GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

ECE 6272 FALL 2010

COMPUTER (Mini)-PROJECT #5

Assigned:   
Tuesday, November 23, 2010

**Due Date for On-Campus Students: Tuesday, Dec. 7 @ 9:35 AM**

**Due Date for DLPE Students: Tuesday, Dec. 14 @ 4:00 PM Eastern Time**

* This project is to be done *individually.* Each student must develop his or her own computer code in its entirety. Students are not to discuss the theory or approaches to coding the theory with one another, nor are they to assist in debugging each other’s work. You may ask Dr. Richards questions regarding theory and implementation of the project, including asking them at the beginning or end of class, when others can benefit as well.
* MATLAB is the preferred language, but others are acceptable; the point is to try the experiments, not to improve your MATLAB skills.
* Data required for this project are available for download from the class T-Square site in the Computer Projects 🡪 DBS Project area. ***Whether you begin working right away or not, be sure you download your data set and make sure you can load it into MATLAB (or other computing environment) as soon as possible to avoid last minute difficulties.***
* Reports will be graded on completeness in addressing the assignment and quality of results. The main table of results is a large fraction of the grade. They will not be graded on programming style or efficiency or on writing quality, except that the programming and the writing should be clear enough to be reasonably understandable. Questions or clarifications about the assignment should be directed to Dr. Richards.[[1]](#footnote-1) Errata, revisions and hints (if any) will be made available via the class T-Square site or during class.

# PROBLEM

You are given two fast-time/slow-time data matrices collected by a sidelooking imaging radar viewing a scene consisting of 9 points scatterers in a 3x3 grid. The two files are named dbs\_data\_6272\_easy and dbs\_data\_6272\_hard. You are also given a mostly-complete MATLAB program (m-file), procSARdata\_DBS\_6272 that forms an image from this data using the Doppler Beam Sharpening technique of Sections 8.3.1 and 8.3.2 of the text. The data and M files are in a single zip archive titled DBS\_project.zip in the Computer Projects 🡪 DBS Project area of the T-Square web site.

The file dbs\_data\_6272\_easy represents a scenario where the central reference point (CRP) of the imaged area is nominally 50 km from the aircraft, and we are trying to obtain 50x50 meter resolution in range and cross-range. The file dbs\_data\_6272\_hard is a more challenging scenario with a 10 km nominal range to the CRP and a goal of 5x5 meter resolution.

Your job is to demonstrate DBS image formation with each file and the provided code. One of the images should show cross-range blurring. You are then asked to modify the data by applying azimuth dechirp and demonstrating that this improves the focus. Finally, you are asked to demonstrate that geometric distortion can be fixed by the code provided.

# procSARdata\_DBS\_6272.m

This file is a straightforward implementation of Doppler Beam Sharpening in accordance with the equations in the text. A copy of the code is included at the end of this assignment. Notice that the code includes a “user input” section in lines 13-23. This section contains two logical switches, which should initially be false, and an oversampling parameter (this controls the size of a later FFT). An integer value of at least 3 is adequate to answer the questions in this assignment, but a larger value (closer to 10) will give you better definition of point target response nulls. However, with larger value, memory requirements will grow and run time will increase. If it takes too long to run, or you get “out of memory” messages, reduce the oversampling factor.

Lines 28 and 29 load one of the two data sets. Comment out one of them to load the desired data set.

Read through the code to see how it is structured. After spitting out all of the simulation parameters, there is an azimuth dechirp section that is commented out. We’ll return to this shortly.

Next up is pulse compression in range. We are using an LFM waveform, so this is exactly the same as was done in the Pulse Doppler project. In this case, I am doing the convolution in the frequency domain, and not using a window. I delete the initial transient from the filtering. The result of this phase is displayed in Figure 1. This data will be compressed in range, but still very spread out in cross-range.

Next up is the cross-range “compression”, which in DBS is simply an FFT on each row. The Doppler frequency axis is scaled to cross-range units using Eqn. (8.36). In this code, all range bins are scaled using the value of the range at the CRP.

The next section is controlled by the fix\_geometry switch at the beginning. If executed, it fixes the “bowing” due to cross-range scaling and the range curvature, both by linear interpolation in the appropriate dimension. First up is the cross-range processing, which consists of rescaling the data in each range bin according to the actual range to that bin, rather than using the range to the CRP for all bins. Next is the curvature correction, which requires shifting each column in the fast-time (range) dimension to straighten out the curvature.

# REQUIREMENTS

You must submit a “report” of your findings that includes only the following items, in this order, and on the pages stated:

1. Page 1: cover page with your name.

2. Page 2: A figure showing the entire formed image using dbs\_data\_6272\_easy, and another figure showing the entire formed image using dbs\_data\_6272\_hard. Label which is which. An example of one of these figures is shown on the next page.

3. Page 3: For each case (“easy” or “hard”), compute the required aperture time *Ta* at the CRP and LFM pulse bandwidth to obtain the desired cross-range and range resolution using appropriate equations from the text. Use an aircraft velocity *v* = 150 m/s. (This is what was used in the simulation that created the data.) State the actual aperture time and pulse bandwidth of the data, as reported by procSARdata\_DBS\_6272 when you run it on the data file. State whether the aperture time and bandwidth requirements you computed are met by the data. Also, compute the test of Eqn. (8.41) in the text for each case and state whether it is satisfied at the CRP.

|  |  |
| --- | --- |
|  |  |
| *Example of DBS Image of the Entire Scene.* | *Example of zoon on the central scatterer, showing 50 m peak-to-null resolution in both dimensions.* |

4. Page 4: A zoom (just using the MATLAB figure window zoom tools will be fine) of the center scatterer in each case. Include an area big enough to cover at least the first two nulls of the sidelobes on both sides of the peak in each dimension. State the actually achieved Rayleigh (peak-to-null) resolution for each case, based on these plots.

5. Page 5: The remainder of this mini-project deals only with the hard case data. Set the dechirp flag to true. Modify the code to insert a phase correction where ??? appears in line 64. (You are referred to Eqn. (8.42) in the text. Note that, when you load the dbs\_data\_6272\_hard data, a vector u is created that may be useful in implementing this equation.) You can restructure this code block if you wish (for instance, to pull out of the loop portions of the calculation that do not depend on *R*). Or not. Provide the modified line(s) of code, showing how you implemented the azimuth dechirp. Provide an image of the entire imaged scene and a zoom on the center scatterer to show that the desired 5 m resolution is now obtained in both dimensions.

6. Page 6: Enable the fix\_geometry section of code (this will run much slower now) and run it on the dbs\_data\_6272\_hard data with dechirp enabled as well. On this page, provide two figures: one zoomed in to show all of the 9 scatterers (but with most of the empty space in cross-range clipped out) with dechirp but without interpolation (this is the result from Page 5 again, just partly zoomed in), and the same portion of the scene after the geometry is corrected. If all goes well, you’ll have 9 well-focused scatterers at the combinations of *x* and *y* locations of +1 km, 0 km, and -1 km relative to the scene center.

*That’s it, you’re done!* I’ve included a program called makeSARdata in the zip file as well so, if you wish, you can play around with creating data sets with various parameters and imaging them with this or other algorithms.

Listing of procSARdata\_DBS\_6272.m

%

% procSARdata\_DBS\_6272v3

%

% process SAR project data using DBS algorithm

%

% Written by M. A. Richards

% December 2010

%

clear all; close all; hold off

format compact

%#######################################################

% User input section ###################################

% algorithm control parameters

dechirp = false; % use azimuth dechirp step or not

oversample\_freq = 5; % oversampling in Doppler; bigger makes better

% picture but needs more memory and time

fix\_geometry = false; % perform geometric corrections or not

% End user input section ###############################

% ######################################################

% get data and dislay its characteristics. Un-comment one of the two

% following lines to load the desired data set

% load dbs\_data\_6272\_hard;

load dbs\_data\_6272\_easy;

fprintf(['\rDESCRIPTION OF DATA\r\r']);

fprintf('\nRange resolution = %g meters',DR);

fprintf('\nCross-range resolution = %g meters',DCR);

fprintf('\nRange to central reference point = %g km',Rcrp/1e3);

fprintf('\nAperture time = %g seconds',Ta);

fprintf('\nRange curvature at CRP = %g meters\n\n',Rc);

fprintf('\rPulse length = %g microseconds\r',tau/1e-6);

fprintf('\rChirp bandwidth = %g Mhz\r',W/1e6);

fprintf('\rTime-bandwidth product = %g',W\*tau);

fprintf('\rFast time sampling rate = %g Msamples/sec\r',fs/1e6);

fprintf('\rWe are simulating %g pulses at an RF of %g GHz',Npulses,fc/1e9);

fprintf('\r and a PRF of %g kHz, giving a PRI of %g usec.',PRF/1e3,PRI/1e-6);

fprintf('\nRange resolution = %g meters',DR);

fprintf('\nCross-range resolution = %g meters',DCR);

fprintf('\nRange to central reference point = %g km',Rcrp/1e3);

fprintf('\nSwath length = %g m',Ls);

fprintf('\nRange curvature at CRP = %g meters\n\n',Rc);

fprintf('\rThe range window limits are %g to %g usec.\r', ...

T\_out(1)/1e-6,T\_out(2)/1e-6);

fprintf('\rThe range window starts at %g km.',c\*T\_out(1)/2/1e3);

fprintf('\rThe range window ends at %g km.',c\*T\_out(2)/2/1e3);

fprintf('\rThere are %g scatterers with the following coordinates:', ...

Ntargets);

for nt = 1:Ntargets

fprintf('\r x=%+8.1f m, R=%+8.1f m',coords(nt,1),coords(nt,2) );

end

tic

% Process the data

R = c\*t\_y/2; % vector of range bin centers

% Pulse compression first. Doing this in the frequency domain. No window.

Ns = length(s);

h = conj(s(Ns:-1:1));

Nconv = Nr+Ns-1;

H = fft(h,Nconv);

fprintf('\n\nBeginning fast-time processing\n')

for m = 1:Npulses

if (mod(m,100)==1)

disp([' ... compressing pulse #',int2str(m),' of ',int2str(Npulses)])

end

ypc = ifft(fft(y(:,m),Nconv).\*H);

y(:,m) = ypc(Ns:end);

end

figure(1)

ydisp = db(max(abs(y),eps),'voltage'); % adding machine epsilon provents a "log of zero" warning later

ydisp = max(ydisp,max(ydisp(:))-60);

imagesc(u,(R-Rcrp)/1e3,ydisp);

xlabel('aperture position (m)');

ylabel('range relative to CRP (km)');

title('After Range Compression, Before Cross-Range Compression')

grid;

% Now azimuth "compression", which is just a DFT in DBS. Initially, scale

% all range bins using the range to the CRP. Will fix this simplification

% later. Dechirp the data first if that option selected.

% if dechirp

% for n = 1:Nr

% y(n,:) = y(n,:).\*exp(j\*(4\*pi/lambda)\*u.^2/2/R(n));

% end

% end

fprintf('\n\nBeginning slow-time processing\n')

Nf = oversample\_freq\*Npulses;

scale\_cr = Rcrp\*lambda/2/v\*((0:Nf-1)/Nf-0.5)\*PRF;

ysar = zeros(Nr,Nf);

for n = 1:Nr

if (mod(n,100)==1)

disp([' ... compressing range bin #',int2str(n),' of ',int2str(Nr)])

end

ysar(n,:) = fftshift(fft(y(n,:),oversample\_freq\*Npulses));

end

figure(2)

ydisp = db(max(abs(ysar),eps),'voltage');

ydisp = max(ydisp,max(ydisp(:))-60);

imagesc(scale\_cr/1e3,(R-Rcrp)/1e3,ydisp);

xlabel('cross-range (km)');

ylabel('range relative to CRP (km)');

title('Fully Compressed Image')

grid;

if (fix\_geometry)

% now resample to square things up. Will work just on the linear scale

% magnitude image now; don't need complex data anymore.

fprintf('\n\nBeginning cross-range resampling\n')

ysarmag = max(abs(ysar),eps);

% find cross-range coordinates at narrowest spot (min range). We will

% interpolate all range bins to this narrowest grid so that we will not

% have to interpolate outside of the cross-range extent for which we

% have data at any range.

Rmin = R(1);

min\_scale\_cr = Rmin\*lambda/2/v\*((0:Nf-1)/Nf-0.5)\*PRF;

% now loop through the range bins, interpolating the cross-range

% magnitude data at each range bin to the same cross-range sample

% positions given in min\_scale\_cr

ysarmag\_rs1 = zeros(size(ysarmag)); % preallocate to increase speed, save memory

for n = 1:Nr

if (mod(n,100)==1)

disp([' ... resampling range bin #',int2str(n),' of ',int2str(Nr)])

end

Rn = R(n);

n\_scale\_cr = Rn\*lambda/2/v\*((0:Nf-1)/Nf-0.5)\*PRF;

ysarmag\_rs1(n,:) = interp1(n\_scale\_cr,ysarmag(n,:),min\_scale\_cr,'linear',NaN);

% cubic interpolation in the above line is somewhat better quality, but

% much slower and not worth it just for display purposes

end

figure(3)

ydisp\_rs1 = db(max(ysarmag\_rs1,eps),'voltage');

ydisp\_rs1 = max(ydisp\_rs1,max(ydisp\_rs1(:))-60);

imagesc(min\_scale\_cr/1e3,(R-Rcrp)/1e3,ydisp\_rs1);

xlabel('cross-range (km)');

ylabel('range relative to CRP (km)');

title('Resampled Image')

grid;

% It's a little more complicated in range, since we need a

% cross-range-dependent shift in range, not just a linear scaling of the

% axis like we had in cross-range.

fprintf('\n\nBeginning range shifting\n')

ysarmag\_rs2 = zeros(size(ysarmag\_rs1)); % preallocate to increase speed, save memory

% Cycle through the cross-range positions

for m = 1:Nf

if (mod(m,100)==1)

disp([' ... shifting cross-range bin #',int2str(m),' of ',int2str(Nf)])

pause(0.01)

% this 'pause' allows output to comand window to update; wasn't

% happening otherwise for some reason, even though it worked

% fine in the three other similar segments above without the pause.

end

% for this cross-range position, do the nonlinear range shift

Ru = sqrt(R.^2 + min\_scale\_cr(m)^2);

for n = 1:length(R)

nu = find(R>=Ru(n),1,'first'); %finds closest nu such that R(nu) >= Ru(n)

if isempty(nu)

ysarmag\_rs2(n,m) = NaN;

else

% This line moves into the current range bin the nearest

% range bin in the warped image, i.e. the shift is limited

% to integer numbers of ragne bins. Relatively fast, but

% crude.

ysarmag\_rs2(n,m) = ysarmag\_rs1(nu,m);

% This line does a linear interpolation between the range

% bin data above and one range bin earlier, so it improves

% the result. However, it runs about 28x slower! If you

% want to try this, comment out the line above and

% uncomment this one.

% ysarmag\_rs2(n,m) = interp1(R,ysarmag\_rs1(:,m),Ru(n),'linear',NaN);

end

end

end

figure(4)

ydisp\_rs2 = db(max(ysarmag\_rs2,eps),'voltage');

ydisp\_rs2 = max(ydisp\_rs2,max(ydisp\_rs2(:))-60);

imagesc(min\_scale\_cr/1e3,(R-Rcrp)/1e3,ydisp\_rs2);

xlabel('cross-range (km)');

ylabel('range relative to CRP (km)');

title('Resampled and Range-Shifted Image')

grid;

end % end of "fix\_geometry" condition

toc

1. Office: Klaus 3354, 404-894-2714, <mark.richards@ece.gatech.edu>. Office hours TBD, but drop-ins and appointments welcome. [↑](#footnote-ref-1)