Final Project Report
Radar Lessons
Sensor Fusion Nano degree
Udacity

Ayush Jain
Submission Date: 30/04/2020

FMCW Waveform Design

Q. Using the given system requirements, design a FMCW waveform. Find its Bandwidth (B), chirp time (Tchirp) and slope of the chirp.

Specification Needed: For given system requirements the calculated slope should be around 2e13

Ans:

```
target_range = 100;
target_velocity = 50;
radar_max_range = 200;
radar_range_resolution = 1;
radar_max_velocity = 100;
speed_of_light = 3e8;
%carrier frequency
fc= 77e9;
% It is frequency bandwidth.
B_sweep = speed_of_light /(2 * radar_range_resolution);
%t_roundtrip is the time needed for a wave to go and come back
t_roundtrip = 2 * (radar_max_range/speed_of_light);
t_chirp = 5.5 * t_roundtrip;
slope = B_sweep / t_chirp;
```

Simulation Loop

Q. Simulate Target movement and calculate the beat or mixed signal for every timestamp.

Specification Needed: A beat signal should be generated such that once range FFT implemented, it gives the correct range i.e the initial position of target assigned with an error margin of +/- 10 meters.

Ans:

```
% The number of doppler cells OR the number of sent periods OR the number of
    chirps
Nd=128;
%The number of samples on each chirp OR the length of time OR the number of
    range cells
Nr=1024;

% Timestamp for running the displacement scenario for every sample on each
    chirp
t=linspace(0,Nd*t_chirp,Nr*Nd);

%Creating the vectors for Tx, Rx and Mix based on the total samples input.
Tx=zeros(1,length(t)); %transmitted signal
Rx=zeros(1,length(t)); %received signal
Mix = zeros(1,length(t)); %beat signal

%Similar vectors for range_covered and time delay.
r_t=zeros(1,length(t));
td=zeros(1,length(t));
for i=1:length(t)
```

```
    %For each time stamp update the Range of the Target for constant velocity.
    % Target was initially at 100 mts. It decreases with time
    r_t(i) = target_range - (target_velocity*t(i));
    % Time needed for radar waves to travel and comeback to the source.
    td(i) = (2 * r_t(i)) / speed_of_light;

    %For each time sample we need to update the transmitted and received signal
    .
    Tx(i) = cos(2 * pi * (fc * t(i) + 0.5 * slope * t(i)^2));
    Rx(i) = cos(2 * pi * (fc * (t(i) - td(i))+0.5 * slope * (t(i) - td(i))^2));

    %Now by mixing the Transmitted and Received signal, we generate the beat
    signal
    %Element wise matrix multiplication of Transmitted and Received Signal
    Mix(i) = Tx(i).* Rx(i);
end
```

Range FFT (1st FFT)

Q. Implement the Range FFT on the Beat or Mixed Signal and plot the result.

Specification Needed: A correct implementation should generate a peak at the correct range, i.e the initial position of target assigned with an error margin of +/- 10 meters.

Ans:

```

% Reshapping Mix. New mix will have 1024 rows and 128 columns.
% A chirp is in a column. So there are 128 column. The length of column is 1024
Mix = reshape(Mix, [Nr, Nd]);

% FFT on the beat signal along the range bins dimension (Nr).
% FFT along coloumn is applied
sig_fft = fft(Mix, Nr);

% Absolute value of FFT output
sig_fft = abs(sig_fft);
% Normalised value of FFT output
sig_fft = sig_fft ./ max(sig_fft);

% One side of the spectrum of FFT.
sig_fft = sig_fft(1 : Nr/2-1);

% Plotting the range
figure ('Name','Range from First FFT')
plot(sig_fft);
axis ([0 200 0 1]);
title('Range from First FFT');
ylabel('Normalized Amplitude');
xlabel('Range');
```

2D CFAR

Q. Implement the 2D CFAR process on the output of 2D FFT operation, i.e the Range Doppler Map.

Specification Needed: The 2D CFAR processing should be able to suppress the noise and separate the target signal. The output should match the image shared in walkthrough.

Ans:

```
%The number of Training Cells in both the dimensions.
train_cells = 10;
train_band = 8;

%The number of Guard Cells in both dimensions around the Cell Under Test (CUT) for accurate estimation
guard_cells = 4;
guard_band = 4;

% Offset the threshold by Signal to Noise (SNR) ration value in dB
offset = 1.4;

% Using RDM[x,y] as the matrix from the output of 2D FFT for implementing CFAR
% Normalising RDM matrix
RDM = RDM / max(RDM(:));

for row1 = train_cells + guard_cells + 1 : (Nr/2) - (train_cells + guard_cells)
  for col1 = train_band + guard_band + 1 : (Nd) - (train_band + guard_band)

    %Create a vector to store noise_level for each iteration on training cells
    noise_level = zeros(1, 1);

    for row2 = row1 - (train_cells + guard_cells) : row1 + (train_cells + guard_cells)
      for col2 = col1 - (train_band + guard_band) : col1 + (train_band + guard_band)
        if (abs(row1 - row2) > guard_cells || abs(col1 - col2) > guard_band)
          noise_level = noise_level + db2pow(RDM(row2, col2));
        end
      end
    end

    % Calculate threshold from noise average then add the offset
    threshold = pow2db(noise_level / (2 * (train_band + guard_band + 1) * 2 * (train_cells + guard_cells + 1) - (guard_cells * guard_band) - 1));
    threshold = threshold + offset;

    cell_under_test = RDM(row1,col1);

    if (cell_under_test < threshold)
      RDM(row1, col1) = 0;
    else
      RDM(row1, col1) = 1;
    end
  end
end
% RDM is Range Doppler Map.
RDM(RDM~=0 & RDM~=1) = 0;

figure('Name', 'CA-CFAR Filtered RDM')
surf(doppler_axis,range_axis,RDM);
colorbar;
title( 'CA-CFAR Filtered RDM surface plot');
xlabel('Speed');
ylabel('Range');
zlabel('Normalized Amplitude');
view(315, 45);
```

Create a CFAR README File

Q. In a README file, write brief explanations for the following:
Specification Needed:

1. Implementation steps for the 2D CFAR process.
2. Selection of Training, Guard cells and offset.
3. Steps taken to suppress the non-thresholded cells at the edges.

Ans: Please refer the code above.

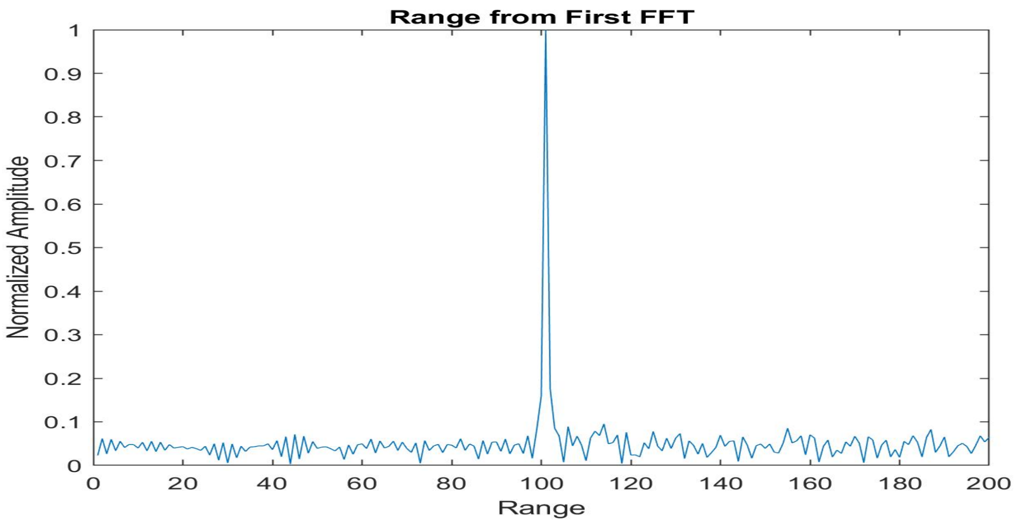

Figure 1:

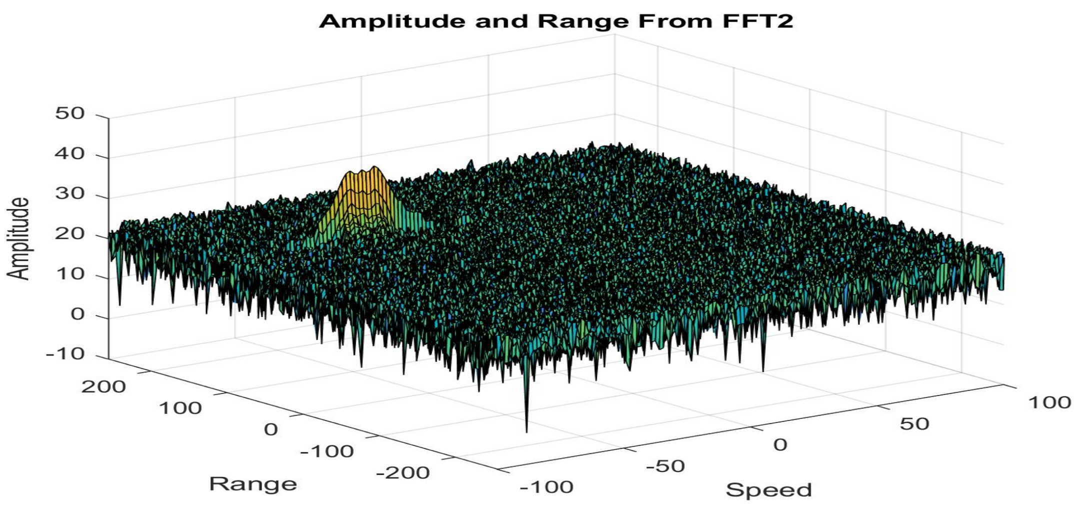

Figure 2: Tracking an Object in 3D Space

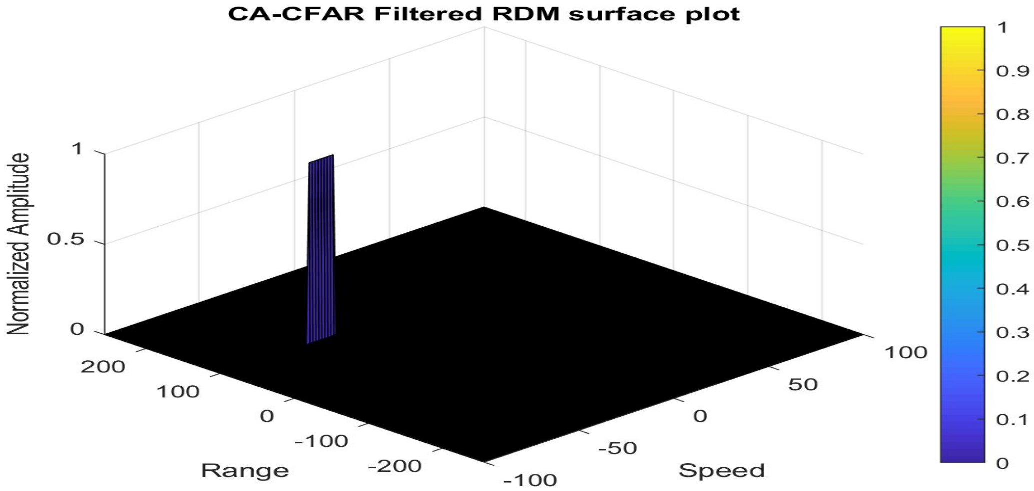

Figure 3: Tracking an Object in 3D Space