```
clear all;
load('Data161130.mat');
%figure(1);

rawData = DataArray; %raw data set name
%angularResolution = 2; %degrees per sample
depthOfNeedle = 30; %in millimeters
dataAngleSet = DataAngle2;
minAngle = 0;
maxAngle = max(dataAngleSet);


FS = 80e6; %%KAI
Filt.BPF = fir1(100, [8e6/(FS/2) 12e6/(FS/2)]); %%KAI

%%variables
samplesPerLine = size(rawData, 1);
numberOfLines = size(rawData, 2);
delayBetweenCollections = 0; %delay between "collections" of A-line
 data
displayDataArray = ones(samplesPerLine, numberOfLines); %array to add
 to to display data


%variables needed for scan conversion
speedOfSound = 1500; %m/s
samplingRate = 10^8; %rate of collection of data (samples/sec)
angleDist =20; %angle distance between A-lines (degrees)
sampleSpacing = (speedOfSound/samplingRate) * 1000; %spatial
 separation between samples in A-line (millimeters)
%maxAngle = (numberOfLines/2)*angularResolution + minAngle;%maximum
 relative angle (degrees) %NEEDS CHANGE
minAngle = min(dataAngleSet);
maxAngle = max(dataAngleSet);
middleAngle = (minAngle + maxAngle)/2;
maxDistance = sampleSpacing*samplesPerLine; %maximum distance of
 sample from needle (millimeters)

%%create empty cartesian matrix
ySize =  ceil(maxDistance + depthOfNeedle); %in mm y size of cartesian
 matrix; ideally would have depth of needle to be able to find more
 accurate size
xSize = ceil(2*maxDistance*sind(maxAngle - middleAngle)); %x size of
 cartesian matrix

cartesianMatrix = ones(110*xSize, 3*ySize-50); %creates cartesian
 matrix which will be displayed
%%figure variables
figure(2);

%%beamforming variables
```

```
windowSize = 77; %must be odd
apodWindow = chebwin(windowSize);
beamformedData = zeros(samplesPerLine, numberOfLines);
count = ones(samplesPerLine, numberOfLines);



%%creating set of Lines Left to Sample
 %random order sample of Alines
for l = 1:numberOfLines %reading A-lines one by one

    beamformedData(:, l) = rawData(:, l);
    %iterate through window for this a line
    for w = l - ((windowSize-1)/2): l + ((windowSize-1)/2)
        %make sure adjacent a line is valid
        if w > 0 && w <= numberOfLines
           width = depthOfNeedle*sind(abs(l-w)*angleDist);
            %iterate down a line
            for s = 1:samplesPerLine
                    beamformedData(s, w) = beamformedData(s, w) +
 apodWindow(w - l + (windowSize - 1)/2 + 1)*rawData(s, l);
                    count(s, w) = count(s, w) + 1;
            end
        end

    end
    beamformedData = beamformedData./count;



    currentALine = (beamformedData(:, l));
    currentALine = convn(currentALine, Filt.BPF', 'same');
```

Scan Conversion stuff %%

```
    %NEEDS CHANGE

  for W = l - ((windowSize-1)/2): l + ((windowSize-1)/2) %iterate
 through backprojection window

        if W > 0 && W <= size(beamformedData, 2)
        currentALine = (beamformedData(:, W));
        currentALine = convn(currentALine, Filt.BPF', 'same');
        currentAngle = middleAngle - dataAngleSet(W);%middle angle
 - current angle or angle from midline (left is positive, right is
 negative)
            for j = 500:samplesPerLine
            %horizontal calculations
            distanceFromProbe = sampleSpacing*j; %straight line
 distance from probe
            xDistanceFromMiddle = (distanceFromProbe
+depthOfNeedle)*sind(currentAngle); %horizontal distance of sample
 from midline
```

```
            xCoordinate = round((xSize/2) - xDistanceFromMiddle);
            %vertical calculations
            yCoordinate = round(100*((depthOfNeedle +
 distanceFromProbe)*cosd(currentAngle)));
            cartesianMatrix(yCoordinate - 2500, xCoordinate+100) =
 currentALine(j);
            end
        end
    end
```

figure stuff %%

```
    %display raw data in polar coordinates
    %figure(1);
    %imagesc(db(displayDataArray(500:end,:))); %%KAI
    %imagesc(displayDataArray);
    %colormap(gray);
    %hold on;


    %display raw data in cartesian coordinates
    %drawnow
    %dn = 1

end
matrixString = sprintf('SCBFwindow%d.mat', windowSize);
save(matrixString, 'cartesianMatrix');

%imagesc(db(cartesianMatrix(1000:end,120:160)));
imagesc(db(beamformedData(:, 1:120)));
    colormap(gray);
    hold on;
```

```
Warning: Variable 'ardui' originally saved as a arduino cannot be
 instantiated
as an object and will be read in as a uint32.
```

Published with MATLAB® R2016a