Coffee Can Radar

Synthetic Aperture Radar Simulation

The simulation will use equations derived in the paper, “CCR SAR Processing.docx” and all equation numbers cited in this document refer to the equation numbers in “CCR SAR Processing.docx,” which will be abbreviated, “CCR SAR”.

**SARsim.m**

INITIALIZATION

Start by initializing parameters for the transmit signal. These parameters should be measured using the AD2 oscilloscope connected to the Coffee Can Radar hardware.

f0 = 2296.3\*1e6; % center frequency: 2.2963 GHz

c = 3e8; % speed of light: m/s

BW = 135.7\*1e6; % chirp bandwidth: 135.7 MHz

tau = 0.020; % pulse width: 20 ms

Initialize parameters for synthetic aperture collection:

Rmax = 200; % meters

R0 = 20; % m – focal point range

Lf = 10; % synthetic aperture [feet]

L = Lf \* 0.3048; % synthetic aperture [m]

d = 2; % distance between steps [in]

delL = d/12\*.3048; % distance between steps [m]

numSteps = round(L/delL,0); % number of steps across synthetic aper.

The wavelength of the signal is calculated from the initialization data:

lambda = c/f0; % wavelength: m

Create the vector, R\_param for passing the appropriate initialization data to other functions:

% create a vector of init values that can be easily passed to singlePulseProc2.m

R\_param = [f0, BW, tau, c, Noise\_Amp, Rmax];

Specify the location of a point target referenced from the focal point at R0.

% target location, (xt, yt) from the synthetic aperture focal point.

tgtPos = [1.2, 0]; % range from scene focal point at range, R0

xt = tftPos(1);

yt = tgtPos(2);

% random additive noise amplitude to be used

Noise\_Amp = 0;

COLLECT SYNTHETIC APERTURE RECEIVE SIGNALS

Create a definite loop that will loop once for every step across the synthetic aperture.

for jj = 1:numSteps

...

end

Within the definite loop, calculate the range to the target located at , using the geometry shown in Figure 1 of “CCR SAR”.

tgtR = sqrt((R0+yt)^2 + (-L/2+delL\*(jj-1)-xt)^2);

Get the range processed target return using the MATLAB function, **singlePulseProc2.m**. The function returns the intermediate frequency (IF) return signal (sif), the Fourier Transform of the IF return signal (SIF), the calculated range from the processed return (Rcalc), and the range axis vector to use in plotting the result (R\_axis). The specific data in each variable is given below

sif: A matrix containing the IF return signal. Each row is the IF return signal from a single pulse of pulsewidth, tau.

SIF: A matrix containing the Fourier Transformed (pulse compressed) IF return signal. Each row is the IF return signal from a single pulse of pulsewidth, tau. A point target at location (xt, yt) will show up in each row as a sinc function located at range = .

Rcalc: A vector containing the range of the detected point targets at each of the steps across the synthetic aperture.

R\_axis: A vector containing the range of the detected target at each of the steps across the synthetic aperture.

**singlePulseProc2.m**

The list of target ranges to each point target and the initialization data is passed to this function to range process a single transmitted single pulse with returns from each target.

INITIALIZATION

The initialization data passed to this function is given below:

f0 = 2296.3\*1e6; % center frequency: 2.2963 GHz

BW = 135.7\*1e6; % chirp bandwidth: 135.7 MHz

tau = 0.020; % pulse width: 20 ms

c = 3e8; % speed of light: m/s

Noise\_Amp = 0; % random additive noise amplitude

Rmax = 200; % meters

The list target ranges is passed to this function in the vector, tgtR. While **SARsim.m** only calculates a single target range, it could be updated to include more than one target. The number of targets is given as:

Ntgt = length(tgtR);

GENERATE LINEAR FM WAVEFORM

Since the transmitted signal is a function of time the sample rate must satisfy the Nyquist criteria with a sample rate at least twice the highest frequency.

and so the sample spacing of the time varying signal will be :

deltRF = 1/(2\*f0);

The values of time at each sample is given by:

tRF = 0:deltRF:tau;

The phase of the transmitted signal is given by equation [1] in “CCR SAR”:

[1]

So, in MATLAB, it will be:

phi = 2\*pi\*((BW/tau/2)\*tRF.^2+(f0-BW/2)\*tRF);

and the transmitted signal will be:

sigOut = exp(i\*phi);

Notice we assume the transmitted signal amplitude is 1 and the constant, , in the phase equation is

The length of the sigOut vector is

sigLen = length(sigOut);

GENERATE RETURN SIGNAL CONTAINING RETURNS FROM EACH TARGET

The return signal is just the transmitted signal delayed by However, simply shifting the transmit signal would increase its length, so we must first create sigRet filled with zeros with the proper length. The longest delay the signal will see is the delay from a target at the maximum range, Rmax, and so the maximum time shift is:

2\*Rmax/c

and so the maximum index shift is:

Nshiftmax = round(2\*Rmax/c/deltRF,0);

The MATLAB code to initialize sigRet with zeros of the proper length is therefore:

sigRet = zeros([1 sigLen+Nshiftmax]);

Now, we set up a loop to add the signal return from each target range contained in tgtR:

for kk = 1:Ntgt

…

end

Inside the FOR loop, for each kk:

Calculate the round-trip time for the target:

rtt = 2\*tgtR(kk)/c;

Calculate the corresponding index shift for that round-trip time:

Nshift = round(rtt/deltRF,0);

Create a temporary return signal for this target that contains zeros and is the same size as sigRet:

sigtemp = zeros(size(sigRet));

Place the transmit signal, attenuated by in sigtemp starting at index Nshift:

sigtemp(Nshift+1:sigLen+Nshift)=tgtR(kk)^-3\*sigOut;

Add the temporary return signal to sigRet:

sigRet = sigRet + sigtemp;

The vector sigRet now contains the complex sinusoid coming from each target whose range is contained in tgtR.

SIMULATE THE RF MIXER

The signal received by the receive antenna enters the mixer where its complex conjugate is mixed with the transmit signal, represented by Equation [3] in “CCR SAR”.

[3]

To accomplish this, we need to pull out the non-zero elements of sigRet and the corresponding elements of sigOut, as shown in the figure below. The worst case shift is Nshiftmax, so we’ll pull out the elements between Nshiftmax and sigLen to guarantee we have non-zero values when mixing the two signals.

Line chart

Description automatically generated

sigIF = sigOut(Nshiftmax+1:sigLen).\*conj(sigRet(Nshiftmax+1:sigLen));

The variable, sigIF, now contains the IF return signal, which is much lower bandwidth than the RF signal, and can therefore, be downsampled to much fewer number of samples.

The sample spacing of the RF signal was,

The sample spacing of the IF signal needs to be,

The ratio of to gives us the downsampling factor, delN.

The highest IF frequency can be found using Equation [4] from “CCR SAR”:

fIFmax = 2\*BW\*Rmax/c/tau;

and

delN = floor(f0/fIFmax);

The downsampled IF return signal can be created as follows:

sigIF\_d = sigIF(1:delN:end);

and the number of elements of the IF return signal is given by

Nmax = length(sigIF\_d);

PULSE COMPRESSION

The next step is to compress the IF return signal into a target at the correct range using the Fast Fourier Transform function in MATLAB. To get a better impulse response shape, we need to zero pad our IF return signal so that it will have more elements. The pad factor is typically 8, meaning the FFT will produce an output with 8 times the elements of sigIF\_d.

pad\_factor = 8;

zeropad = Nmax \* pad\_factor;

SIF = fft(sigIF\_d,zeropad);

The variable SIF contains the complex, pulse compressed IF signal as shown in Figure 3 of “CCR SAR”. To verify the range processing worked correctly, the range of the pulse compressed target is calculated by finding the location (index) of the peak of the sinc function and then converting the index to range in meters.

[maxVal, maxLoc]=max(abs(SIF));

Rcalc = maxLoc\*Rmax/zeropad;

The last step is to create the vector, R\_axis, to be used for plotting later.

R\_axis = [0:zeropad-1]\*Rmax/zeropad;

The function returns the following variables, sigIF\_d, SIF, Rcalc, R\_axis, as described above.

CROSS RANGE PROCESSING

The next step is to deramp the linear frequency signal formed step to step across the synthetic aperture. This is equivalent to what the mixer did for the single pulse. The reference signal to “mix” with the range compressed signal is given by equation [5] in “CCR SAR”.

To accomplish this step in MATLAB, set up a definite loop where each pass through the loop corresponds to a step across the synthetic aperture.

kk = 1:numSteps

Calculate the “mixer” reference phase, which is a vector with the same length as kk:

XRg\_phase = (4\*pi/lambda)\*(R0+(-L/2+delL\*(kk-1)).^2/2/R0);

and the complex reference signal is given by:

refSig = exp(i\*XRg\_phase);

The deramped signal will be the same for each column of SIF and can be created as a matrix by multiplying refSig, transposed as as column vector, by a vector of ones whose length is the number of columns in SIF.

sar = refSig’.\*(SIF(:,kk));

CROSS RANGE PULSE COMPRESSION

Lastly, we need to pulse compress across the steps of the synthetic aperture for each column (range cell) of the signal. We do that using the Inverse Fourier Transform, with a zeropad factor equal to that for the range dimension.

zeropad = numSteps \* pad\_factor;

SARc(:,kk) = ifft(sar(:,kk),zeropad);

The final product of the range and cross range processing is the absolute value of SARc expressed in dB:

SAR = 10\*log10(abs(SARc));