Tractography of The Heart Through DTI

Project Members: Merrill Datwyler, Michael Valencia, Omar Bahasan, Taylor Hatch

Sponsoring Faculty: Edward Hsu, PhD – University of Utah Small Animal Imaging Lab

Assigned Tasks

- Tylor Hatch:
- Merrill Datwyler:
- Micheal Valencia:
- Omer Bahasan:

What is Tractography

Tractography is the use of Diffusion-weighted MRI (DWMRI), data to simulate the 3D axonal directionality in tissues i.e. brains and hearts.

Complications

- Seed points.
- Fourth dimension: time of a heart beat.
- Integration methods.
- Data size.

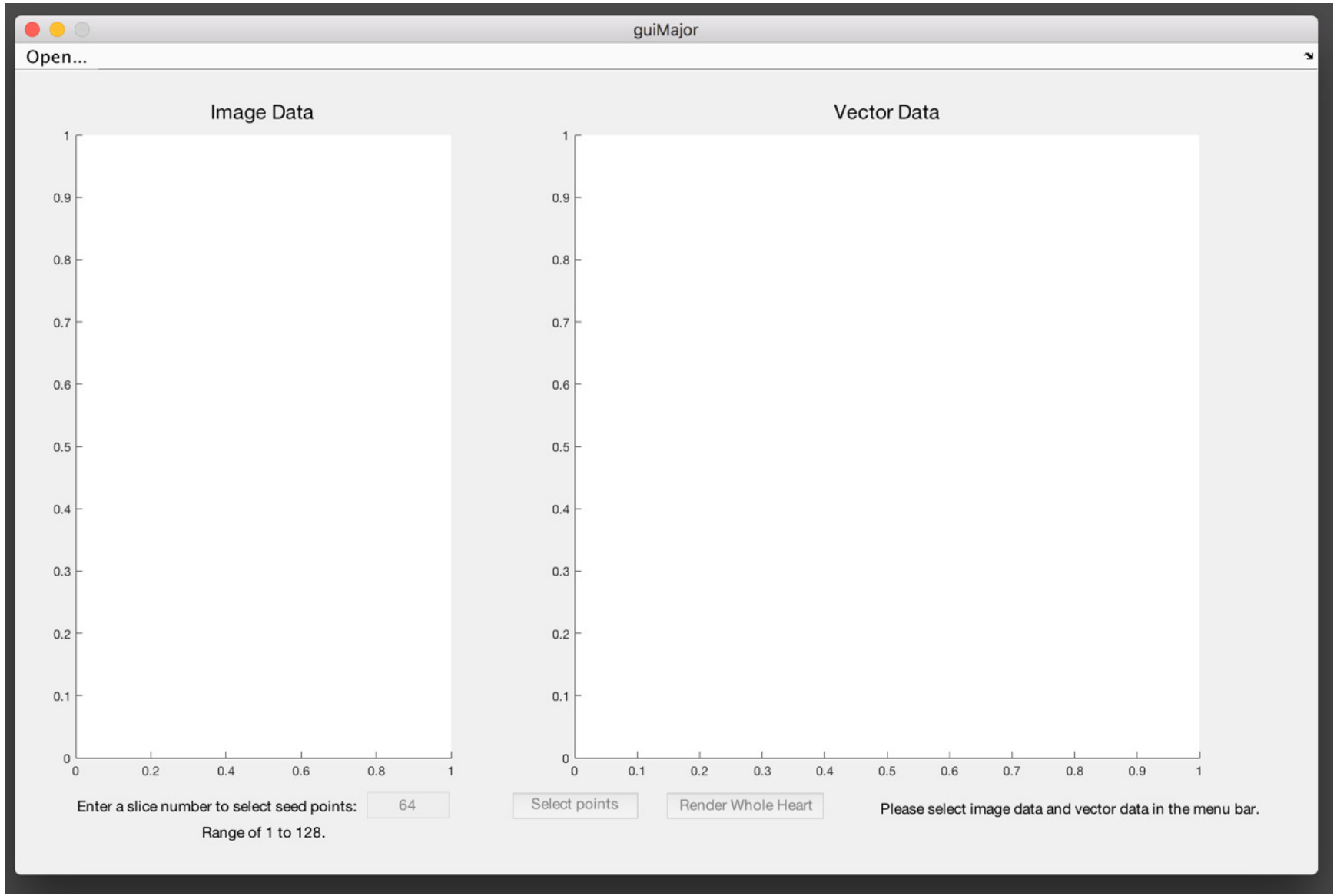
Approach

- Modify MRI image data into readable context for MATLAB i.e. RBG data
- Evaluation of the Eigenvectors for the matrix for determining directionality of water diffusion
- Create 3D vector field of the matrix using Quiver (3D)
- Create userdefined function to replace MATLAB's streamline using eulers integration, in order to trace by setting diffusion conditions
- Creating seed point, and boundaries for tracing
- Create GUI to set seed point, line width, Frames for mp4, display image, run time, and create movie file.

Improving MRI

- MRI imaging display general structure of soft tissue through aligning proton spin in water molecules, however it is often desired to have not only the structure but also the appropriate network.

Application flow



Click on ‘Open…’ to begin:

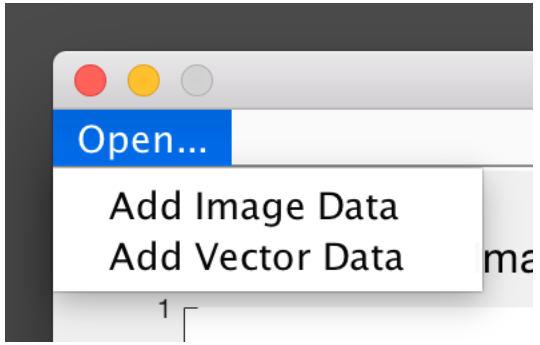
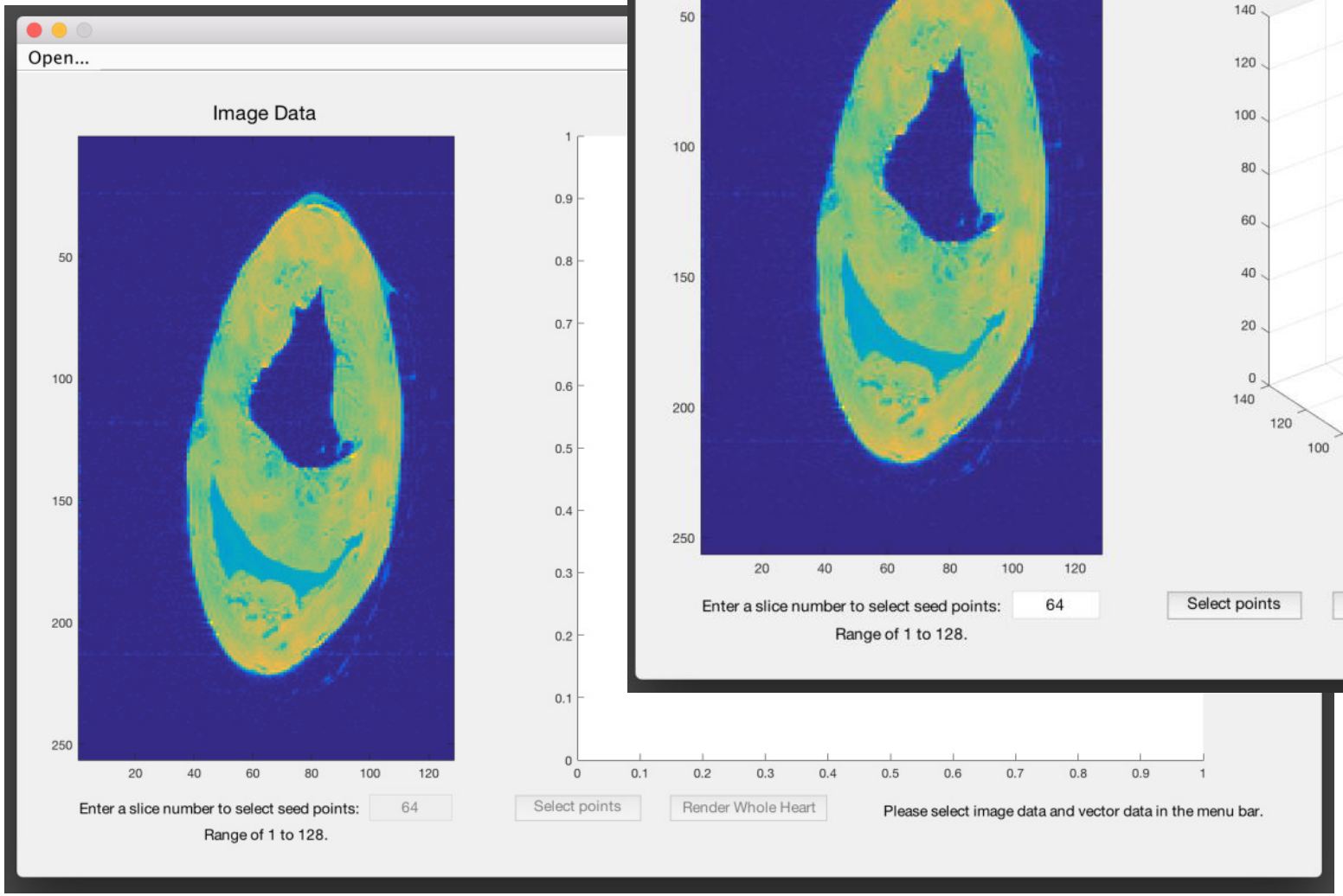


Image data added:



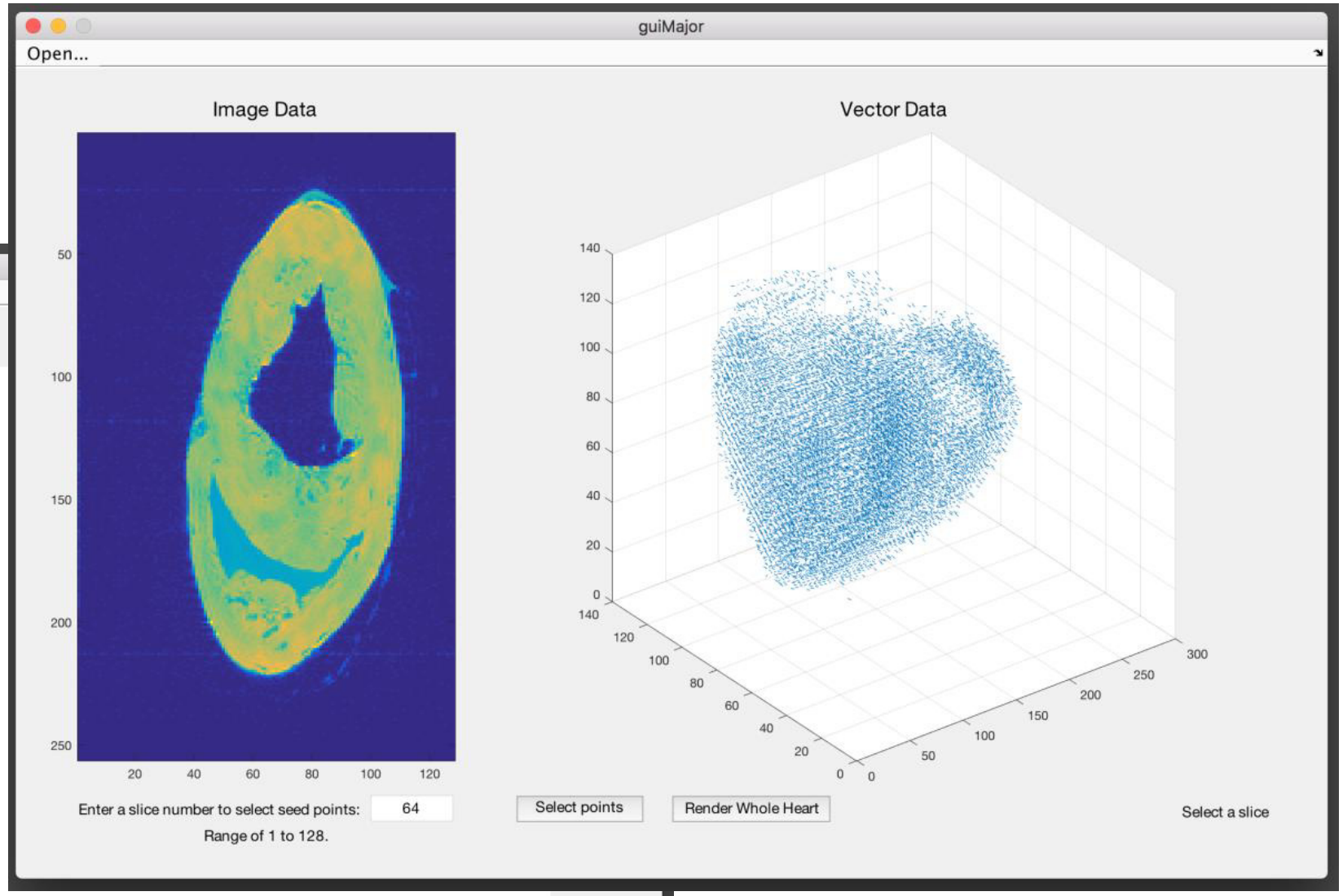
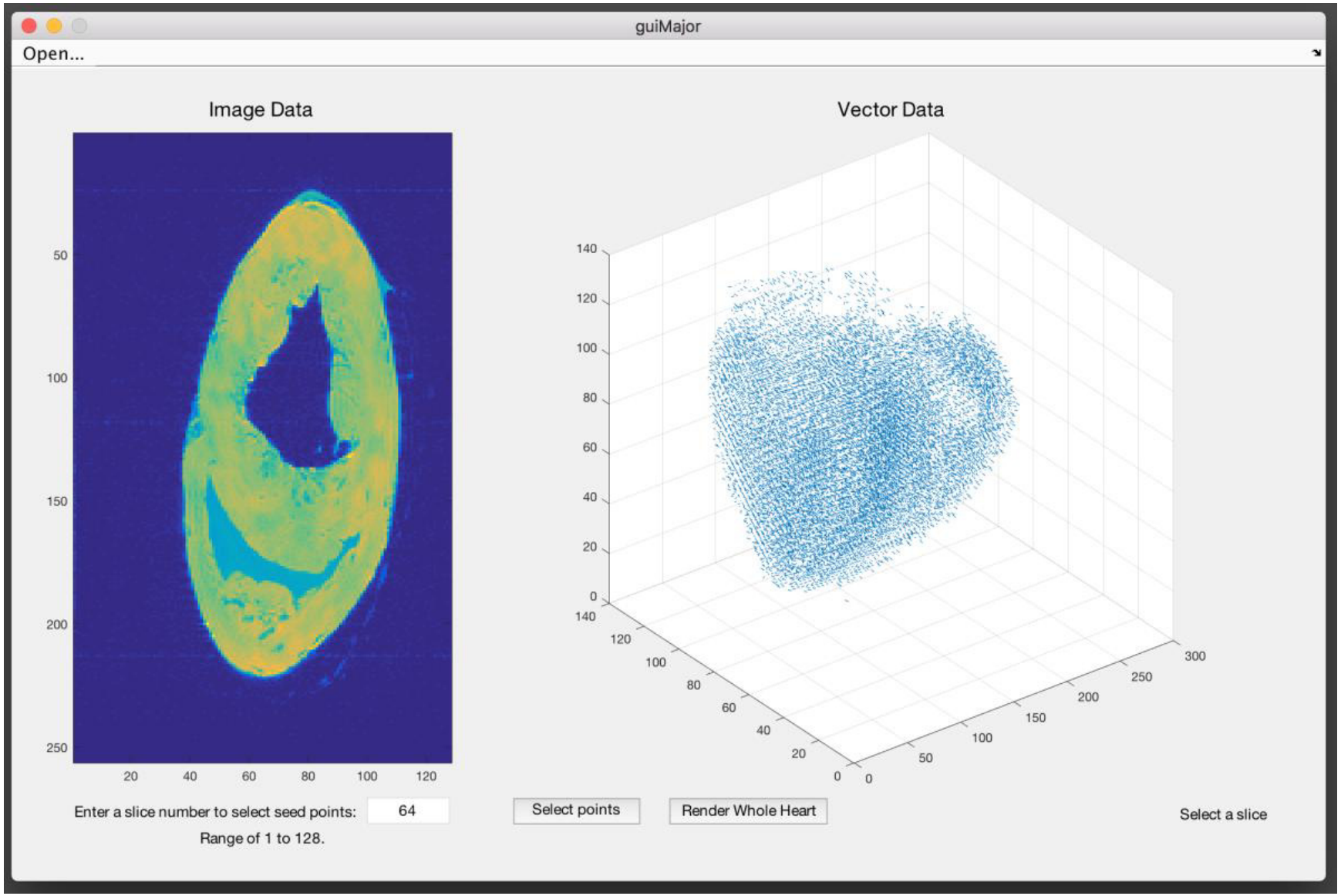
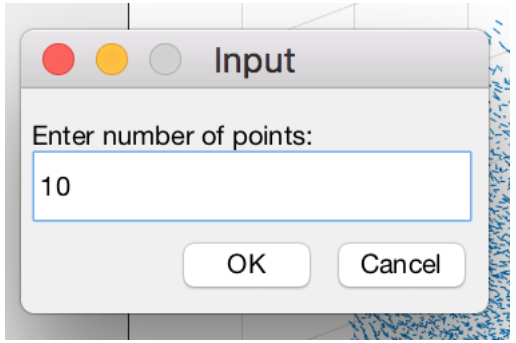
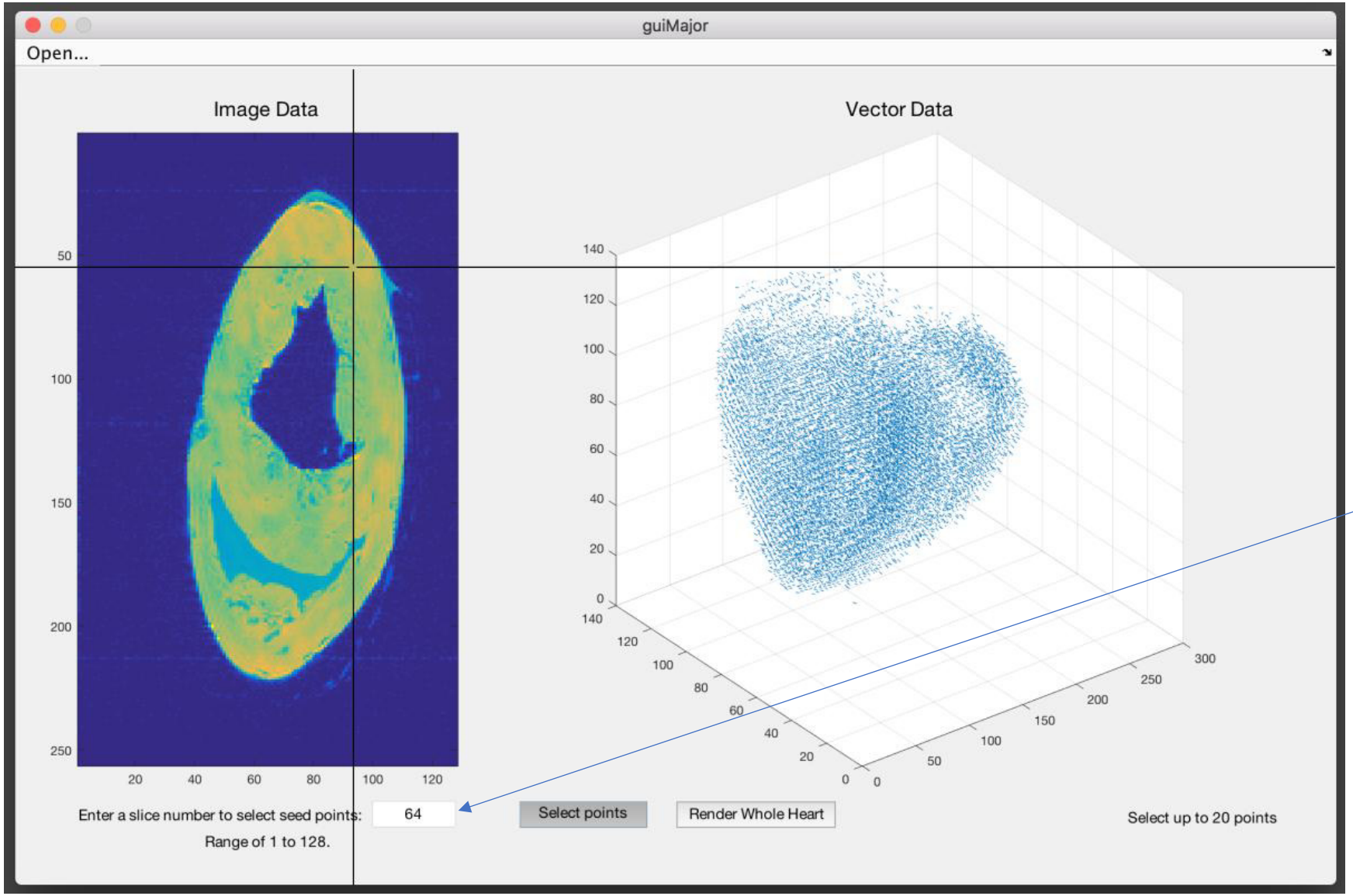


Image and vector data added



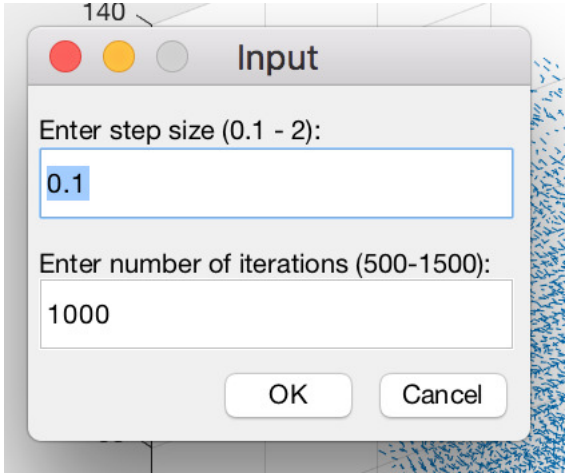
If you choose to select specific points, the following will appear:



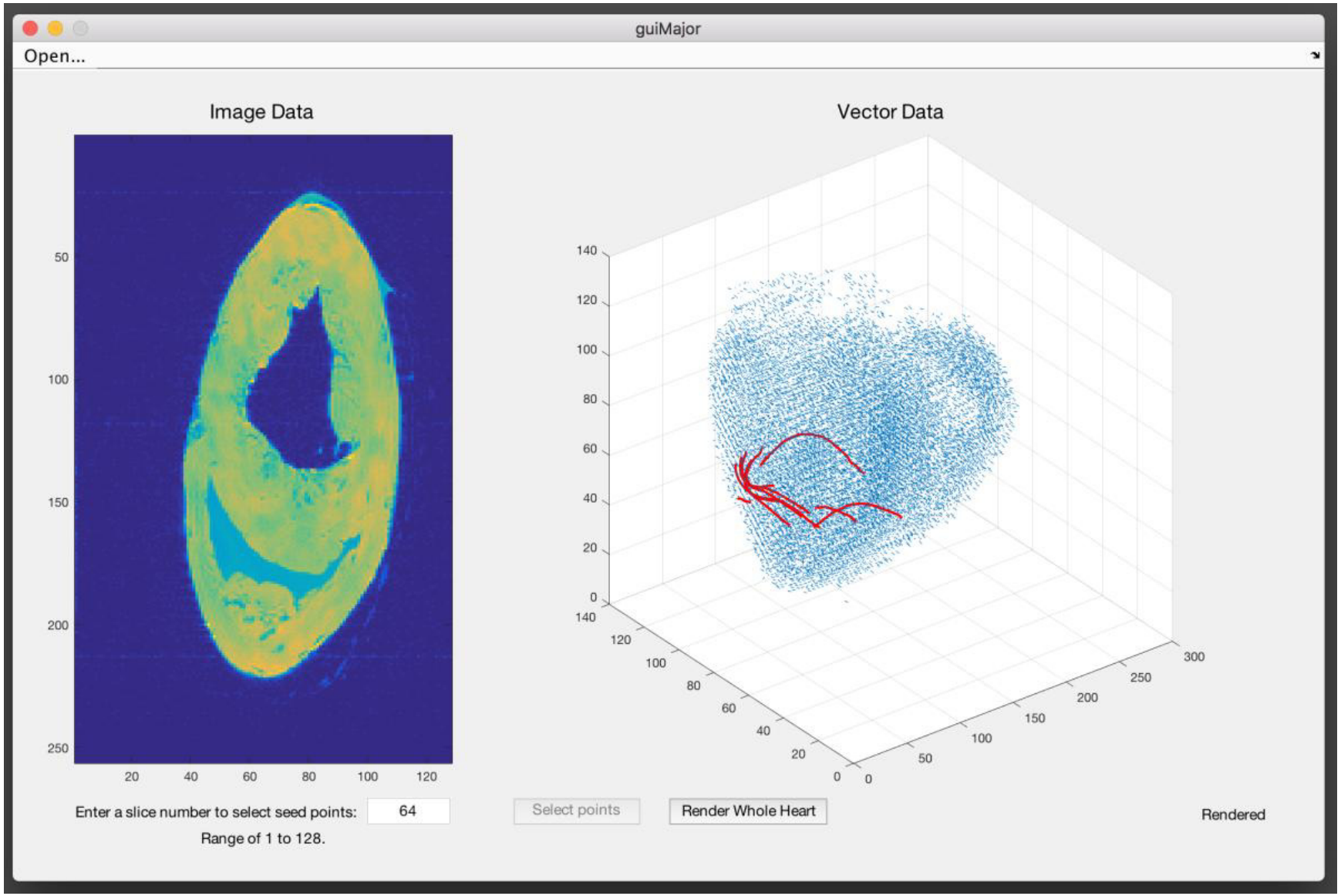


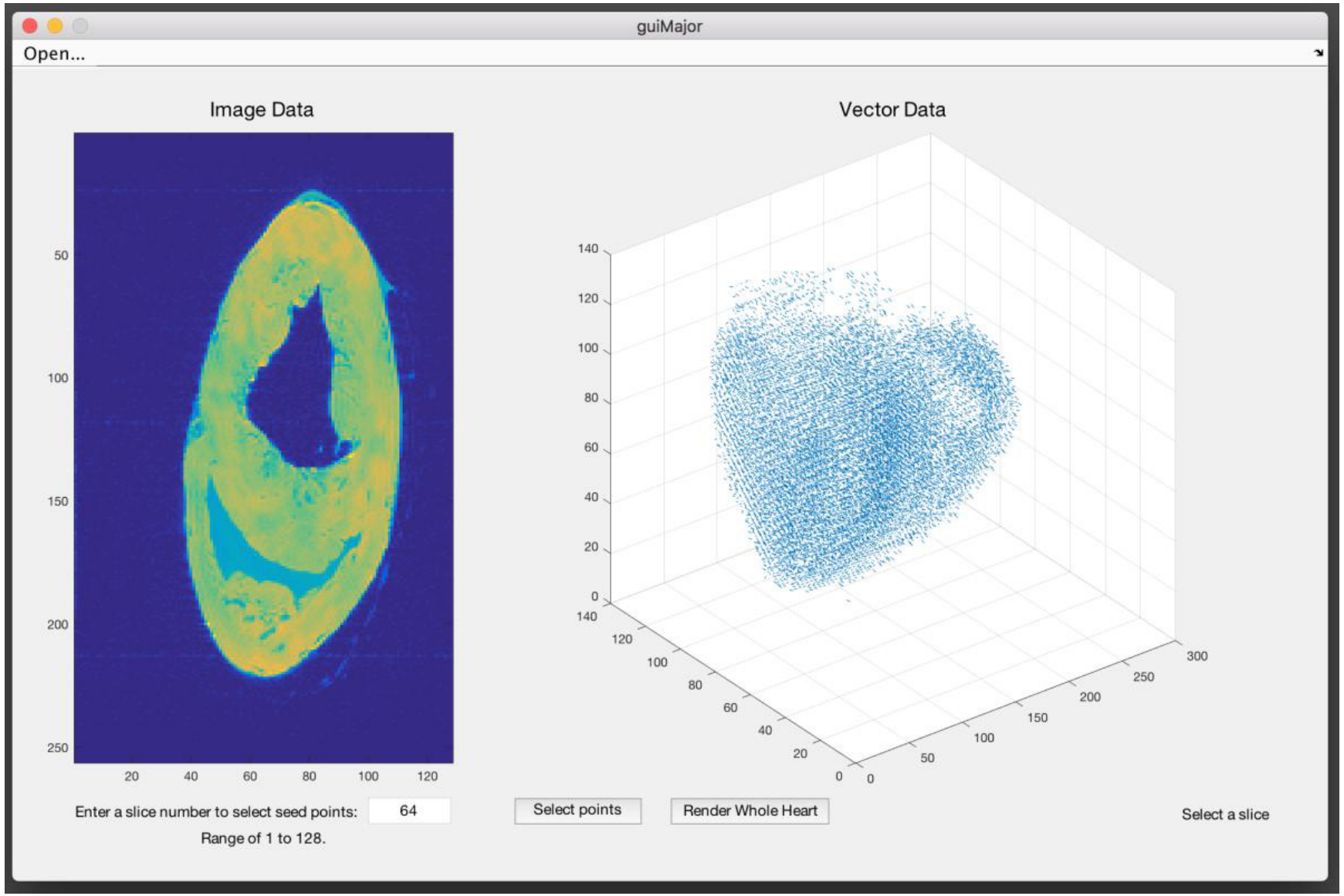
You can then select the desired number of points using ginput.

The input data will be the the x and y values as shown. The z value will be the slice number input here:

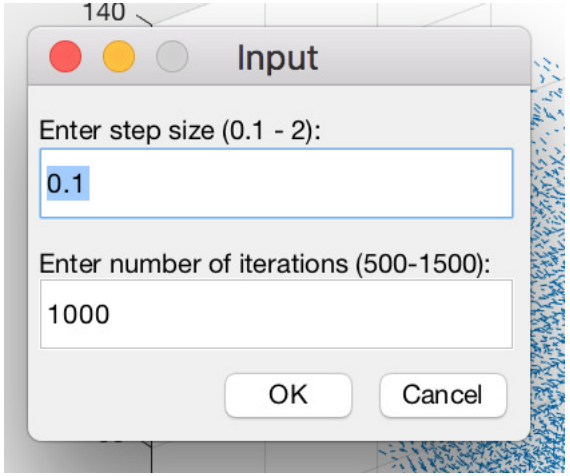


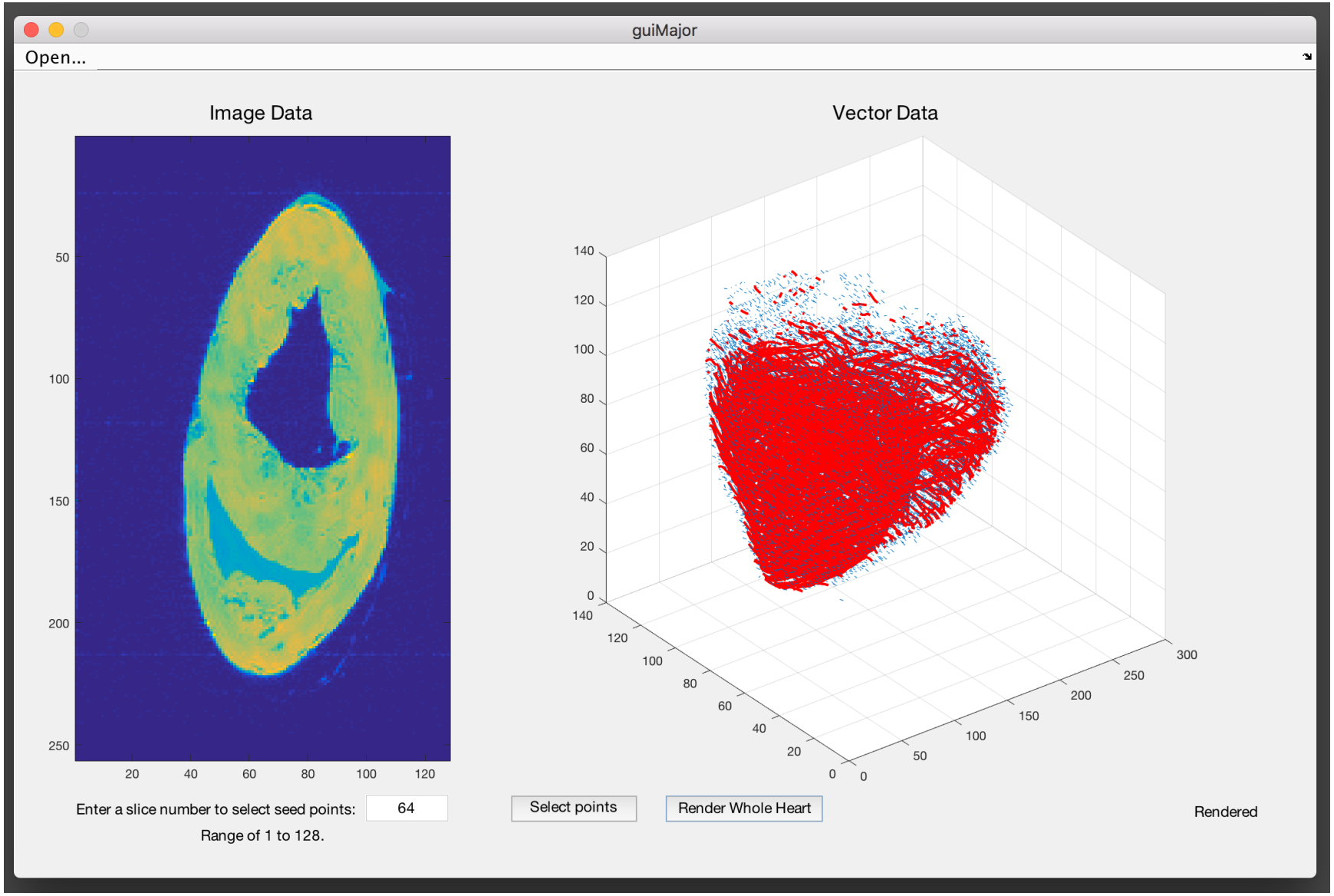
After entering the desired step size and number of iterations, a partial render appears on the vector data





Alternatively, a whole heart render will just ask for the step size and number of iterations:





Resulting in an image like this.

GUI functions

Application launched

```
% --- Executes just before guiMajor is made visible.
function guiMajor_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 guiMajor (see VARARGIN)

% Choose default command line output for guiMajor
% Update handles structure
handles.slicenum = 64; % sets the slice at 64
handles.ev = []; % establishes an empty handles.ev
handles.im = []; % establishes an empty handles.im
guidata(hObject, handles);
```

Add image data

```
function addimdata_Callback(hObject, eventdata, handles)
% hObject    handle to addimdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[name, folder] = uigetfile({'*.*','All Files (*.*)'
    '*.m;*.fig;*.mat;*.mdl', 'All MATLAB Files (*.m, *.fig, *.mat, *.mdl)';}, ...
    'Pick a file'); % opens file explorer / finder
if isequal(name,0) || isequal(folder,0)
    disp('User pressed cancel')
else
    disp(['User selected ', fullfile(folder, name)])
end % prints to Command Window what the user did
folderName = fullfile(folder,name); % get file path
dir(folderName); % add file path to directory
handles.file = fopen('IM','r','b'); % imports heart MRI
dataim = fread(handles.file, inf, 'float'); % record values
fclose(handles.file); % close file
handles.im = reshape(dataim,256,128,128); % set it to a handle
axes(handles.leftplot); % set the left plot
imagesc(handles.im(:,:,64)); % show image at handles.slicenum = 64
if isempty(handles.ev) % if vector field hasn't been added yet
    set(handles.output, 'String', 'Please select image data and vector data in the menu bar.',
'fontsize', 12);
    set(handles.slice, 'Enable', 'Off');
    set(handles.points, 'Enable', 'Off');
    % makes sure handles.output stays the same, makes sure handles.slice
    % and handles.points are inactive
else
    set(handles.output, 'String', 'Select a slice');
    set(handles.slice, 'Enable', 'On');
    set(handles.points, 'Enable', 'On');
    set(handles.renderall, 'Enable', 'On');

    % asks user to change slice if desired, and to choose points.
end
guidata(hObject,handles); % updates guidata
```

Add vector data

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

[name, folder] = uigetfile({'*.*','All Files (*.*)'
    '*.m;*.fig;*.mat;*.mdl', 'All MATLAB Files (*.m, *.fig, *.mat, *.mdl)';}, ...
    'Pick a file'); % opens file explorer / finder
if isequal(name,0) || isequal(folder,0)
    disp('User pressed cancel')
else
    disp(['User selected ', fullfile(folder, name)])
end % prints to the Command Window the choice of the user
folderName = fullfile(folder,name); % gets the file path
dir(folderName); % add file path to directory
handles.file2 = fopen('EV1','r','b'); % imports Vector field
datav = fread(handles.file2, inf, 'float'); % record values
fclose(handles.file2); % close file
handles.ev = reshape(datav, 3,256,128,128); % record to handles.ev
handles.u = squeeze(handles.ev(1,1:256,:,:)); % formats vector field to work with quiver3
handles.v = squeeze(handles.ev(2,1:256,:,:));
handles.w = squeeze(handles.ev(3,1:256,:,:));
[handles.X,handles.Y,handles.Z] = ndgrid(1:256,1:128,1:128);
% actual data is based off of the data above. The quiver plot displayed is
% based off of the following handles (for rendering time and ginput responsitivity):
handles.X2 = handles.X(1:4:256,1:4:128,1:4:128);
handles.Y2 = handles.Y(1:4:256,1:4:128,1:4:128);
handles.Z2 = handles.Z(1:4:256,1:4:128,1:4:128);
handles.u2 = handles.u(1:4:256,1:4:128,1:4:128);
handles.v2 = handles.v(1:4:256,1:4:128,1:4:128);
handles.w2 = handles.w(1:4:256,1:4:128,1:4:128);
axes(handles.rightplot) % will write to the right plot
quiver3(handles.X2,handles.Y2,handles.Z2,handles.u2,handles.v2,handles.w2);
axis square;
set(handles.output, 'String', 'Please select image data and vector data in the menu bar.',
'fontsize', 12);
if isempty(handles.im) % if handles.im has not been added yet
    set(handles.output, 'String', 'Please select image data and vector data in the menu bar.',
'fontsize', 12);
    set(handles.slice, 'Enable', 'Off');
    set(handles.points, 'Enable', 'Off');
    % make sure handles.output stays the same and that handles.slice and
    % handles.points remain inactive
else
    set(handles.output, 'String', 'Select a slice');
    set(handles.slice, 'Enable', 'On');
    set(handles.points, 'Enable', 'On');
    set(handles.renderall, 'Enable', 'On');

    % change handles.output and turn on handles.slice and handles.points
end
guidata(hObject,handles); % updates guidata
```

Executes with change in slice

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

% Hints: get(hObject,'String') returns contents of slice as text
%        str2double(get(hObject,'String')) returns contents of slice as a double

handles.slicenum = str2double(get(hObject,'String')); % in case anther value is entered
axes(handles.leftplot); % makes sure we are looking at the left plot
imagesc(handles.im(:,:,handles.slicenum)); % produce image
guidata(hObject,handles); % updates the handle's structure array
```

Executes with ‘Select points’

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

% Hint: get(hObject,'Value') returns toggle state of points
handles.status = get(hObject,'Value'); % returns whether or not handles.points has been pressed

prompt = {'Enter number of points:'};
dlg_title = 'Input';
num_lines = 1;
defaultans = {'20'};
answer = inputdlg(prompt,dlg_title,num_lines,defaultans);

axes(handles.leftplot); % looking at the left plot
while handles.status == 1 % if it has been pressed
    set(handles.output, 'String', 'Select up to 20 points'); % update handles.output
    [handles.ys,handles.xs] = ginput(str2double(answer{1})); % get 20 ginput points
    set(hObject,'Value',0) % turn handles.points off
    set(handles.points, 'Enable', 'Off'); % disable handles.points
    handles.status = get(hObject,'Value'); % exit the while loop
end

prompt = {'Enter step size (0.1 - 2):','Enter number of iterations (500-1500):'};
dlg_title = 'Input';
num_lines = 1;
defaultans = {'0.1','1000'};
result = inputdlg(prompt,dlg_title,num_lines,defaultans);

set(handles.output, 'String', 'Rendering...'); % update handles.output
axes(handles.rightplot); % switch to the right plot
hold on % keep the quiver plot
handles.xs = round(handles.xs); % round the crude data from ginput
handles.ys = round(handles.ys);

for ind = 1:length(handles.xs) % for as many points chosen
    myStream2(handles.xs(ind),handles.ys(ind),handles.slicenum,...
        handles.u,handles.v,handles.w, str2double(result{1}), str2double(result{2}));
    hold on % plot the streamlines according to myStream
end
set(handles.output, 'String', 'Rendered'); % update handles.output
hold off
guidata(hObject,handles); % updates guidata
```

Executes with ‘Render Whole Heart’

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

set(handles.points, 'Enable', 'Off'); % diables other option
set(handles.output, 'String', 'Rendering...'); % displays status in lower right

prompt = {'Enter step size (0.1 - 2):','Enter number of iterations (500-1500):'};
dlg_title = 'Input';
num_lines = 1;
defaultans = {'0.1','1000'};
result = inputdlg(prompt,dlg_title,num_lines,defaultans);

axes(handles.rightplot); % record the follwing in the right plot
for m = 1:9:256 % scans the entire MRI - periodically.
    for n = 1:9:128
        for o = 1:9:128
            if handles.im(m,n,o) > 50 % if above than 50, start streamline
                myStream2(m,n,o,handles.u,handles.v,handles.w, ...
                    str2double(result{1}), str2double(result{2})); % calls the function that draws the streamline
                hold on
            end
        end
    end
end
set(handles.output, 'String', 'Rendered'); %  update status
set(handles.points, 'Enable', 'On'); % turn on option to choose points again.
```

myStream – not used in final product

```
function [] = myStream(x,y,z,u,v,w)
% This function will plot streamline data on a ndgrid given vector fields
% defined by u, v, and w. The streamline will begin at point x, y, and z,
% and proceed in two directions, aided by dt and -dt.

% interpn allows us to interpolate the data in this dimension. we update
% points in s as we go along applying Euler's method.

[X,Y,Z] = ndgrid(1:256,1:128,1:128);
dt = 1; % set the step size. normally small
L = 200; % number of iterations
s = zeros(3,L);
s(:,1) = [x, y, z];  % initialization
Vt = zeros(1,3);
for n = 1:L-1
     if Vt < realmax
         Vt(1) = interpn(X,Y,Z,u,s(1,n),s(2,n),s(3,n));
         Vt(2) = interpn(X,Y,Z,v,s(1,n),s(2,n),s(3,n));
         Vt(3) = interpn(X,Y,Z,w,s(1,n),s(2,n),s(3,n));
         s(:,n+1) = s(:,n) + Vt'*dt;
         % records the next value of s based off of the current value of s
         % and the interpolated data multiplied by the step.
     else % breaks the loop if the data ever grows exponentially
         break;
     end
end
plot3(s(1,:),s(2,:),s(3,:),'r') % plot that data
s2 = zeros(3,L); % initialization
s2(:,1) = [x, y, z];
Vt = zeros(1,3);
for n = 1:L-1
     if Vt < realmax
         Vt(1) = interpn(X,Y,Z,u,s2(1,n),s2(2,n),s2(3,n));
         Vt(2) = interpn(X,Y,Z,v,s2(1,n),s2(2,n),s2(3,n));
         Vt(3) = interpn(X,Y,Z,w,s2(1,n),s2(2,n),s2(3,n));
         s2(:,n+1) = s2(:,n) + Vt'*-dt;
         % this time we do the same as above, except that we apply a
         % negative step to grow the streamline in the other direction.
     else % breaks the loop if the data ever grows exponentially
         break;
     end
end
hold on;
plot3(s2(1,:),s2(2,:),s2(3,:),'r') % plot that data
axis square;
```

We found that interpn was not only inefficient, but insufficient for our needs. We decided to change it to local vector evaluation in myStream2, and in turn the rendering time decreased.

myStream2 – only using local vector data

```
function [] = myStream2(x,y,z,u,v,w, dt, L)
% This function will plot streamline data on a ndgrid given vector fields
% defined by u, v, and w. The streamline will begin at point x, y, and z,

s(:,1) = [x y z];  % initialization of s, which will become our line in the end
for n = 1:L-1
    if round(s(1,n)) >= 256 || round(s(2,n)) >= 128 || round(s(3,n)) >= 128
        break
    else
        Vt(1) = u(round(s(1,n)),round(s(2,n)),round(s(3,n))); % get x component of vector
        Vt(2) = v(round(s(1,n)),round(s(2,n)),round(s(3,n))); % get y component of vector
        Vt(3) = w(round(s(1,n)),round(s(2,n)),round(s(3,n))); % get z component of vector
    end
    if Vt(1) == 0 || Vt(2) == 0 || Vt(3) == 0 % if any of those values is 0
        break % end loop
    else
        Vt = Vt / sqrt(Vt(1).^2 + Vt(2).^2 + Vt(3).^2); % make the vector one unit length
    end

    if n ==1 % only on the first iteration
        s(:,n+1) = s(:,n) + Vt'*dt; % update via Eulers
    elseif Vtprev*Vt' > -0.75 && Vtprev*Vt' < 0.75 % if the dot product is close to 0
        break % end loop
    elseif Vtprev*Vt' <= -0.75 % if the vector is pointing in almost the other direction
        Vt = -Vt; % invert the vector direction and
        s(:,n+1) = s(:,n) + Vt'*dt; % update via Eulers
    else % if the vector is pointing in the same general direction, then
        s(:,n+1) = s(:,n) + Vt'*dt; % update via Eulers
    end
    Vtprev = Vt; % record this round's vector components for comparison in the next round.
end
hold on
plot3(s(1,:),s(2,:),s(3,:),'Color','r','LineWidth',2) % plot that data
s2(:,1) = [x, y, z];
for n = 1:L-1
    if round(s2(1,n)) > 256 || round(s2(2,n)) > 128 || round(s2(3,n)) > 128
        break
    else
        Vt(1) = u(round(s2(1,n)),round(s2(2,n)),round(s2(3,n))); % get x component of vector
        Vt(2) = v(round(s2(1,n)),round(s2(2,n)),round(s2(3,n))); % get y component of vector
        Vt(3) = w(round(s2(1,n)),round(s2(2,n)),round(s2(3,n))); % get z component of vector
    end
    if Vt(1) == 0 || Vt(2) == 0 || Vt(3) == 0 % if any of those values is 0
        break % end loop
    else
        Vt = Vt / sqrt(Vt(1).^2 + Vt(2).^2 + Vt(3).^2); % make the vector one unit length
    end

    if n ==1 % only on the first iteration
        s2(:,n+1) = s2(:,n) + Vt'*-dt; % update via Eulers
    elseif Vtprev*Vt' > -0.75 && Vtprev*Vt' < 0.75 % if the dot product is close to 0
        break % end loop
    elseif Vtprev*Vt' <= -0.75 % if the vector is pointing in almost the other direction
        Vt = -Vt; % invert the vector direction and
        s2(:,n+1) = s2(:,n) + Vt'*-dt; % update via Eulers
    else % if the vector is pointing in the same general direction, then
        s2(:,n+1) = s2(:,n) + Vt'*-dt; % update via Eulers
    end

    Vtprev = Vt; % record this round's vector components for comparison in the next round.
end
hold on
plot3(s2(1,:),s2(2,:),s2(3,:),'Color','r','LineWidth',2) % plot that data
axis square;
```

Potential code improvements

- This code was written to specifically handle data provided from the sponsor. Ideally, we would want it to accommodate any size of data to analyze, including raw MRI data.
- The 3D render could further be rendered to provide shading for improved visualization. It could also be manipulated by the user (zoom, spin, etc)
- Allow further customization by allowing the user to modify line color and width. This could be added to the dialog box that already asks for the step size and number of iterations.