&xit','accelerator','x','tag','fexit');

% text box for folder name
temp = dir;
uicontrol('Style','text',  'position',[ 10 760 140  20],'horizontalalignment','right',...
   'string','Folder: ','backgroundcolor',yellow)
uicontrol('Style','edit',  'position',[150 760 800  20],'horizontalalignment','left',...
   'String', temp(1).folder,'tag','folderbox');

% status
uicontrol('style','text',  'position', [10 735 140 20],'horizontalalignment','right',...
   'string','Status: ','backgroundcolor',yellow)
uicontrol('style','text',  'position',[150 735  800  20], 'horizontalalignment', 'left',...
   'string','Load Images','tag','status')

% load button
uicontrol('Style','pushbutton','position',[950 760 80  20],...
   'string','LOAD','tag','loadbutton')

% popupmenu for segment selection
uicontrol('Style','text',      'position',[ 10 710 140  20],'horizontalalignment','right',...
   'string','Segment: ','backgroundcolor',yellow)
uicontrol('Style','popupmenu', 'position',[150 710 100  20], ...
   'String', {'1';'2';'3';'4';'5';'6'},'tag','segment');
```

```
%Zoom box
uicontrol('style','text',  'position', [10 685 140 20],'horizontalalignment','right',...
   'string','Zoom Box: ','backgroundcolor',yellow)
uicontrol('Style','edit', 'position', [150 685 100 20], 'horizontalalignment', 'right', ...
   'String','100','backgroundcolor','white','tag','zoom_box');

% global front view
axes('units','pixels',     'position',[270 400 200 200],'tag','gl_f_axes');
uicontrol('style','text',  'position',[270 640 128  20],'string','Front View')
% global y-position box
uicontrol('style','text',  'position',[388 640  64  20],'string','Y (mm): ')
uicontrol('style','text',  'position',[438 640  64  20],'string','0','tag','gl_y_tval')
% global y-slider
uicontrol('style','slider','position',[ 30 355 256  20],'min',-1,'max',1,'value',0,'tag','gl_y_slider');
uicontrol('style','text',  'position',[ 10 355  20  20],'string','Y','backgroundcolor',yellow)

% global side view
axes('units','pixels',     'position',[30 400 200 200],'tag','gl_s_axes');
uicontrol('style','text',  'position',[30 640 128  20],'string','Side View')
% global x-position box
uicontrol('style','text',  'position',[128 640  64  20],'string','X (mm): ')
uicontrol('style','text',  'position',[192 640  64  20],'string','0','tag','gl_x_tval')
% global x-slider
uicontrol('style','slider','position',[270 10 256  20],'min',-1,'max',1,'value',0,'tag','gl_x_slider');
uicontrol('style','text',  'position',[250 10  20  20],'string','X','backgroundcolor',yellow)

% global bottom view
axes('units','pixels',     'position',[270 60 200 200],'tag','gl_b_axes');
uicontrol('style','text',  'position',[270 300 128  20],'string','Bottom View')
% global z-position box
uicontrol('style','text',  'position',[388 300  64  20],'string','Z (mm): ')
uicontrol('style','text',  'position',[438 300  64  20],'string','0','tag','gl_z_tval')
% global z-slider
uicontrol('style','slider','position',[506 350 20  256],'min',-1,'max',1,'value',0,'tag','gl_z_slider');
uicontrol('style','text',  'position',[506 606  20  20],'string','Z','backgroundcolor',yellow)
% global stacked view
axes('units','pixels', 'position',[30 60 200 200],'tag','gl_g_axes');

% zoomed front view
x0 = 700;
axes('units','pixels',     'position',[x0+150 400 200 200],'tag','zm_f_axes');
uicontrol('style','text',  'position',[x0+150 640 128  20],'string','Front View')
```

```
% zoomed z-position box
uicontrol('style','text',  'position',[x0+278 300  64  20],'string','Z (mm): ')
uicontrol('style','text',  'position',[x0+342 300  64  20],'string','0','tag','zm_z_tval')
% zoomed z-slider
uicontrol('style','slider','position',[x0+375 350  20
256],'min',-1,'max',1,'value',0,'tag','zm_z_slider');
uicontrol('style','text',  'position',[x0+375 606  20  20],'string','Z','backgroundcolor',yellow)

% zoomed side view
x0 = 350;
axes('units','pixels',     'position',[x0+250 400 200 200],'tag','zm_s_axes');
uicontrol('style','text',  'position',[x0+250 640 128  20],'string','Side View')
% zoomed y-position box
uicontrol('style','text',  'position',[x0+608 640  64  20],'string','Y (mm): ')
uicontrol('style','text',  'position',[x0+662 640  64  20],'string','0','tag','zm_y_tval')
% zoomed y-slider
uicontrol('style','slider','position',[x0+230 355  256
20],'min',-1,'max',1,'value',0,'tag','zm_y_slider');
uicontrol('style','text',  'position',[x0+210 355  20  20],'string','Y','backgroundcolor',yellow)

% zoomed bottom view
x0 = 700;
axes('units','pixels',     'position',[x0+150 60 200 200],'tag','zm_b_axes');
uicontrol('style','text',  'position',[x0+150 300 128  20],'string','Bottom View')
% zoomed y-position box
uicontrol('style','text',  'position',[x0+20 640  64  20],'string','X (mm): ')
uicontrol('style','text',  'position',[x0+70 640  64  20],'string','0','tag','zm_x_tval')
% zoomed y-slider
uicontrol('style','slider','position',[x0+150 10  256 20],'min',-1,'max',1,'value',0,'tag','zm_x_slider');
uicontrol('style','text',  'position',[x0+130 10  20  20],'string','X','backgroundcolor',yellow)
% zoomed stacked view
axes('units','pixels', 'position',[x0-100 60 200 200],'tag','zm_g_axes');

% isosurface
uicontrol('Style','pushbutton','position',[x0+605 530 40 20],...
   'string','Extract','tag','extractbutton')
uicontrol('style','text',  'position', [x0+485 550 140 20],'horizontalalignment','right',...
   'string','Pixel Intensity: ','backgroundcolor',yellow)
uicontrol('Style','edit', 'position', [x0+500 530 100 20], 'horizontalalignment', 'right', ...
   'String','100','backgroundcolor','white','tag','pixelintensity_box');
   %isosurface sliders
uicontrol('style','slider','position',[ x0+480 250 256
20],'min',-2*pi,'max',2*pi,'value',0,'tag','iso_ro_slider');
```

```
uicontrol('style','text',  'position',[ x0+430 248  50  20],'string','Rotate:','backgroundcolor',yellow)
uicontrol('style','slider','position',[2*x0+10 300  20
256],'min',-pi/2,'max',pi/2,'value',0,'tag','iso_phi_slider');
uicontrol('style','text',  'position',[2*x0 560  40  20],'string','Turn:','backgroundcolor',yellow)
   %isosurface graph
axes('units','pixels', 'position',[x0+500 300 200 200],'tag','isosurf');

uicontrol('style','text',  'position', [x0+450 640 140 20],'horizontalalignment','right',...
   'string','Pre-Programmed: ','backgroundcolor',yellow)
uicontrol('style','popupmenu', 'position',[x0+500 610 100  20], ...
   'String',{" "},'tag','pre_programmed');

%pixel intensity from cross hairs
uicontrol('style','text',  'position', [300 670 140 20],'horizontalalignment','right',...
   'string','Pixel Intensity: ','backgroundcolor',yellow)
uicontrol('style','text',  'position',[450 670  64  20],'string','0','tag','pbox')

uicontrol('style','text',  'position', [700 670 140 20],'horizontalalignment','right',...
   'string','Pixel Intensity: ','backgroundcolor',yellow)
uicontrol('style','text',  'position',[850 670  64  20],'string','0','tag','pbox1')
% -+-+-+-+- Boiler Plate: Single Callback -+-+-+-+-+-+-+-+-+-+-+-+-+-
handles = guihandles(fig);
guidata(fig,handles);
hca = struct2cell(handles);
for k = 1:length(hca)
   obj = hca{k};
   if isfield(obj,'callback') | isprop(obj,'callback')
      set(obj,'callback', { @main_cb, handles } );
   end
end
global counter;
counter = 0;
% main callback: process button and menu evens .....................
function main_cb(uiobject,eventdata,handles)
tag  = get(uiobject,'tag');
switch tag
   case {'gl_y_slider','gl_x_slider','gl_z_slider'}
      gshow(handles);
      zreset_sliders(handles);
      zshow(handles);
   case {'zm_z_slider','zm_x_slider','zm_y_slider'}
      zshow(handles);
   case {'extractbutton'}
```

```
        extract(handles);
    case {'iso_ro_slider','iso_phi_slider'}
        rotate(handles);
    case {'pre_programmed'}
        preprogram(handles);
    case {'segment','folderbox','loadbutton','zoom_box'}
        MR_load(handles);
        greset_sliders(handles)
        gshow(handles)
        zreset_sliders(handles);
        zshow(handles);
    case 'fopen'
        fpath = uigetdir;
        if fpath
            set(handles.folderbox,'string',fpath);
            MR_load(handles);
            greset_sliders(handles)
            gshow(handles)
            zreset_sliders(handles);
            zshow(handles);
        end
    case 'fexit'
        p = get(get(uiobject,'parent'),'parent');
        delete(p);
end

% reset the global view sliders .....................................
function greset_sliders(handles)
global imstack x y z cscale
% reset the global y-slider

set(handles.gl_y_slider,'min',min(y) + str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_y_slider,'max',max(y) - str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_y_slider,'value',0)
set(handles.gl_y_tval,'string',num2str(get(handles.gl_y_slider,'value')))

% reset the global x-slider
set(handles.gl_x_slider,'min',min(x) + str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_x_slider,'max',max(x) - str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_x_slider,'value',0)
set(handles.gl_x_tval,'string',num2str(get(handles.gl_x_slider,'value')))

% reset the global z-slider
```

```
set(handles.gl_z_slider,'min',min(z) + str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_z_slider,'max',max(z) - str2num(get(handles.zoom_box,'String'))/2)
set(handles.gl_z_slider,'value',round(mean(z)))
set(handles.gl_z_tval,'string',num2str(get(handles.gl_z_slider,'value')))

set(handles.iso_ro_slider,'min',0);
set(handles.iso_ro_slider,'max',360);
set(handles.iso_phi_slider,'min',0);
set(handles.iso_phi_slider,'max',360);

% reset the zoomed view sliders .....................................
function zreset_sliders(handles)
global imstack x y z cscale
% get size of zoom box
zoomsz = str2num(get(handles.zoom_box,'String'));           % <- *** get this from a text box
instead
% get position of global sliders
xslv   = get(handles.gl_x_slider,'value');
yslv   = get(handles.gl_y_slider,'value');
zslv   = get(handles.gl_z_slider,'value');        % <- *** get this from a slider instead
% build z coord vector for zoomed sub-imstack
[dummy,izz0] = min(abs(zslv-zoomsz/2-z));
[dummy,izz1] = min(abs(zslv+zoomsz/2-z));
zz = z(izz0:izz1);         % z-coords vector for zoomed sub-imstack
% reset the zoomed-view z-slider
set(handles.zm_z_slider, 'min',    min(zz))
set(handles.zm_z_slider, 'max',    max(zz))
set(handles.zm_z_slider, 'value',  round(mean(zz)))
set(handles.zm_z_tval,   'string', num2str(get(handles.zm_z_slider,'value')))

% build x coord vector for zoomed sub-imstack
[dummy,ixx0] = min(abs(xslv-zoomsz/2-x));
[dummy,ixx1] = min(abs(xslv+zoomsz/2-x));
xx = x(ixx0:ixx1);         % z-coords vector for zoomed sub-imstack
% reset the zoomed-view x-slider
set(handles.zm_x_slider, 'min',    min(xx))
set(handles.zm_x_slider, 'max',    max(xx))
set(handles.zm_x_slider, 'value',  round(mean(xx)))
set(handles.zm_x_tval,   'string', num2str(get(handles.zm_x_slider,'value')))

% build y coord vector for zoomed sub-imstack
[dummy,iyy0] = min(abs(yslv-zoomsz/2-y));
[dummy,iyy1] = min(abs(yslv+zoomsz/2-y));
```

```
yy = y(iyy0:iyy1);        % z-coords vector for zoomed sub-imstack
% reset the zoomed-view y-slider
set(handles.zm_y_slider, 'min',    min(yy))
set(handles.zm_y_slider, 'max',    max(yy))
set(handles.zm_y_slider, 'value',  round(mean(yy)))
set(handles.zm_y_tval,   'string', num2str(get(handles.zm_y_slider,'value')))


% global: display front, side, bottom and 3-D slice views .....
function gshow(handles)
global imstack x y z cscale
% get size of zoom box
zoomsz = str2num(get(handles.zoom_box,'String'));          % <- *** get this from a text box
instead
% get global x-y-z level from sliders
xslv = get(handles.gl_x_slider,'value');         % <- *** get this from a slider instead
yslv = get(handles.gl_y_slider,'value');
zslv = get(handles.gl_z_slider,'value');         % <- *** get this from a slider instead
% update text box(es)
set(handles.gl_x_tval,'string',num2str(round(xslv)))
set(handles.gl_y_tval,'string',num2str(round(yslv)))
set(handles.gl_z_tval,'string',num2str(round(zslv)))
% find indices corresponding to global x-y-z levels
[dummy,iysl] = min(abs(yslv-y));        % iysl = index for y-level
[dummy,ixsl] = min(abs(xslv-x));
[dummy,izsl] = min(abs(zslv-z));
% show global front  view (x-z) at specified y position
ax = handles.gl_f_axes;
pcolor(ax,x,z,squeeze(imstack(:,iysl,:))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[xslv xslv],[min(z) max(z)],'y',[min(x) max(x)],[zslv zslv],'y')
rectangle(ax,'position',[[xslv zslv]-zoomsz/2 zoomsz zoomsz],'edgecolor','red');
hold(ax,'off')
% show global bottom view (x-y) at specified z position
ax = handles.gl_b_axes;
pcolor(ax,x,y,squeeze(imstack(:,:,izsl))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[xslv xslv],[min(y) max(y)],'y',[min(x) max(x)],[yslv yslv],'y')
rectangle(ax,'position',[[xslv yslv]-zoomsz/2 zoomsz zoomsz],'edgecolor','red');
hold(ax,'off')
% show global side view (y-z) at specified x position
```

```
ax = handles.gl_s_axes;
pcolor(ax,y,z,squeeze(imstack(ixsl,:,:))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[yslv yslv],[min(z) max(z)],'y',[min(y) max(y)],[zslv zslv],'y')
rectangle(ax,'position',[[yslv zslv]-zoomsz/2 zoomsz zoomsz],'edgecolor','red');
hold(ax,'off')
% show global 3-D slices
ax3 = handles.gl_g_axes;
slice(ax3,y,x,z,imstack,yslv,[],[])
hold(ax3,'on')
slice(ax3,y,x,z,imstack,[],xslv,[])
slice(ax3,y,x,z,imstack,[],[],zslv)
hold(ax3,'off')
shading(ax3, 'flat'); view(ax3, [60,30])
axis(ax3,'equal'); axis(ax3,'tight')
xlabel(ax3,'x (mm)');ylabel(ax3,'y (mm)');zlabel(ax3, 'z (mm)');
colormap(ax3,gray);

set(handles.pbox,'string', num2str(imstack(ixsl, iysl, izsl)))



% zoomed: display front, side, bottom and 3-D slice views .....
function zshow(handles)
global imstack x y z cscale
% get size of zoom box
zoomsz = str2num(get(handles.zoom_box,'String'));          % <- *** get this from a text box
instead
intensityval = str2num(get(handles.pixelintensity_box,'String'));
% get global x-y-z level from global sliders
xslv = get(handles.gl_x_slider,'value');          % <- *** get this from a slider instead
yslv = get(handles.gl_y_slider,'value');
zslv = get(handles.gl_z_slider,'value');          % <- *** get this from a slider instead
% update global x-y-z text box(es)
set(handles.gl_x_tval,'string',num2str(round(xslv)))
set(handles.gl_y_tval,'string',num2str(round(yslv)))
set(handles.gl_z_tval,'string',num2str(round(zslv)))
% get stack indices corresponding to x-y-z levels +/- zoom size/2 ...
[dummy,ixz0] = min(abs(xslv-zoomsz/2-x));
[dummy,ixz1] = min(abs(xslv+zoomsz/2-x));
[dummy,iyz0] = min(abs(yslv-zoomsz/2-y));
[dummy,iyz1] = min(abs(yslv+zoomsz/2-y));
[dummy,izz0] = min(abs(zslv-zoomsz/2-z));
```

```
[dummy,izz1] = min(abs(zslv+zoomsz/2-z));
zx = x(ixz0:ixz1);
zy = y(iyz0:iyz1);
zz = z(izz0:izz1);
% get zoomed sub-image stack from global image stack
zimstack = imstack(ixz0:ixz1,iyz0:iyz1,izz0:izz1);
% get zoomed x-y-z level from zoom sliders
xslv = get(handles.zm_x_slider,'value');        % <- *** get this from a slider instead
yslv = get(handles.zm_y_slider,'value');        % <- *** get this from a slider instead
zslv = get(handles.zm_z_slider,'value');
% update zoomed x-y-z text box(es)
set(handles.zm_z_tval,'string',num2str(round(zslv)))
set(handles.zm_x_tval,'string',num2str(round(xslv)))
set(handles.zm_y_tval,'string',num2str(round(yslv)))
% find indices corresponding to zoomed x-y-z levels
[dummy,ixsl] = min(abs(xslv-zx));
[dummy,iysl] = min(abs(yslv-zy));       % iysl = index for y-level
[dummy,izsl] = min(abs(zslv-zz));
% show zoomed front view (x-z) at specified y position
ax = handles.zm_f_axes;
pcolor(ax,zx,zz,squeeze(zimstack(:,iysl,:))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[xslv xslv],[min(zz) max(zz)],'g',[min(zx) max(zx)],[zslv zslv],'g')
hold(ax,'off')
% show zoomed bottom view (x-y) at specified z position
ax = handles.zm_b_axes;
pcolor(ax,zx,zy,squeeze(zimstack(:,:,izsl))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[xslv xslv],[min(zy) max(zy)],'g',[min(zx) max(zx)],[yslv yslv],'g')
hold(ax,'off')
% show zoomed side view (y-z) at specified x position
ax = handles.zm_s_axes;
pcolor(ax,zy,zz,squeeze(zimstack(ixsl,:,:))')
decorate(ax,cscale)
hold(ax,'on')
plot(ax,[yslv yslv],[min(zz) max(zz)],'g',[min(zy) max(zy)],[zslv zslv],'g')
hold(ax,'off')
% show zoomed 3-D slices
ax3 = handles.zm_g_axes;
slice(ax3,zy,zx,zz,zimstack,yslv,[],[])
hold(ax3,'on')
```

```
slice(ax3,zy,zx,zz,zimstack,[],xslv,[])
slice(ax3,zy,zx,zz,zimstack,[],[],zslv)
hold(ax3,'off')
shading(ax3, 'flat'); view(ax3, [60,30])
axis(ax3,'equal'); axis(ax3,'tight')
xlabel(ax3,'x (mm)');ylabel(ax3,'y (mm)');zlabel(ax3, 'z (mm)');
colormap(ax3,gray);

set(handles.pbox1,'string', num2str(zimstack(ixsl, iysl, izsl)))

function extract(handles)
global imstack x y z
%isosurface
zoomsz = str2num(get(handles.zoom_box,'String'));
xslv = get(handles.gl_x_slider,'value');          % <- *** get this from a slider instead
yslv = get(handles.gl_y_slider,'value');
zslv = get(handles.gl_z_slider,'value');
set(handles.iso_ro_slider,'value',60);
set(handles.iso_phi_slider, 'value',30);
[dummy,ixz0] = min(abs(xslv-zoomsz/2-x));
[dummy,ixz1] = min(abs(xslv+zoomsz/2-x));
[dummy,iyz0] = min(abs(yslv-zoomsz/2-y));
[dummy,iyz1] = min(abs(yslv+zoomsz/2-y));
[dummy,izz0] = min(abs(zslv-zoomsz/2-z));
[dummy,izz1] = min(abs(zslv+zoomsz/2-z));
zx = x(ixz0:ixz1);
zy = y(iyz0:iyz1);
zz = z(izz0:izz1);
% get zoomed sub-image stack from global image stack
zimstack = imstack(ixz0:ixz1,iyz0:iyz1,izz0:izz1);

intensityval = str2num(get(handles.pixelintensity_box,'String'));
axs = handles.isosurf;
xlabel(axs,'x (mm)');
ylabel(axs,'y (mm)');
zlabel(axs,'z (mm)');
cla(axs)
colormap(axs,gray);
s = isosurface(zy,zx,zz,zimstack,intensityval);
patch(axs, s, 'FaceColor', [0.75,0.75,0.75] ,'EdgeColor','none');
view(axs, [60 30])
camlight(axs,'left')
lighting(axs,'gouraud')
```

```
function preprogram(handles)
global imstack x y z
%isosurface
popup = get(handles.pre_programmed,'String');
value = popup{get(handles.pre_programmed, 'value')};
switch value
    case "calf (skin)"
        zoomsz = 150;
        xslv = 22;
        yslv = -1;
        zslv = 211;
        intensityval = 250;
     case "head (skin)"
        zoomsz = 250;
        xslv = 0;
        yslv = 0;
        zslv = 64;
        intensityval = 250;
     case "left shoulder"
        zoomsz = 150;
        xslv = 145;
        yslv = 0;
        zslv = 151;
        intensityval = 250;
     case "abdomen (skin)"
        zoomsz = 300;
        xslv = -21;
        yslv = 0;
        zslv = 150;
        intensityval = 250;
     case "thigh (skin)"
        zoomsz = 250;
        xslv = 95;
        yslv = 0;
        zslv = 125;
        intensityval = 50;
     case "toes"
        zoomsz = 150;
        xslv = 32;
        yslv = 0;
        zslv = 75;
```

```
        intensityval = 100;
end
set(handles.iso_ro_slider,'value',60);
set(handles.iso_phi_slider, 'value',30);
[dummy,ixz0] = min(abs(xslv-zoomsz/2-x));
[dummy,ixz1] = min(abs(xslv+zoomsz/2-x));
[dummy,iyz0] = min(abs(yslv-zoomsz/2-y));
[dummy,iyz1] = min(abs(yslv+zoomsz/2-y));
[dummy,izz0] = min(abs(zslv-zoomsz/2-z));
[dummy,izz1] = min(abs(zslv+zoomsz/2-z));
zx = x(ixz0:ixz1);
zy = y(iyz0:iyz1);
zz = z(izz0:izz1);
% get zoomed sub-image stack from global image stack
zimstack = imstack(ixz0:ixz1,iyz0:iyz1,izz0:izz1);
axs = handles.isosurf;
xlabel(axs,'x (mm)');
ylabel(axs,'y (mm)');
zlabel(axs,'z (mm)');
cla(axs)
colormap(axs,gray);
s = isosurface(zy,zx,zz,zimstack,intensityval);
patch(axs, s, 'FaceColor', [0.75,0.75,0.75] ,'EdgeColor','none');
view(axs, [60 30])
camlight(axs,'left')
lighting(axs,'gouraud')

function rotate(handles)
axs = handles.isosurf;
az = get(handles.iso_ro_slider,'value');
el = get(handles.iso_phi_slider, 'value');
view(axs, [az el])

% set colormap, color scale, shading, etc, for plot .....................
function decorate(ax,cminmax,xlb,ylb,zlb,azel)
colormap(ax,gray);
caxis(ax,cminmax);
shading(ax,'flat'); axis(ax,'equal'); axis(ax,'tight')
if nargin > 2
   if ~isempty(xlb)
      xlabel(ax,xlb)
   end
end
```

```
if nargin > 3
   if ~isempty(ylb)
      ylabel(ax,ylb)
   end
end
if nargin > 4
   if ~isempty(zlb)
      zlabel(ax,zlb)
   end
end
if nargin > 5
   view(ax,azel);
end

% load data based on selected segment, and selected folder .....
function MR_load(handles)
global imstack x y z cscale
folder     = get(handles.folderbox,'string');
MR_channel = folder(end-6:end);
dirlist    = dir(folder);
sect       = get(handles.segment,'value');

set(handles.status, 'string', 'Ready')
if sect > 1
   sect = sect+1;
end
fids = [];
for k = 1:length(dirlist)
   fname = dirlist(k).name;
   if length(fname) >= 4
      if fname(end-3:end) == '.png' & fname(end-4) == MR_channel(end)
         fidx = str2num(fname(4:7));
         if floor(fidx/1000) == sect
            fids = [fids ; fidx];
         end
      end
   end
end
if ~isempty(fids)
   fids = sort(fids,'descend');
   nim  = length(fids);
   k = 1;
   fname = [folder '/mvf' num2str(fids(k)) MR_channel(end) '.png'];
```

```
im1  = imread(fname);
if sect == 1
   imstack = single(zeros(size(im1,1),size(im1,2),nim));
   imstack(:,:,k) = fliplr(im1');
else
   imstack = single(zeros(size(im1,2),nim,size(im1,1)));
   imstack(:,nim-k+1,:) = fliplr(im1');
end
tic;
for k = 2:nim
   fname = [folder '/mvf' num2str(fids(k)) MR_channel(end) '.png'];
   if sect == 1
      imstack(:,:,k) = fliplr(imread(fname)');
   else
      imstack(:,nim-k+1,:) = fliplr(imread(fname)');
   end
end
% define x, y, z axis extents ----------------------------------------
if sect == 1
   y = linspace(-110,110,256);
   x = linspace(-110,110,256);
   z = 4*(0:nim-1);
else
   x = 2*linspace(-110,110,256);
   y = 4*(0:nim-1) - 2*nim;
   z = 2*linspace(0,220,256);
end
% find min and max pixel intensities of imstack
cscale = [min(imstack(:)) max(imstack(:))];
popup = get(handles.segment,'value');
switch popup
   case {1}
      set(handles.pre_programmed,'string',{'head (skin)'});
   case {2}
      set(handles.pre_programmed,'string',{'left shoulder'});
   case {3}
      set(handles.pre_programmed,'string',{'abdomen (skin)'});
   case {4}
      set(handles.pre_programmed,'string',{'thigh (skin)'});
   case {5}
      set(handles.pre_programmed,'string',{'calf (skin)'});
   case {6}
      set(handles.pre_programmed,'string',{'toes'});
```

```
    end
else
    set(handles.folderbox,'string','Files Not Found');    % <- *** put this in a different text box
end
```