Diagnostics - USRP Attacking Subsystem

Initialize the configuration to test against

```
clear;
clf;
simulator = Simulator_revB();
kristen_path = "C:\Users\krist\OneDrive\Documents\2022-2023 School Year\Radar Security
windows_file_path = "C:\Users\Operator\Documents\RadarSecurityResearch\MATLAB\Simulink
david_path = "/home/david/Documents/BlackBoxRadarAttacks/MATLAB/config_files/";
% file_path = "20MHz_USRP.json";
% file_path = "20MHz_USRP_revA.json";
file_path = "20MHz_USRP_revB.json";
% file_path = "20MHz.json";
% file_path = "100MHz.json";
% file_path = "100MHz_USRP.json";
% file_path = "1GHz.json";
% file_path = "4GHz.json";
simulator.load_params_from_JSON(david_path + file_path);

%apply timing offsets as desired
simulator.Victim.timing_offset_us = 0;
simulator.Attacker.Subsystem_tracking.timing_offset_us = 0;

%configure the FMCW parameters
simulator.configure_FMCW_Radar_parameters();

%load default attacker, and victim positions and velocities
simulator.load_usrp_attacker_and_victim_position_and_velocity();

%print out key parameters

simulator.Victim.print_radar_parameters;
```

```
Chirp Parameters
     Start Frequency:          3.00 GHz
     Frequency Slope:          0.10 MHz/us
     Idle Time:               11.28 us
     Tx Start Time:           0.00 us
     ADC Valid Start Time:          7.52 us
     ADC Samples:              64
     ADC Sample Rate:          0.27 MSps
     Ramp End Time:           248.12 us
     Chirp Tx Bandwidth:           24.99 MHz
     Chirp Sampling Bandwidth:      24.23 MHz
     ADC Sampling Period:          240.60 us
     Chirp Cycle Time:          259.40 us
     Chirp Wavelength:          99.93 mm
Frame Parameters
     Number of Chirps          256
     FramePeriodicity          100.13 ms
     Active Frame Time          66.41 ms
Performance Specifications
     Max Range              395.95 m
```

```
     Range Resolution          6.19 m
     Max Velocity              96.31 m/s
     Velocity Resolution          0.75 m/s
FMCW Specifications
     FMCW sampling rate          25.00 MHz
     Downsampling factor          94
     Sweep time              248.12 us
     Samples per chirp          6486.00
CFAR Detection Region
     Range Detection Region          61.868 m to 327.898 m
     Velocity Detection Region          -75.242 m/s to 75.995 m/s
```

Precompute victim chirps

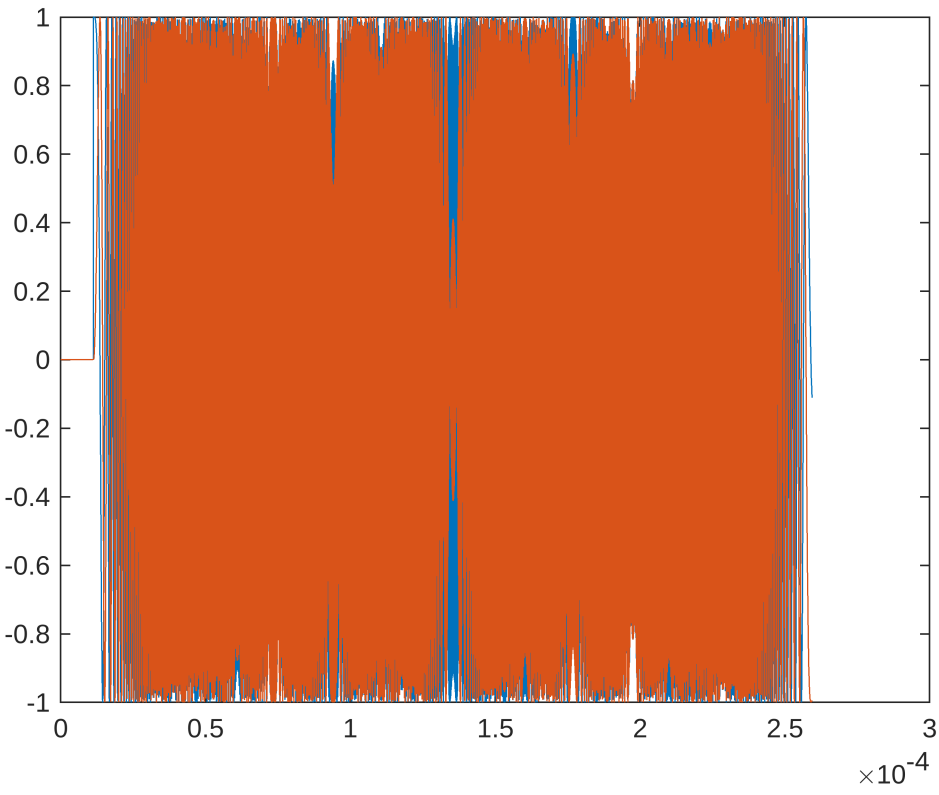
```
%pre-compute the victim's chirps
simulator.Victim.precompute_radar_chirps();
```

Save the victim chirp to a file

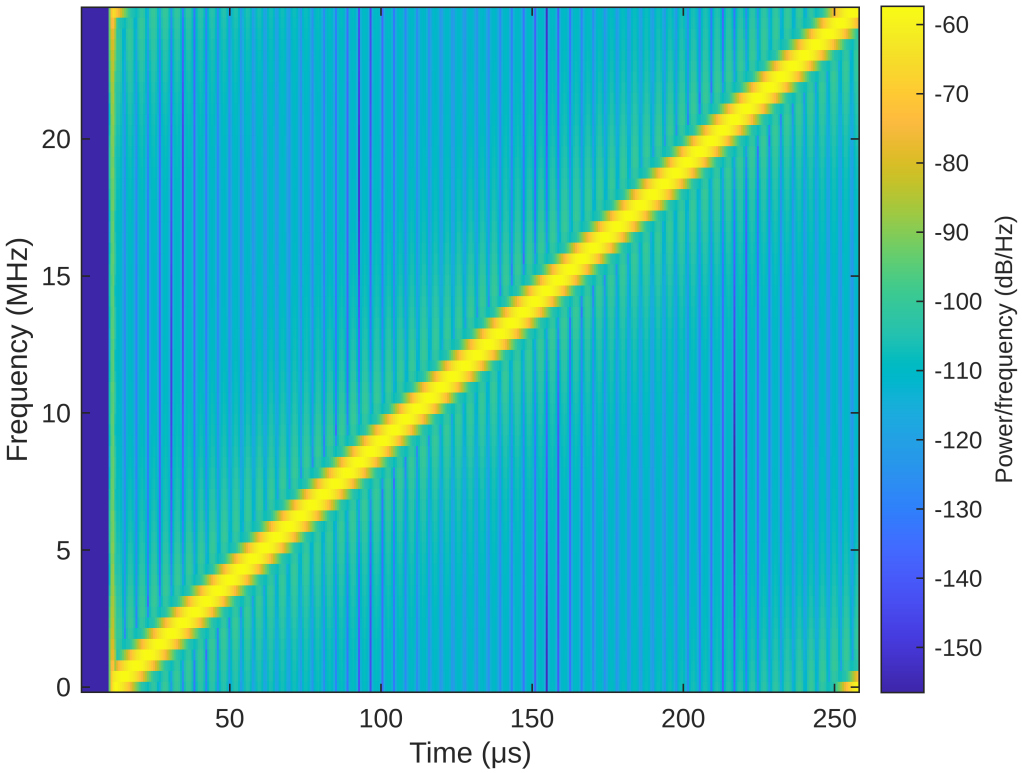
```
%save the full chirp as a binary file
path = "/home/david/Documents/MATLAB_generated/MATLAB_full_chirps/MATLAB_chirp_full.bin
simulator.save_to_file(simulator.Victim.chirp,path,'float32');
```

Plot Tx Chirp to confirm correctness

```
%plot the chirp signal
t = 0:simulator.Victim.FMCW_sampling_period_s:...
    simulator.Victim.ChirpCycleTime_us * 1e-6 - ...
    simulator.Victim.FMCW_sampling_period_s;
data = simulator.Victim.chirp;
plot(t,real(data),t,imag(data))
```




```
spectrogram(data,64,48,64,simulator.Victim.FMCW_sampling_rate_Hz,'yaxis')
```



RUN ATTACK ON MATLAB

Read Rx data from MATLAB

```
path = "/home/david/Documents/MATLAB_generated/cpp_rx_data_files/cpp_rx_data.bin";
read_data = simulator.read_from_file(path,true,"float");

num_chirps = simulator.Victim.NumChirps;
samples_per_chirp = simulator.Victim.ChirpCycleTime_us * 1e-6 * simulator.Victim.FMCW_s
read_data = reshape(read_data,int32(samples_per_chirp),num_chirps,[]);

%determine the number of frames recorded
num_frames = size(read_data,3);
```

Configure the movie to focus in on specific arease of the range-doppler and CFAR plots

```
%specify whether or not to record a move of the range-doppler plot
record_movie = true;
target_pos = 50;
target_vel = 0;
simulator.Victim.Radar_Signal_Processor.configure_movie_capture(num_frames, ...
    record_movie,target_pos,-1 * target_vel,80);
```

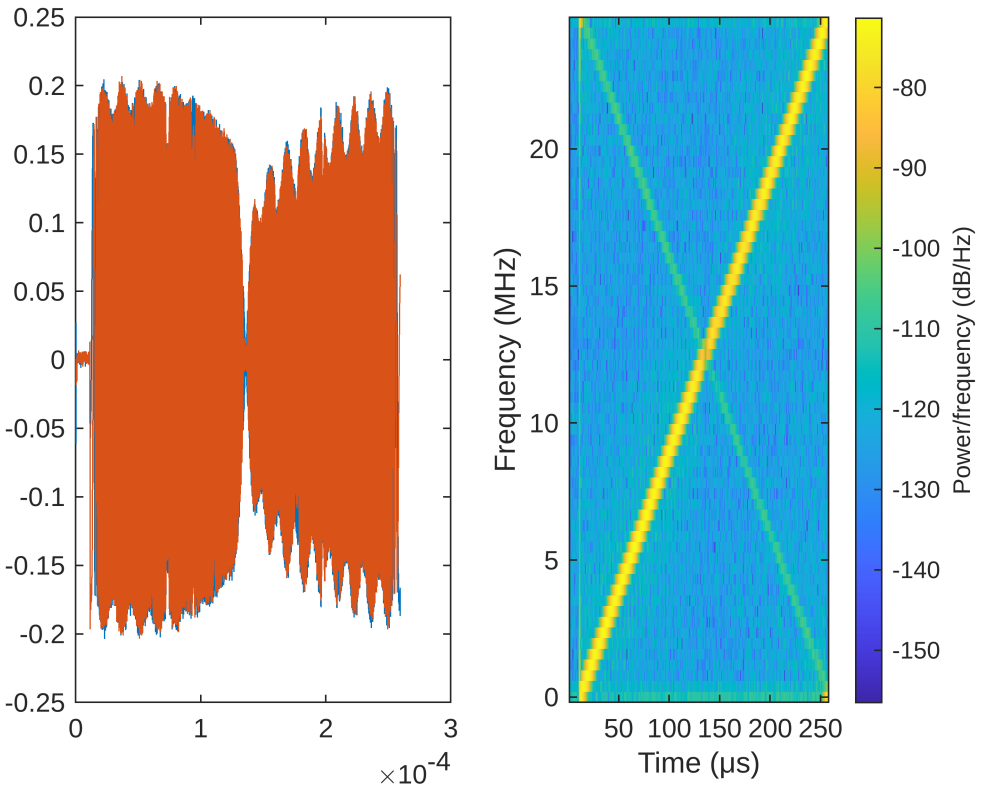
Plot Rx Chirp to confirm correctness

Specify chirp:

```
chirp =47;
```

Specify frame:

```
frame = 10;
first_frame = read_data(:,:,1);
t = 0:simulator.Victim.FMCW_sampling_period_s:...
    simulator.Victim.ChirpCycleTime_us * 1e-6 - ...
    simulator.Victim.FMCW_sampling_period_s;
clf;
subplot(1,2,1);
plot(t,real(read_data(:,chirp,frame)),t,imag(read_data(:,chirp,frame)))
subplot(1,2,2)
spectrogram(read_data(:,chirp,frame),64,48,64,simulator.Victim.FMCW_sampling_rate_Hz,'y
```



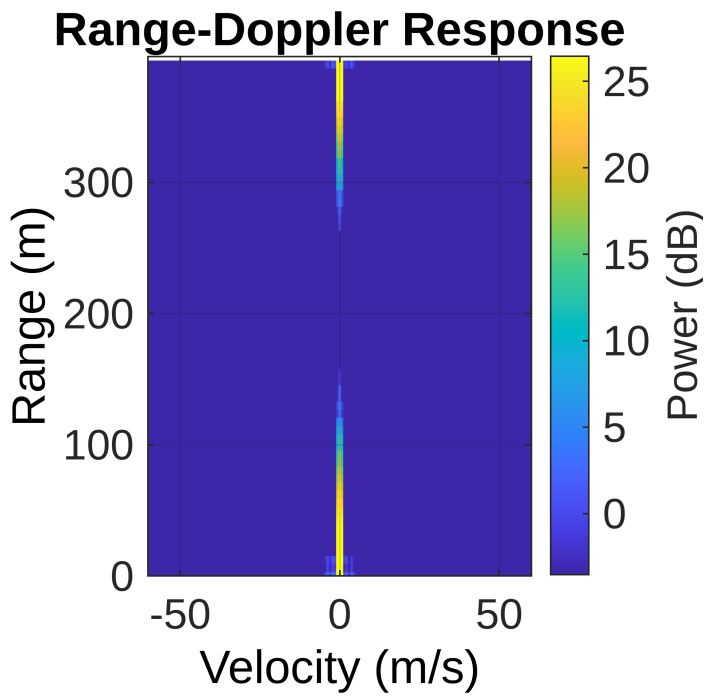
Initialize the Victim and Simulation Parameters

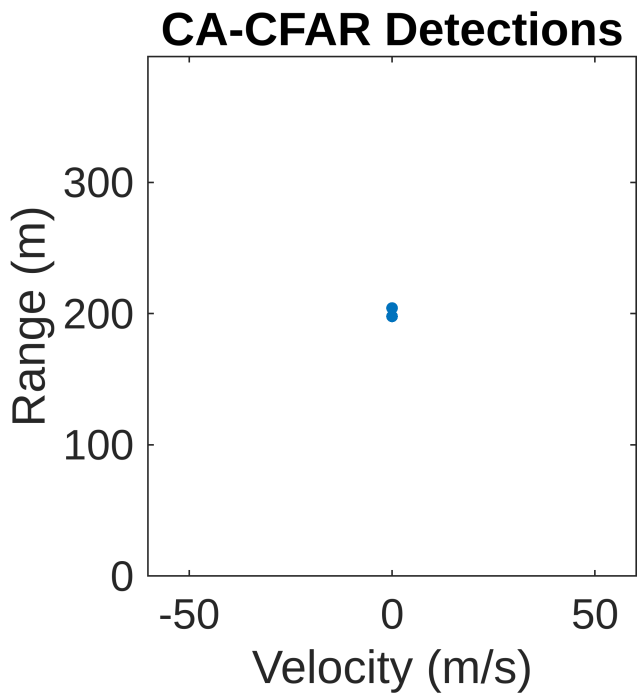
```
%specify the number of frames and chirps to compute
clf;
% num_frames = size(read_data,3);
% num_chirps = simulator.Victim.NumChirps;
% num_frames = 30;

%reset the movie and range estimates
% simulator.Victim.Radar_Signal_Processor.F_rngdop = struct('cdata',[],'colormap',[]);
% simulator.Victim.Radar_Signal_Processor.F_clusters = struct('cdata',[],'colormap',[])
% simulator.Victim.Radar_Signal_Processor.range_estimates = [];
% simulator.Victim.Radar_Signal_Processor.velocity_estimates = [];

%specify whether or not to record a move of the range-doppler plot
% record_movie = true;
% simulator.Victim.Radar_Signal_Processor.configure_movie_capture(num_frames, ...
%     record_movie,target_pos,-1 * target_vel,80);

%pre-compute the victim's chirps
% simulator.Victim.precompute_radar_chirps();
```

Process Received Rx Signal

```
%get the radar chirp generator and the target emulator ready
```

```
clf;
simulator.Victim.precompute_radar_chirps();

for frame = int32(1:num_frames)

    %initialize a clean radar cube
    simulator.Victim.Radar_Signal_Processor.reset_radar_cube();
    simulator.Victim.current_frame = frame;

    for chirp = int32(1:num_chirps)

        %get the Tx chirp and the chirp received from the emulator
        Rx_sig = read_data(:,chirp,frame);
        Tx_sig = simulator.Victim.chirps(:,chirp);

        %assemble the radar cube
        simulator.Victim.Radar_Signal_Processor.update_radar_cube(Tx_sig,Rx_sig,chirp);
    end

    simulator.Victim.Radar_Signal_Processor.process_radar_cube();
end
```





Results

```
%get the estimated ranges and velocities
simulator.Victim.Radar_Signal_Processor.range_estimates
```

```
ans = 60×5
  278.5169  199.9009  115.4079       NaN       NaN
  200.0246  277.9224  115.8326       NaN       NaN
  200.3151  277.9306  116.3010       NaN       NaN
  199.7237  278.1225  115.5832       NaN       NaN
  278.0193  199.8065  115.9354       NaN       NaN
  199.5692  278.3801  115.4876       NaN       NaN
  199.3280  278.1028  115.5975       NaN       NaN
  199.5030  278.0048  115.5638       NaN       NaN
  199.6281  278.0703  117.2204       NaN       NaN
  277.9340  116.8232  115.0570       NaN       NaN
     ⋮
```

```
simulator.Victim.Radar_Signal_Processor.velocity_estimates
```

```
ans = 60×5
   75.3809   -0.0122  -75.3893       NaN       NaN
   -0.0091   75.4086  -75.3616       NaN       NaN
   -0.0076   75.3814  -75.3777       NaN       NaN
   -0.0099   75.3722  -75.3825       NaN       NaN
   75.3752    0.0019  -75.3921       NaN       NaN
   -0.0122   75.3840  -75.3804       NaN       NaN
    0.0030   75.3556  -75.3942       NaN       NaN
   -0.0333   75.4001  -75.3709       NaN       NaN
    0.0066   75.3862  -75.3689       NaN       NaN
   75.3919  -19.0774  -75.3408       NaN       NaN
     ⋮
```

```
%play the movie for the range doppler
if record_movie
    simulator.Victim.Radar_Signal_Processor.play_range_doppler_movie();
end
```

```
%play the movie for the clusters
if record_movie
    simulator.Victim.Radar_Signal_Processor.play_clustering_movie()
end
```

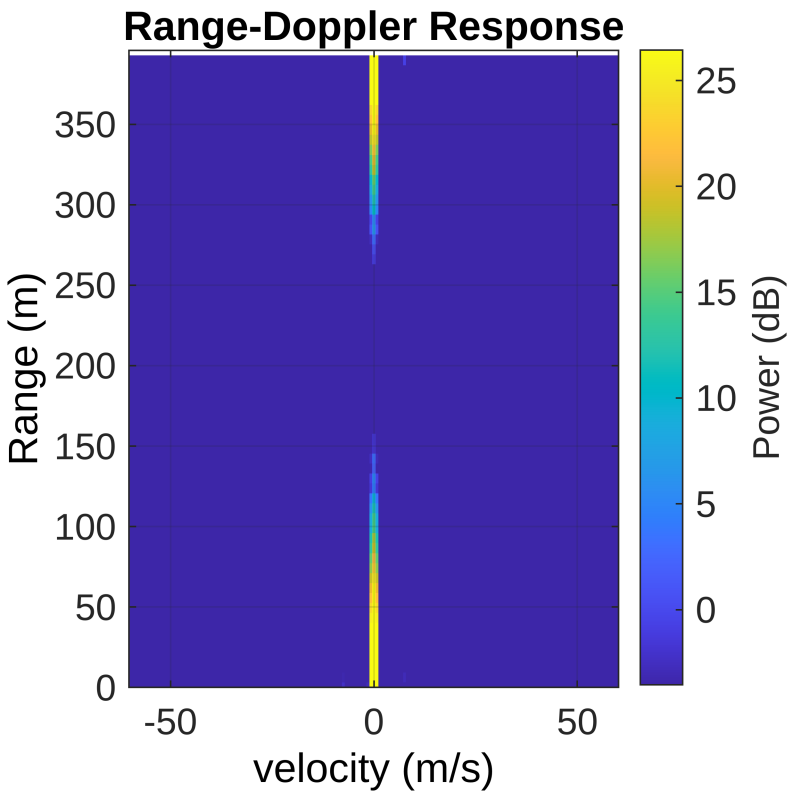
Generating Clustering and Range-Doppler Response Figures

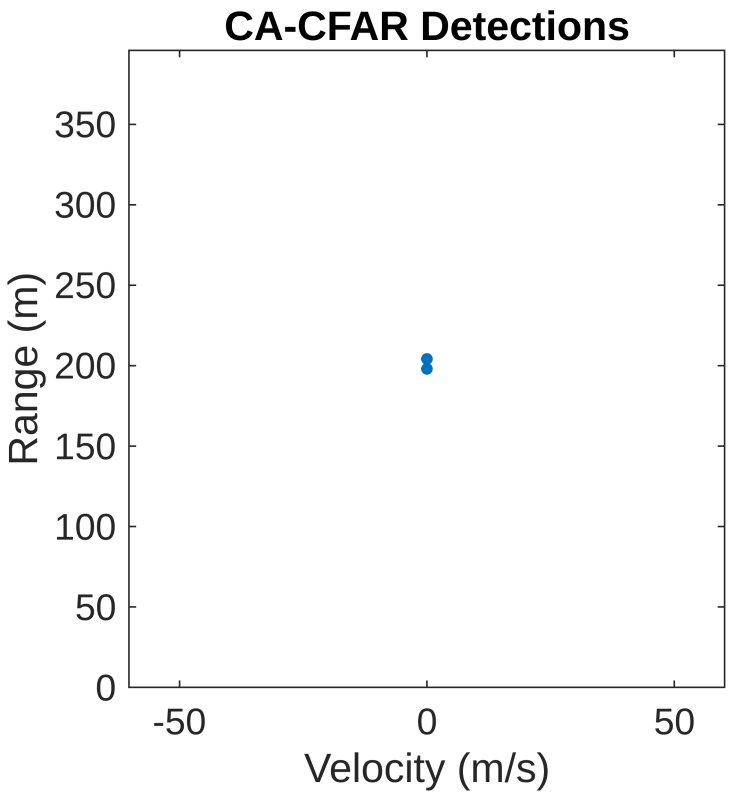
```
%range-doppler response
clf;
[resp, rnggrid,dopgrid] = simulator.Victim.Radar_Signal_Processor.RangeDopplerResponse(
    simulator.Victim.Radar_Signal_Processor.radar_cube);
resp_max = 10*log10(abs(max(resp,[],"all")));

%CA CFAR 2-D
detections = simulator.Victim.Radar_Signal_Processor.CFARDetector2D(...
    abs(resp).^2,simulator.Victim.Radar_Signal_Processor.CUT_indicies);
detected_velocities = dopgrid(detections(2,:));
detected_ranges = rnggrid(detections(1,:));



%estimate the range and the velocities
if ~isempty(detections)

    %DBSCAN Clustering
    idx = dbscan(detections.',...
        simulator.Victim.Radar_Signal_Processor.Epsilon,...
        simulator.Victim.Radar_Signal_Processor.minpts);

    %plot the range-doppler and clustering responses
```

```
%plot range doppler
plotResponse( ...
    simulator.Victim.Radar_Signal_Processor.RangeDopplerResponse, ...
    simulator.Victim.Radar_Signal_Processor.radar_cube);

%zoom the range doppler plot
tgt_velocity = simulator.Victim.Radar_Signal_Processor.tgt_velocity;
tgt_range = simulator.Victim.Radar_Signal_Processor.tgt_range;
num_bins_zoom = simulator.Victim.Radar_Signal_Processor.num_bins_zoom;

clim([resp_max-30, resp_max]);
xlim([max(tgt_velocity - num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
    -1 * simulator.Victim.V_Max_m_per_s), ...
    min(tgt_velocity + num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
    simulator.Victim.V_Max_m_per_s)])
ylim([max(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeric
    - num_bins_zoom * simulator.Victim.Range_Res_m,...
    0), ...
    min(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodi
    * tgt_velocity) ...
    + num_bins_zoom * simulator.Victim.Range_Res_m,...
    simulator.Victim.Range_Max_m)])

font_size = 14;
h = colorbar;
h.FontSize = font_size;
h.Label.String = "Power (dB)";
h_label = h.Label;
set(gcf,'Position',[100 100 400 400])
title("Range-Doppler Response","FontSize",font_size)
xlabel("velocity (m/s)","FontSize",font_size)
ylabel("Range (m)","FontSize",font_size)
ax = gca;
ax.FontSize = font_size;
print('-r300',"generated_plots/range_doppler",'-dsvg')
print('-r300',"generated_plots/range_doppler",'-dpng')

%plot the clusters
gscatter(detected_velocities,detected_ranges,idx);
xlim([max(tgt_velocity - num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
    -1 * simulator.Victim.V_Max_m_per_s), ...
    min(tgt_velocity + num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
    simulator.Victim.V_Max_m_per_s)])
ylim([max(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeric
    - num_bins_zoom * simulator.Victim.Range_Res_m,...
    0), ...
    min(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodi
    * tgt_velocity) ...
    + num_bins_zoom * simulator.Victim.Range_Res_m,...
```

```
        simulator.Victim.Range_Max_m)])

    font_size = 14;
%     colorbar(gca,"FontSize",font_size)
    legend('off')
    set(gcf,'Position',[100 100 400 400])
    title("CA-CFAR Detections","FontSize",font_size)
    xlabel("Velocity (m/s)","FontSize",font_size)
    ylabel("Range (m)","FontSize",font_size)
    ax = gca;
    ax.FontSize = font_size;
    print('-r300',"generated_plots/clustering",'-dsvg')
    print('-r300',"generated_plots/clustering",'-dpng')
else
    %plot range doppler
    plotResponse( ...
        simulator.Victim.Radar_Signal_Processor.RangeDopplerResponse, ...
        simulator.Victim.Radar_Signal_Processor.radar_cube);

    %zoom the range doppler plot
    tgt_velocity = simulator.Victim.Radar_Signal_Processor.tgt_velocity;
    tgt_range = simulator.Victim.Radar_Signal_Processor.tgt_range;
    num_bins_zoom = simulator.Victim.Radar_Signal_Processor.num_bins_zoom;

    clim([resp_max-30, resp_max]);
    xlim([max(tgt_velocity - num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        -1 * simulator.Victim.V_Max_m_per_s), ...
        min(tgt_velocity + num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        simulator.Victim.V_Max_m_per_s)])
    ylim([max(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeric
        - num_bins_zoom * simulator.Victim.Range_Res_m,...
        0), ...
        min(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodi
        * tgt_velocity) ...
        + num_bins_zoom * simulator.Victim.Range_Res_m,...
        simulator.Victim.Range_Max_m)])

    font_size = 14;
    h = colorbar;
    h.FontSize = font_size;
    h.Label.String = "Power (dB)";
    h_label = h.Label;
    set(gcf,'Position',[100 100 400 400])
    title("Range-Doppler Response","FontSize",font_size)
    xlabel("Velocity (m/s)","FontSize",font_size)
    ylabel("Range (m)","FontSize",font_size)
    ax = gca;
    ax.FontSize = font_size;
    print('-r300',"generated_plots/range_doppler",'-dsvg')
    print('-r300',"generated_plots/range_doppler",'-dpng')
```

```
    %plot the clusters
    clf;
    detected_velocities = [0];
    detected_ranges = [0];
    idx = [1];
    gscatter(detected_velocities,detected_ranges,idx);
    xlim([max(tgt_velocity - num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        -1 * simulator.Victim.V_Max_m_per_s), ...
        min(tgt_velocity + num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        simulator.Victim.V_Max_m_per_s)])
    ylim([max(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeric
        - num_bins_zoom * simulator.Victim.Range_Res_m,...
        0), ...
        min(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodi
        * tgt_velocity) ...
        + num_bins_zoom * simulator.Victim.Range_Res_m,...
        simulator.Victim.Range_Max_m)])
    cla;
    font_size = 14;
%     colorbar(gca,"FontSize",font_size)
    legend('off')
    set(gcf,'Position',[100 100 400 400])
    title("CA-CFAR Detections","FontSize",font_size)
    xlabel("Velocity (m/s)","FontSize",font_size)
    ylabel("Range (m)","FontSize",font_size)
    ax = gca;
    ax.FontSize = font_size;
    print('-r300',"generated_plots/clustering",'-dsvg')
    print('-r300',"generated_plots/clustering",'-dpng')
end
```





Generate a subplot sampling of multiple frames for a preview of the movie

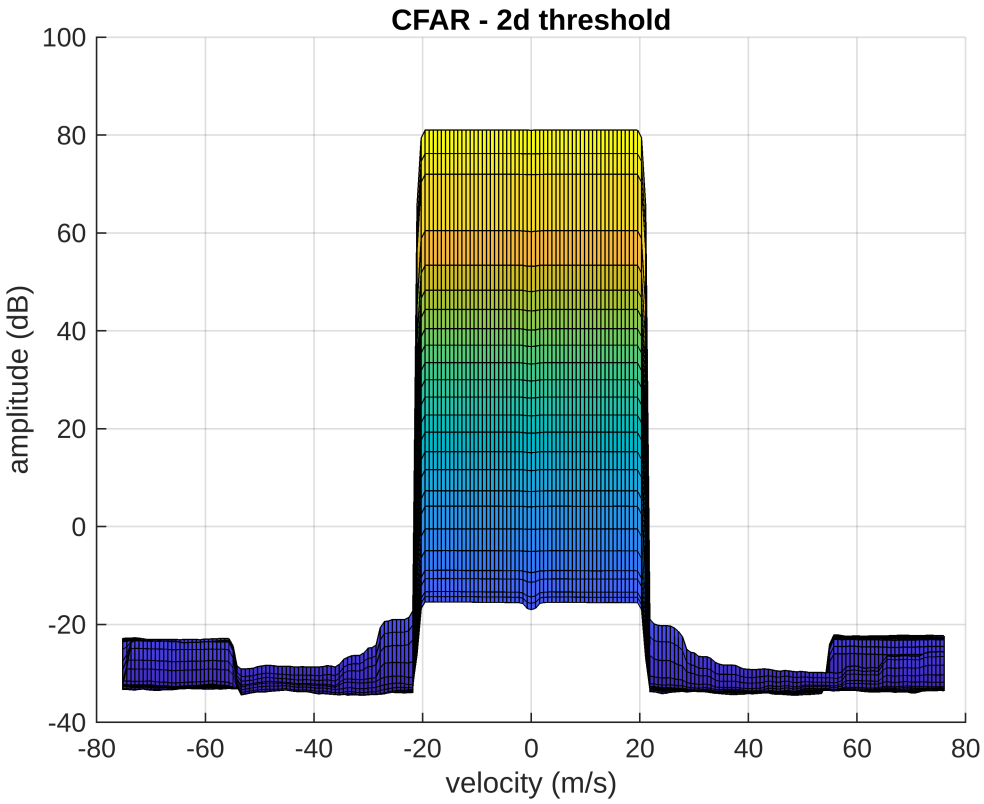
```
% clf;
% figure('Position',[1,1,900,800]);
% axis fill;
```

```
% num_frames = 30
% tiledlayout(2,2,TileSpacing="none",Padding="tight")
%
%     %subplot of the Doppler in the left column
%     nexttile
%     [X,map] = frame2im(simulator.Victim.Radar_Signal_Processor.F_rngdop(ceil(1)));
%     imshow(X,map)
%     nexttile
%     [X,map] = frame2im(simulator.Victim.Radar_Signal_Processor.F_rngdop(ceil(num_fram
%     imshow(X,map)
%
%     %subplot of the clustering in the right column
%     nexttile
%     [X,map] = frame2im(simulator.Victim.Radar_Signal_Processor.F_clusters(ceil(1)));
%     imshow(X,map)
%     nexttile
%     [X,map] = frame2im(simulator.Victim.Radar_Signal_Processor.F_clusters(ceil(num_fr
%     imshow(X,map)
%
% print('-r500',"generated_plots/attack_timeline",'-dsvg')
% print('-r500',"generated_plots/attack_timeline",'-dpng')
```

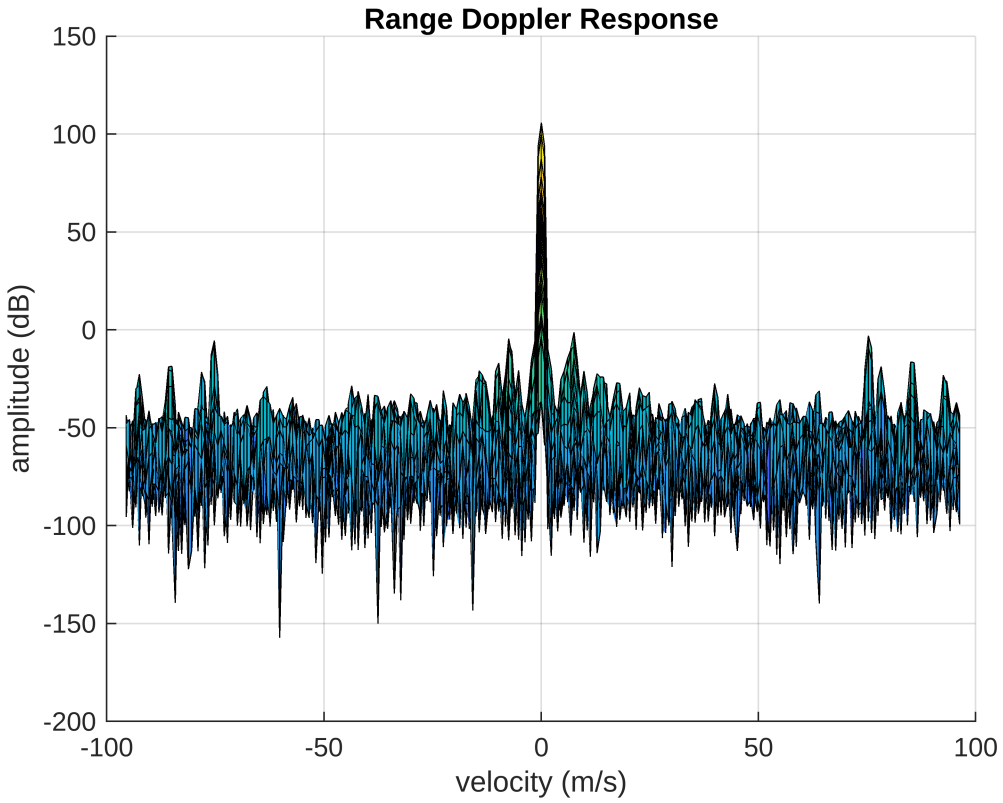
Evaluating CFAR Performance with surf plots

```
clf;
figure;

%evaluate the 2d CFAR detections for the desired frame
radar_cube = simulator.Victim.Radar_Signal_Processor.radar_cube;
[frame_resp, rnggrid,dopgrid] = simulator.Victim.Radar_Signal_Processor.RangeDopplerRes
CUT_indicies = simulator.Victim.Radar_Signal_Processor.CUT_indicies;

%configure the CFAR
release(simulator.Victim.Radar_Signal_Processor.CFARDetector2D);
simulator.Victim.Radar_Signal_Processor.CFARDetector2D.ThresholdOutputPort = true;
simulator.Victim.Radar_Signal_Processor.CFARDetector2D.OutputFormat ="CUT result";


%compute the CFAR
[detections,th] = simulator.Victim.Radar_Signal_Processor.CFARDetector2D(abs(frame_resp

%determine range indicies in threshold detector
ranges = rnggrid(CUT_indicies(1,:)); %y
velocities = dopgrid(CUT_indicies(2,:)); %x


dopgrid_th = linspace(min(velocities), max(velocities),max(CUT_indicies(2,:)) - min(CUT
rnggrid_th = linspace(min(ranges), max(ranges),max(CUT_indicies(1,:)) - min(CUT_indicie
[X,Y] = meshgrid(dopgrid_th, rnggrid_th);
```

```
%convert the thresholds to db
th = 20 * log10(th);

th_surface = griddata(velocities,ranges,th,X,Y);
surf(X, Y, th_surface);
view(0,0);
title("CFAR - 2d threshold");
xlabel("velocity (m/s)");
ylabel("range (m)");
zlabel("amplitude (dB)");
```



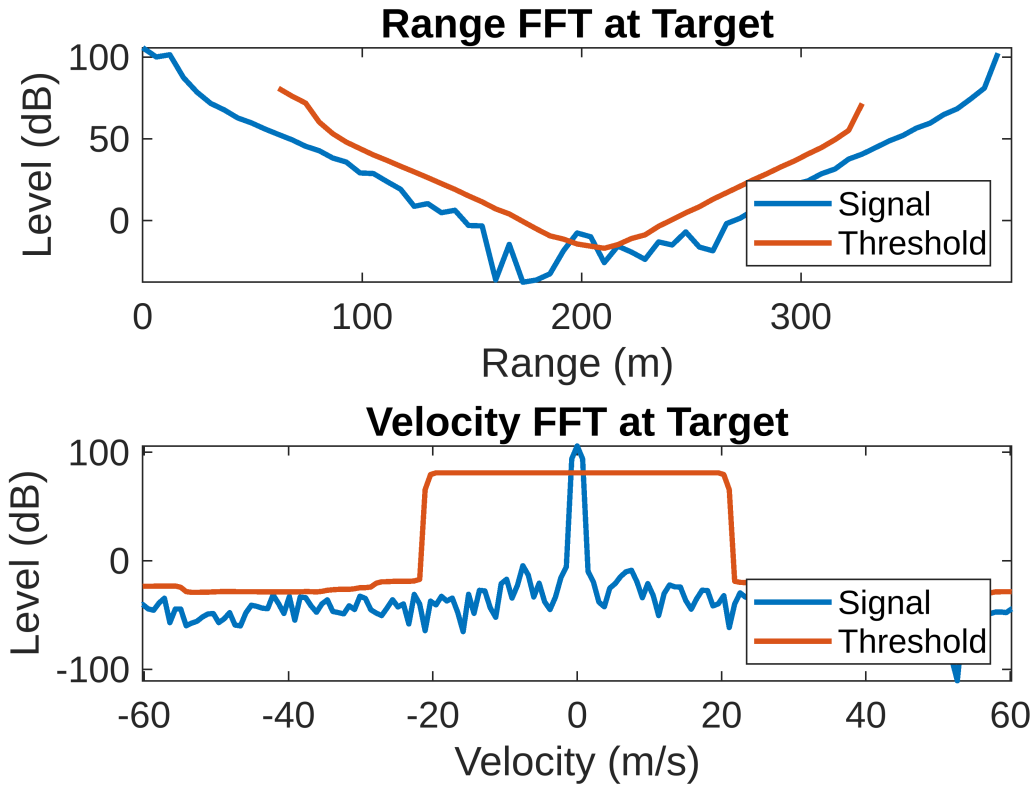

```
%convert the response to db
frame_resp = 20 * log10(abs(frame_resp).^2);
surf(dopgrid,rnggrid,frame_resp)
view(0,0);
title("Range Doppler Response");
xlabel("velocity (m/s)");
ylabel("range (m)");
zlabel("amplitude (dB)");
```



Evaluating Range Performance

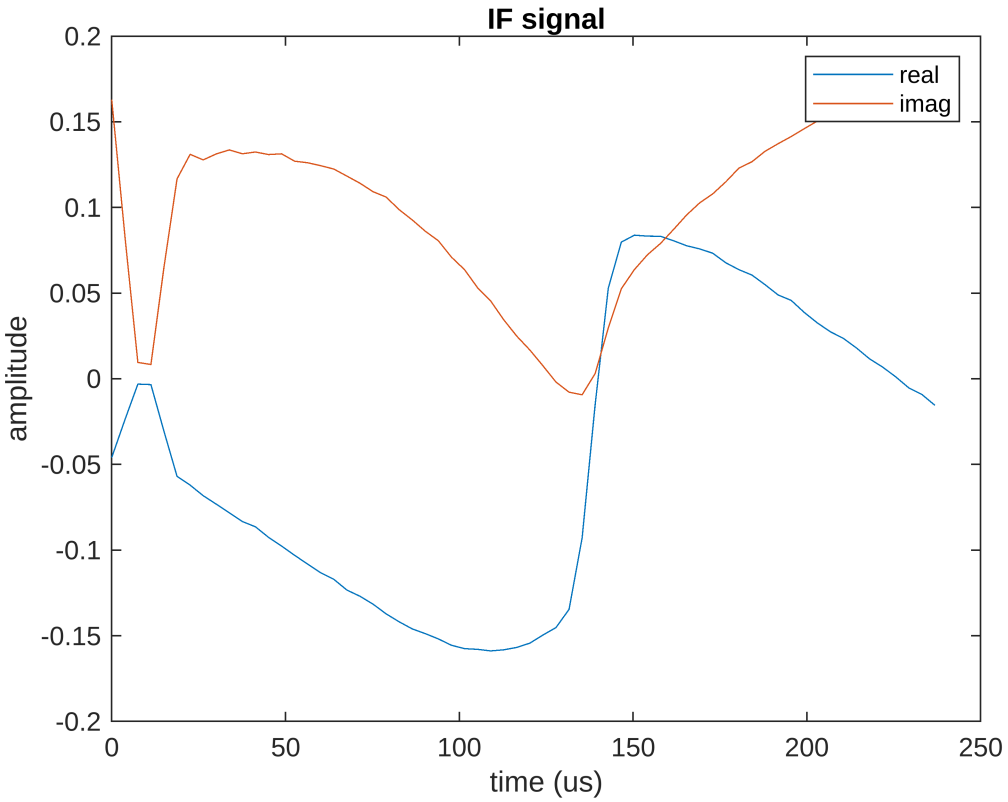
```
%find the maximum point
[M,I] = max(frame_resp,[],'all');
[rng_idx,vel_idx] = find(frame_resp == M);

%compute the indicies for the computed threshold
rng_idx_th = find(abs(rnggrid_th - rnggrid(rng_idx)) < 1e-4);
if isempty(rng_idx_th)
    rng_idx_th = 1;
end
vel_idx_th = find(abs(dopgrid_th - dopgrid(vel_idx)) < 1e-4);
if isempty(vel_idx_th)
    vel_idx_th = 1;
end

%plot range
rng_fft = frame_resp(:,vel_idx);
rng_th = th_surface(:,vel_idx_th);

subplot(2,1,1)
plot(rnggrid,rng_fft,rnggrid_th,rng_th,"LineWidth",2.0)

xlim([max(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodici
    - num_bins_zoom * simulator.Victim.Range_Res_m,...
```

```
    0), ...
    min(tgt_range - (simulator.Victim.current_frame * simulator.Victim.FramePeriodicity
    * tgt_velocity) ...
    + num_bins_zoom * simulator.Victim.Range_Res_m,...
    simulator.Victim.Range_Max_m)])

font_size = 14;
title("Range FFT at Target","FontSize",font_size)
xlabel("Range (m)","FontSize",font_size)
ylabel("Level (dB)","FontSize",font_size)
ax = gca;
ax.FontSize = font_size;
legend('Signal','Threshold','Location','Southeast')

%plot velocity
dop_fft = frame_resp(rng_idx,:);
dop_th = th_surface(rng_idx_th,:);

subplot(2,1,2)
plot(dopgrid,dop_fft,dopgrid_th,dop_th, "lineWidth",2.0)
legend('Signal','Threshold','Location','Southeast')
xlabel('Velocity')
ylabel('Level')
title("Velocity FFT at target")

xlim([max(tgt_velocity - num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        -1 * simulator.Victim.V_Max_m_per_s), ...
        min(tgt_velocity + num_bins_zoom * simulator.Victim.V_Res_m_per_s, ...
        simulator.Victim.V_Max_m_per_s)])

font_size = 14;
title("Velocity FFT at Target","FontSize",font_size)
xlabel("Velocity (m/s)","FontSize",font_size)
ylabel("Level (dB)","FontSize",font_size)
ax = gca;
ax.FontSize = font_size;

%save the plots
% print('-r300',"generated_plots/CFAR_output",'-dsvg')
print('-r300',"generated_plots/CFAR_output",'-dpng')
```



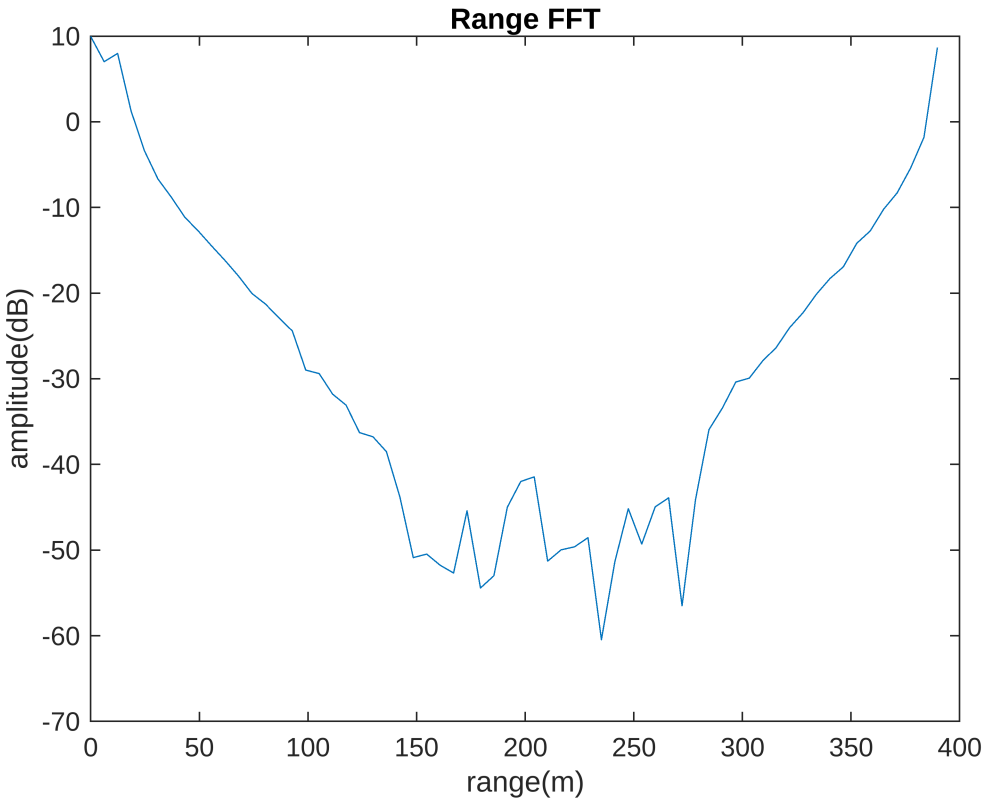
Generate plot of IF signals

```
sampled_IF_sig = simulator.Victim.Radar_Signal_Processor.radar_cube(:,1);
inter_sample_period = 1/simulator.Victim.ADC_SampleRate_MSps;
times = 0:inter_sample_period:(simulator.Victim.ADC_Samples - 1) * inter_sample_period;
clf;
plot(times,real(sampled_IF_sig),times, imag(sampled_IF_sig));
legend("real","imag")
xlabel("time (us)")
ylabel("amplitude")
title("IF signal")
print('-r300',"generated_plots/if_signal",'-dsvg')
print('-r300',"generated_plots/if_signal",'-dpng')
```



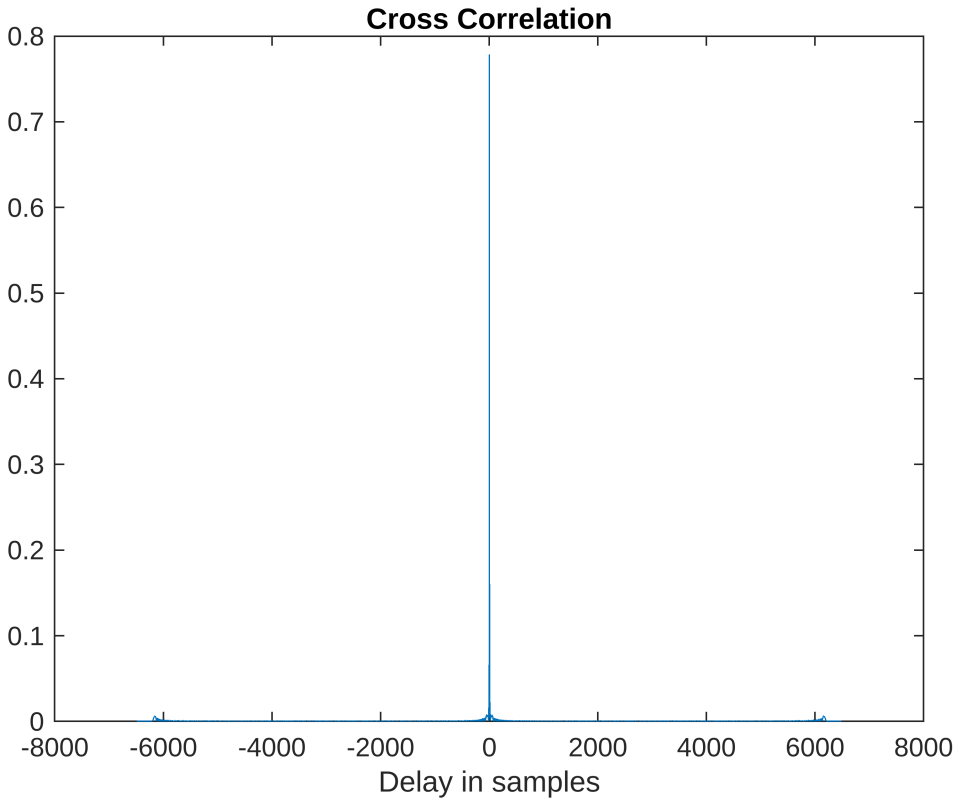
Plot Range FFT

```
window = hann(simulator.Victim.ADC_Samples);
windowed_sig = window .* radar_cube(:,1);
range_fft = 20 * log10(abs(fft(windowed_sig)));
clf;
plot(simulator.Victim.Ranges,range_fft)
title("range FFT at target")
xlabel("range(m)")
ylabel("amplitude(dB)")
title("Range FFT")
print('-r300',"generated_plots/Range_fft",'-dsvg')
print('-r300',"generated_plots/Range_fft",'-dpng')
```



Code to align the rx and the tx signal using cross correlation

```
[C,lag] = xcorr(Rx_sig,Tx_sig,"normalized");
clf;
plot(lag,abs(C));
xlabel("Delay in samples")
title("Cross Correlation")
```




```
[M,I] = max(abs(C));
delay_samps = lag(I)
```

```
delay_samps = 0
```

```
delay_us = delay_samps * simulator.Victim.FMCW_sampling_period_s * 1e6
```

```
delay_us = 0
```

```
%positive lag is ahead of time (decrease the offset_us term)
```