Cairo University - Faculty Of Engineering

EECE Department – Fall 2025

Analysis of Continuous and Discrete -Time Signals - EECG231 Project

Target Detection and Velocity Estimation using FMCW Radar

Submitted to:

Dr. Michael Melek Abdel-Sayed

Dr. Mohammed Abdelghany

Eng. Sayed Kamel

Name	Code
Ahmed Hassan Abdelhalim Labib	91240075
Hasan Salah Abdelnaby Eisa	91241306

Table of Contents

Introduction:

This report details the design, simulation, and performance analysis of a Frequency Modulated Continuous Wave (FMCW) radar system operating in the 76.5 GHz automotive band. The primary objective was to design a radar able to detect multiple moving targets and estimate their range and radial velocity with high precision using MATLAB.

The system utilizes a Linear Frequency Modulated (LFM) waveform with a 1 GHz bandwidth, providing range resolution $\boldsymbol{\Delta R = 0.15}$. The signal processing architecture employs a mixing (de-chirping) technique followed by 2D-FFT processing to generate Range-Doppler Maps (RDM). A critical aspect of the design involved the implementation of a peak detection algorithm. This method effectively reduces spectral leakage and "ghost peaks" caused by windowing sidelobes, ensuring a one-to-one mapping between detected peaks and true targets.

Performance validation was conducted using multi-target scenarios with varying ranges (50m, 150m) and velocities (-50m/s, +80m/s). The simulation results demonstrate accurate target localization, precise velocity estimation, and successful directional classification (approaching vs. receding), validating the effectiveness of the processing chain for automotive radar applications.

System Design and Parameters:

The Carrier Frequency $fc = 76.5\,GHz$.

The BandWidth $BW = 1\,GHz$. ($fmin = -0.5\,GHz\,, fmax = 0.5GHz$)

It's required to Make $Rmax = \frac{C\ Tch\ Fs}{4\ BW} > 250\,m$ and $Vmax = \frac{C}{4\ fc\ Tch} > 100\frac{m}{s}$, so we set $Tch = 2.1\ \mu m$ to achieve $Rmax = 315\,m > 250\,m$ and $Vmax = 466.85\frac{m}{s} > 100\frac{m}{s}$.

The Range Resolution $\Delta R = \frac{C}{2\ BW} = 0.15\,m < 0.75\,m$ as required.

The Velocity Resolution $\Delta V = \frac{C}{2\ fc\ Tch\ Npulses}$ is required to be less than $0.5\frac{m}{s}$, so we set $Npulses = 1880$ so that $\Delta V = 0.49\frac{m}{s} < 0.5\frac{m}{s}$ as required. The sampling time is $dt = 0.5\ ns$.

```
% Parameters
c = 3e8; % Speed of light
fc = 76.5e9; % Carrier frequency = 76.5 GHz
Tch = 2.1e-6;
PRI = 8.4e-6;
PRF = 1/PRI;
fmin = -0.5e9;
fmax = 0.5e9;
BW = fmax - fmin;
s = (fmax - fmin)/Tch;
N_pulses = 1880;
Fs = 2e 9;
dt = 1/Fs;
t = 0:dt:PRI-dt;


%% Targets ------> [ positive = Going away ----------  negative = Approaching ]
R_true = [50, 150];
V_true = [-50, 80];
A_true = [0.8, 0.5];  % Amplitude of targets
L = length(R_true);
```

Signal Generation:

This section details how the theoretical radar equations were translated into MATLAB code to simulate the physical environment.

1. Transmitted Signal:

   The radar waveform is a Linear Frequency Modulated (LFM) chirp. The code generates a single base pulse and then replicates it to create the full transmission sequence. The Transmitted Signal $P = \exp\left(j * 2\pi * (fmin * t + 0.5S * t^2)\right)$ for a single Pulse. This equation implements the mathematical definition of an FMCW chirp.

```
%% Transmitted Pulse
P = exp(1j * 2*pi*( fmin*t + 0.5*s*t.^2 )) .* (t<=Tch);
Tx = repmat(P, N_pulses, 1);
```

- The window $(t <= Tch)$this operation acts as a time gate. It ensures the signal is active only for the chirp duration. For the remainder of the Pulse Repetition Interval PRI, the signal is forced to zero. This simulates the radar's idle time between pulses.
- Pulse Train Construction The $\boldsymbol{repmat(P, N\_pulses, 1)}$ function efficiently stacks the single pulse P vertically $Npulses$ times (1880 times). This results in a matrix Tx where each row represents a distinct pulse in time, establishing the "slow time" dimension required for Doppler processing as we see in Figure 1.

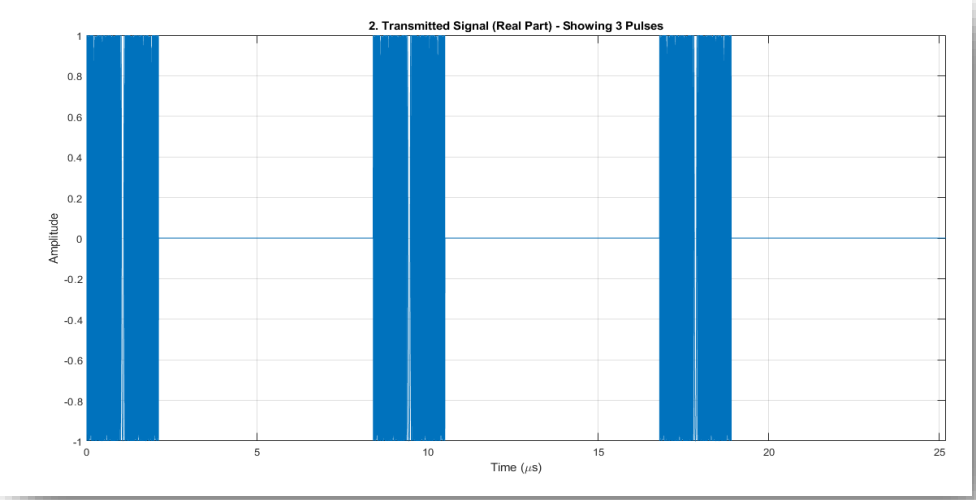

Figure 1

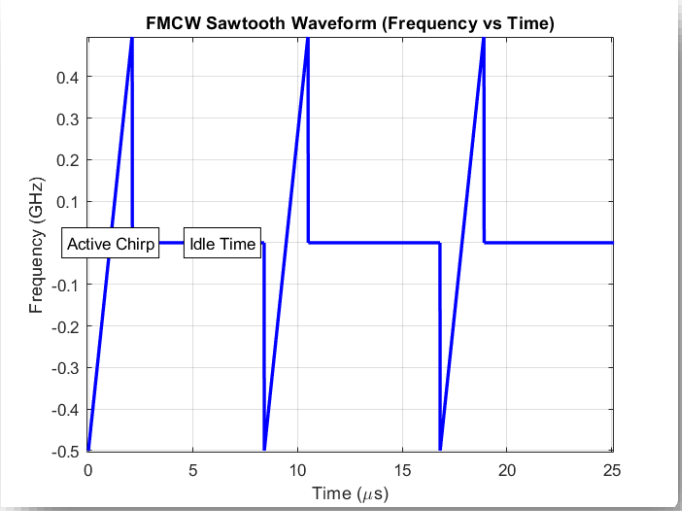

Figure 2

2. Received Signal:

```
%% Received Signal
Rx = zeros(N_pulses, length(t));
SNR_dB = 15;
noise_power = 10^(-SNR_dB/10);

fprintf('Generating Signals for %d Fixed Targets...\n', L);

for n = 1:N_pulses
    pulse_echo = zeros(size(t));
    for k = 1:L
        RTT = 2*R_true(k)/c;
        Ns = round(RTT/dt);
        if Ns+1 <= length(t)
            fd = 2*V_true(k)*fc/c;
            doppler_phase = exp(1j*2*pi*fd*n*PRI);
            shifted = [zeros(1,Ns), P(1:end-Ns)] * A_true(k);
            pulse_echo = pulse_echo + shifted*doppler_phase;
        end
    end
    Rx(n,:) = pulse_echo + sqrt(noise_power/2)*(randn(size(t)) + 1j*randn(size(t)));
end
```

This code simulates the received FMCW radar signal from multiple fixed targets over multiple radar pulses, including:

- Time delay due to target range
- Doppler phase shift due to target velocity
- Additive noise based on a chosen SNR

The output Rx is a matrix where each row is one pulse, and each column is fast-time samples within that pulse.

- Delay Calculation: The code calculates the Round-Trip Time $\left(RTT = \frac{2R}{c}\right)$ for each target. This time delay is converted into a discrete number of samples: $Ns = round\left(\frac{RTT}{dt}\right)$ where dt is the sampling time.
- Doppler Phase Shift: A target's velocity induces a phase shift that changes from pulse to pulse. This is modeled by the term: $phi_{doppler} = exp(j * 2\pi * f_d n T_{PRI})$ where n is the current pulse number. This accumulating phase shift is what the Doppler FFT later detects to estimate velocity.
- Time Shifting: The delay is physically simulated by padding the start of the pulse with $zeros\,(1, Ns)$ and truncating the end. This shifts the signal "to the right" in the time vector, representing the travel time of the wave.
- Signal Superposition: The line $pulse_{echo} = pulse_{echo} + \ldots$ sums the echoes from all L targets. If targets are close together, their signals naturally overlap (interfere), accurately simulating a multi-target environment.
- Noise Addition: Complex Gaussian White Noise is added to the signal to simulate thermal noise and environmental interference. The noise power is scaled according to the specified SNR (5dB,10dB,15 dB) as we see in figure 2, figure 3 and figure 4. Randn() generates normally distributed random numbers. Separate noise components are generated for the real and imaginary parts (1j*randn) to create realistic complex noise.

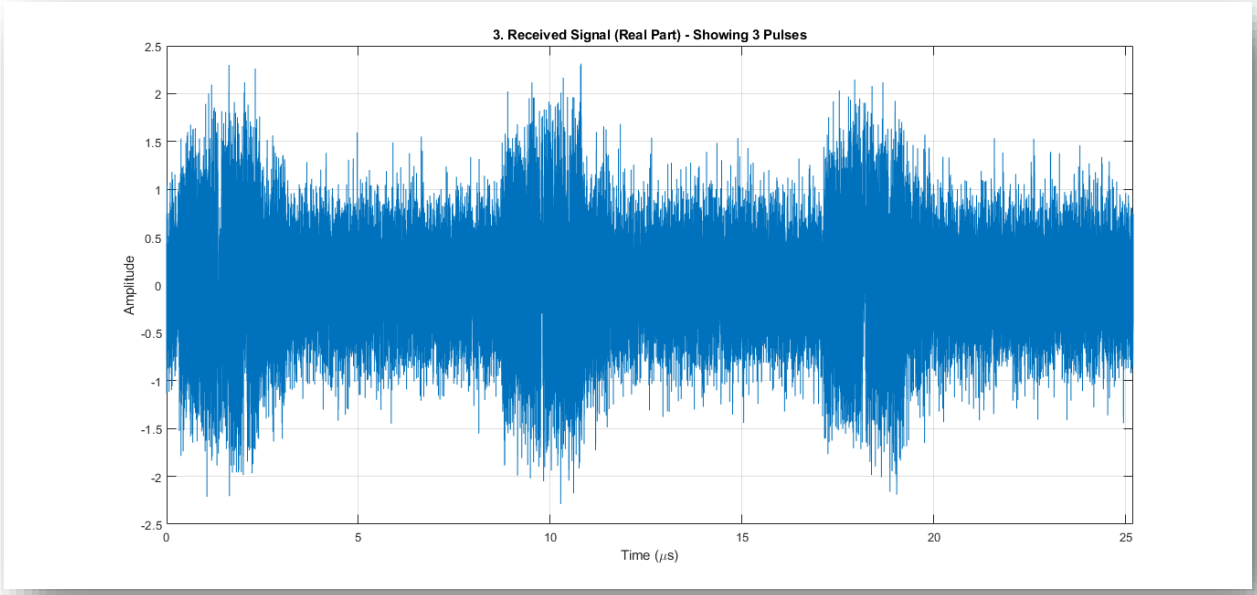

Figure 3 (SNR_ dB=5)

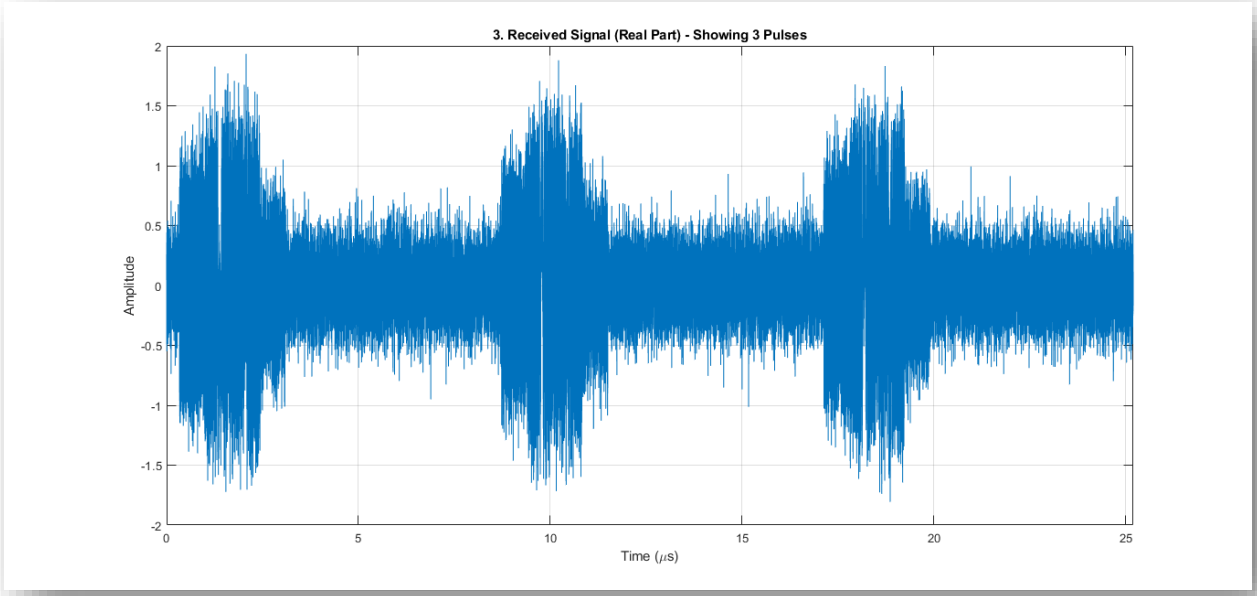

Figure 4 (SNR_ dB=10)

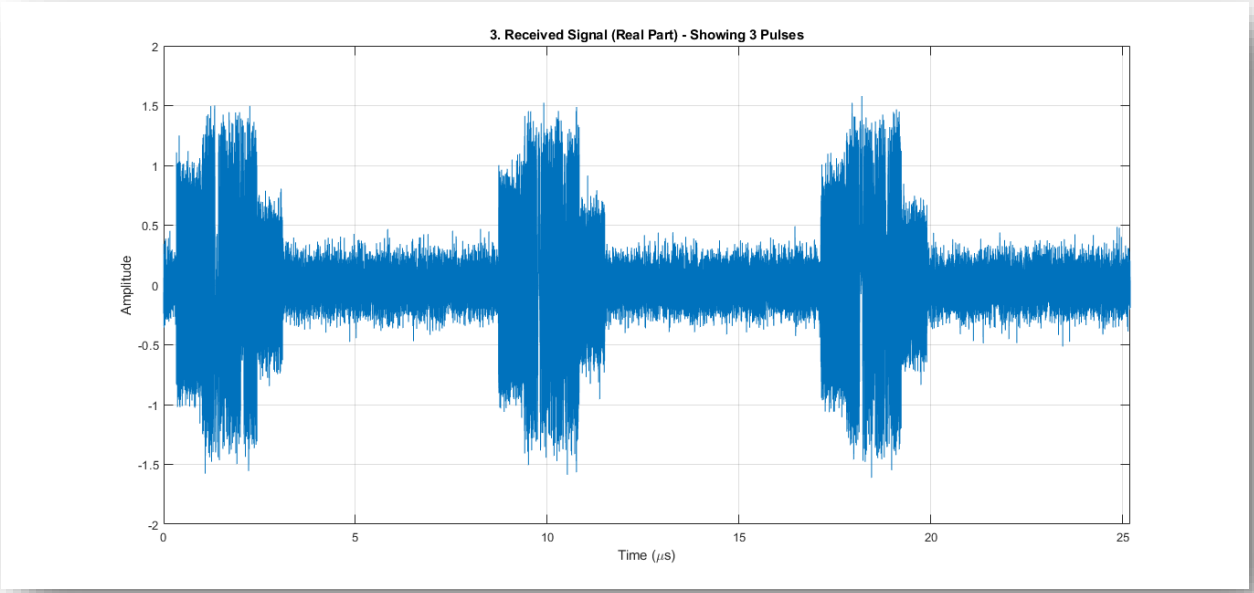

Figure 5 (SNR_dB=15)

3.Effect of Changing SNR on the Received Signal

- Case 1 SNR_dB=5:

  Very high noise, The sinusoidal structure of the chirp is almost unclear and obscured by high-amplitude random spikes.

- Case 2 SNR_dB=10:

  More realistic and the chirp amplitude is visible.

- Case 3 SNR_dB=15:

  It's more close to ideal case.

Range Detection and Velocity Estimation

This block of code takes the raw transmitted and received FMCW radar signals and turns them into physical target measurements by following the standard FMCW processing flow in one continuous chain.

First, the processing timer is started to measure how long the entire signal-processing and detection stage takes. The transmitted signal is then mixed with the complex conjugate of the received signal. This dechirping step removes the carrier and chirp modulation and produces the beat signal, which contains low-frequency components whose frequencies encode target range, while the phase change of those components across pulses encodes target velocity.

Only the portion of the mixed signal that corresponds to the active chirp duration is kept, discarding any idle time between chirps. A fast-time FFT is then applied to this signal for every pulse, converting the beat frequencies into a range spectrum. Zero-padding is used to improve spectral resolution, and only the positive frequency half is retained since FMCW beat frequencies are one-sided. At this point, the data is arranged so that each column corresponds to a range bin and each row corresponds to a pulse.

Next, a slow-time FFT is applied across pulses for every range bin. This extracts Doppler frequency information caused by target motion. After shifting the spectrum so that zero Doppler is centered, the result forms a two-dimensional range–Doppler map whose magnitude represents reflected signal power as a function of range and velocity. The magnitude is converted to decibels for easier interpretation, and the processing timer is stopped. Physical axes are then generated so that FFT bins can be interpreted in real units. Beat frequency bins are mapped to range using the FMCW slope and the speed of light, while Doppler frequency bins are mapped to radial velocity using the radar wavelength and pulse repetition frequency. This allows each point in the range–Doppler map to correspond to a specific distance and speed. For detection, the algorithm first looks at the range spectrum of a single pulse and searches for significant peaks above a relative threshold, ensuring that detected peaks are separated by at least a minimum physical distance to avoid multiple detections of the same target. These peak locations are converted into detected target ranges. For each detected range, a Doppler slice is taken from the Doppler map, and the strongest Doppler peak is selected. This Doppler peak is then converted into a radial velocity estimate. The final output is a set of detected targets, each characterized by an estimated range and velocity, derived from the FMCW radar signal.

1. Mixing:

```
% Signal Processing & Detection
timer_start = tic;
% Mixing
Mix = Tx .* conj(Rx); % The Most Important
```

The code performs element-wise multiplication of the Transmitted signal (Tx) and the complex conjugate of the Received signal, The Result (Beat Signal):Subtracting the phases of two linear chirps results in a signal with a constant frequency, known as the Beat Frequency (fb).The frequency of this beat signal is directly proportional to the target's range.This operation effectively demodulates the signal, removing the high-frequency carrier and leaving only the difference information.

2. Range FFT (Fast-Time Processing):

```
% Range FFT
N_samples_active = round(Tch * Fs);
Mix_active = Mix(:, 1:N_samples_active);
n_fft_range = 2^nextpow2(N_samples_active) * 4;
range_spectrum_matrix = fft(Mix_active, n_fft_range, 2);
range_spectrum_matrix = range_spectrum_matrix(:, 1:n_fft_range/2);
% Doppler FFT
RDM = fftshift(fft(range_spectrum_matrix, N_pulses, 1), 1);
RDM_dB = 10*log10(abs(RDM));
processing_time = toc(timer_start);
% Axis Generation
freq_range = (0:n_fft_range/2-1) * (Fs / n_fft_range);
rng_axis = (freq_range * c) / (2 * s);
lambda = c / fc;
freq_doppler = -(-N_pulses/2 : N_pulses/2 - 1) * (PRF / N_pulses);
vel_axis = (freq_doppler * lambda) / 2;
```

- Truncation (Mix_active): We isolate only the active portion of the pulse ($t \leq T_{ch}$). Processing the idle time (zeros) would introduce unnecessary noise into the spectrum.
- Zero Padding (* 4): The FFT size n_fft_range is calculated to be the next power of 2, multiplied by 4. This technique, called Zero Padding, it just improve the plotting by making it smoother and increasing peak detection accuracy.
- FFT Dimension (dim=2): The FFT is applied along the rows (Fast Time). This converts the time-domain samples into the Range Domain as we see in figures in different SNR (5,10,15).

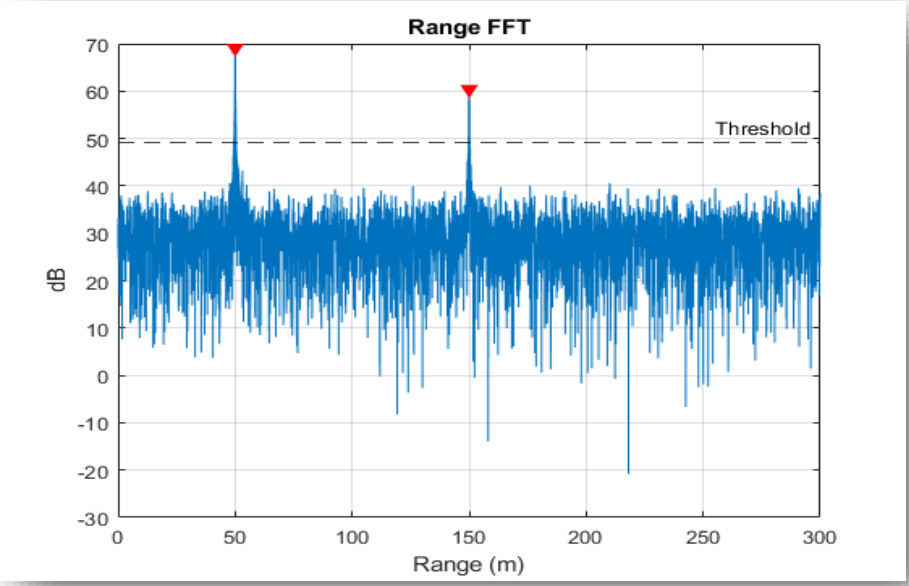

Figure 6 (SNR_dB=5)

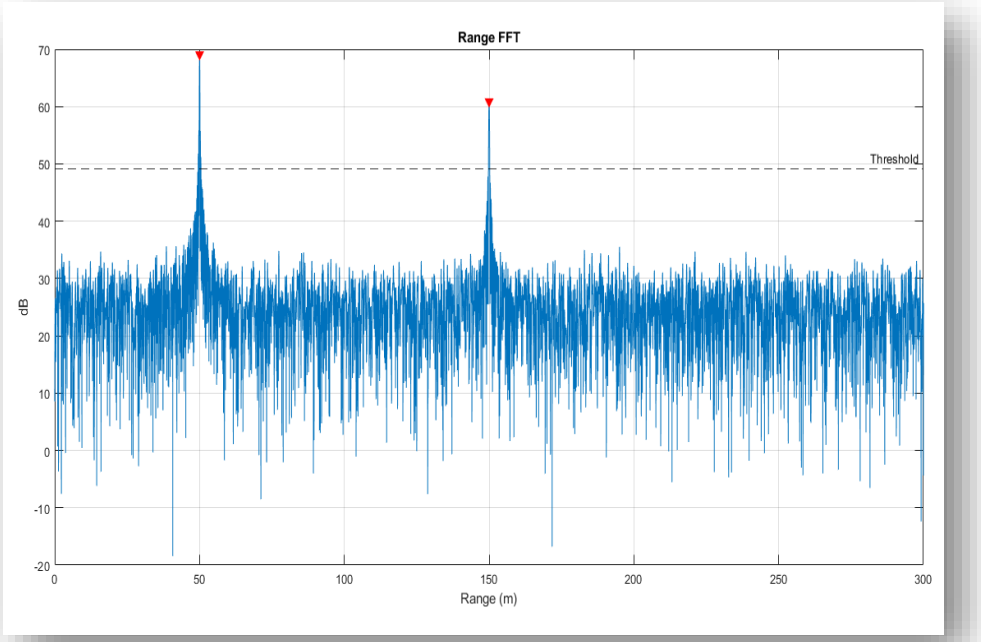

Figure 7 (SNR_ dB=10)

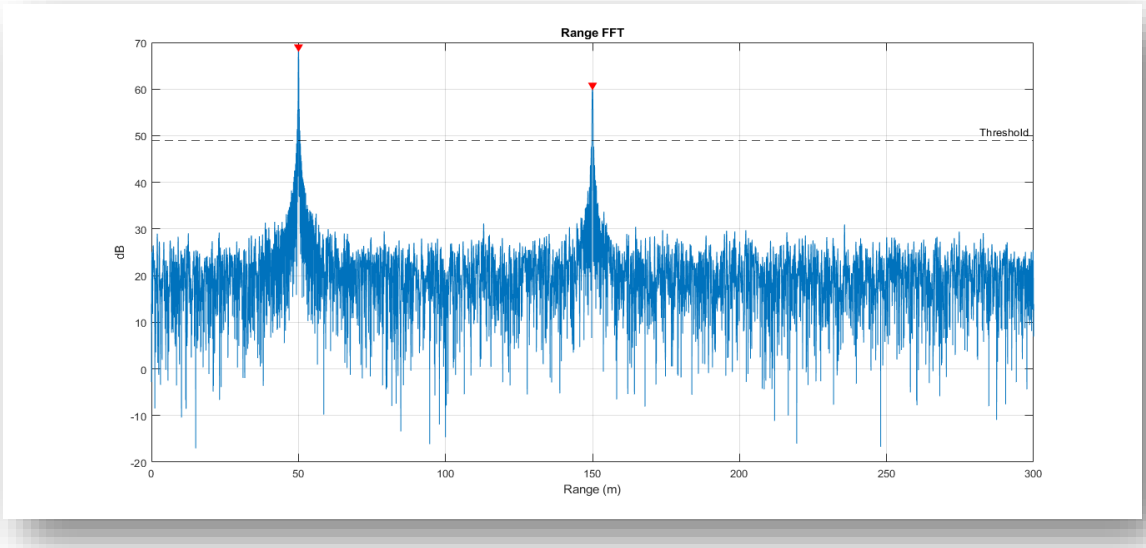

Figure 8 (SNR_ dB=15)

3. Doppler FFT (Slow-Time Processing):

```
% Doppler FFT
RDM = fftshift(fft(range_spectrum_matrix, N_pulses, 1), 1);
RDM_dB = 10*log10(abs(RDM));
processing_time = toc(timer_start);
% Axis Generation
freq_range = (0:n_fft_range/2-1) * (Fs / n_fft_range);
rng_axis = (freq_range * c) / (2 * s);
lambda = c / fc;
freq_doppler = -(-N_pulses/2 : N_pulses/2 - 1) * (PRF / N_pulses);
vel_axis = (freq_doppler * lambda) / 2;
% PEAK DETECTION
% Detect Range Peaks
first_pulse_fft = abs(range_spectrum_matrix(1, :));
threshold = max(first_pulse_fft) * 0.1;
range_resolution = rng_axis(2) - rng_axis(1);
min_dist_meters = 10;
min_dist_indices = round(min_dist_meters / range_resolution);
[pks, locs] = findpeaks(first_pulse_fft, ...
                        'MinPeakHeight', threshold, ...
                        'MinPeakDistance', min_dist_indices);
detected_ranges = rng_axis(locs)
% Detect Velocity
detected_velocities = zeros(1, length(detected_ranges));
for i = 1:length(detected_ranges)
    dop_cut = abs(RDM(:, locs(i)));
    [~, v_idx] = max(dop_cut);
    detected_velocities(i) = vel_axis(v_idx);
end
```

- FFT Dimension (dim=1): A second FFT is applied along the columns (Slow Time). A target moving towards the radar induces a phase shift that changes linearly with each pulse index. This FFT converts that phase progression into Doppler Frequency.
- fftshift: The output of a standard FFT starts at 0 Hz. The fftshift function rearranges the array so that Zero Velocity is in the center, negative velocity is on the left, and positive velocity is on the right. This is essential for distinguishing between Approaching and Receding targets.
- The calculated $fdoppler$ we will put a negative , due to the mixing we did, we took conj(received signal) so we must put a negative.

4. Axis Generation:

- Range Conversion: Based on the FMCW equation $R = \frac{c.fbeat}{2Slope}$
- Velocity Conversion: Based on the Doppler equation $V = \frac{\lambda.fdoppler}{2}$

5. Peak Detection:

- Adaptive Thresholding: The detection threshold is set dynamically to 10% of the maximum signal strength. This ensures that weak noise floor fluctuations are ignored, while strong targets are kept.
- Non-Maximum Suppression: Radar windowing functions create "sidelobes" small ghost peaks adjacent to the real target. The code calculates how many array bins correspond to 10 meters (min_dist_indices).
- The findpeaks function is instructed to enforce a Minimum Peak Distance. If multiple peaks appear within a 10m window, the algorithm compares them and retains only the highest peak. This effectively filters out sidelobes and ensures a clean list of unique targets.

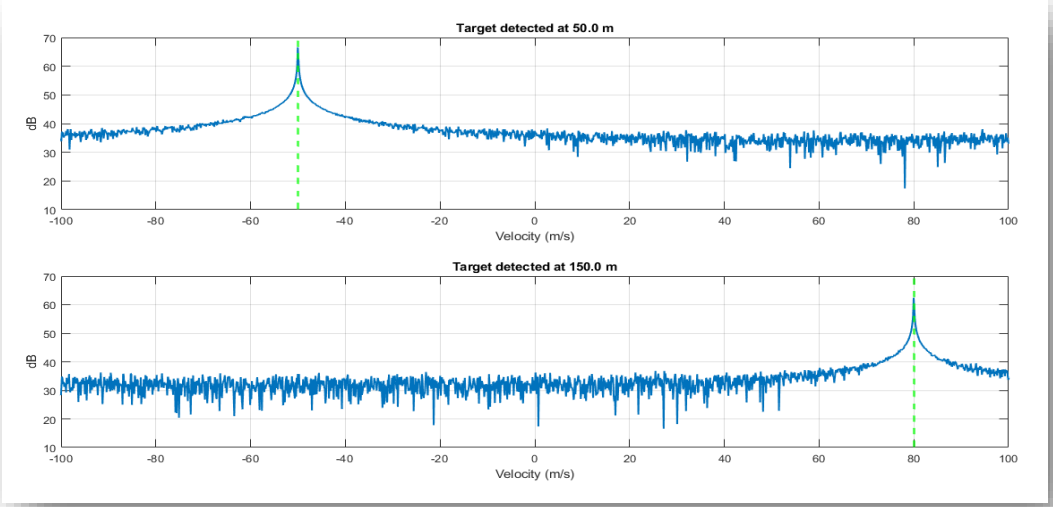

Figure 9 (SNR_dB=5)

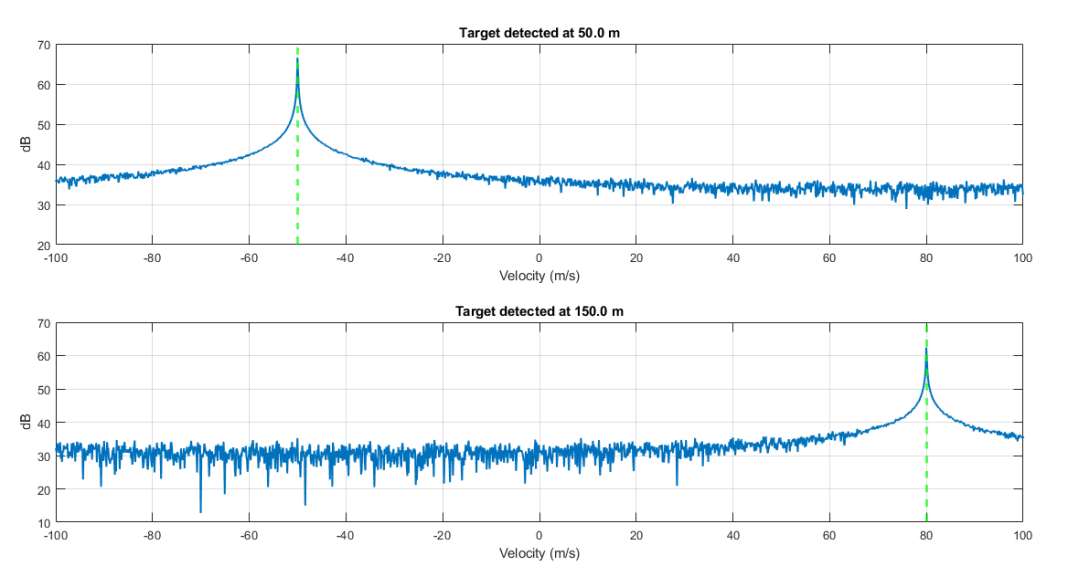

Figure 10 (SNR_ dB=10)

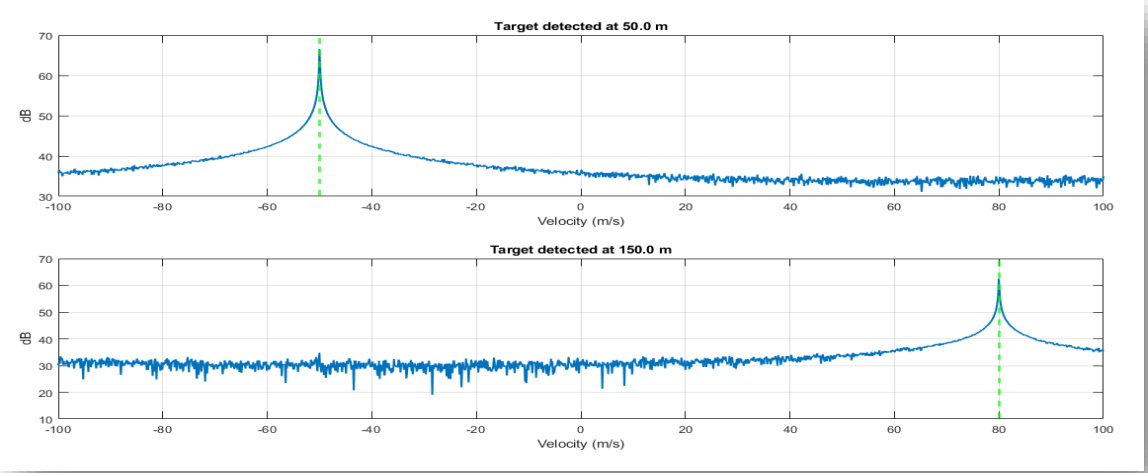

Figure 11 (SNR_ dB=15)

Range FFT Analysis the Range FFT results consistently show target peaks at the exact same range location across all SNR values (5 dB, 10 dB, and 15 dB). This is expected

because the beat frequency is determined solely by the target range and the chirp slope constants that remain unaffected by noise levels. Changing the SNR simply raises or lowers the noise floor; it does not alter the frequency content of the signal.

It is also worth noting that the difference between the 10 dB and 15 dB cases is minimal. This is due to the significant processing gain provided by the FFT. Because each chirp consists of many samples, the resulting coherent integration boosts the effective SNR. Once a target peak rises clearly above the noise floor, increasing the SNR further mostly just cleans up the background noise without shifting the peak itself.

Doppler Spectra Analysis Similarly, the Doppler spectra display peaks at identical velocities regardless of the SNR. This happens because Doppler estimation relies on phase shifts across multiple chirps rather than instantaneous amplitude. The slow-time FFT performs coherent integration across these chirps, which significantly improves the Doppler SNR. As long as the peak is distinguishable from the noise, increasing the signal strength will not impact on the estimated velocity or error rate.

Range-Doppler Map:

Here is the Range-Doppler Map where the negative Velocity means approaching and positive Velocity means going away.

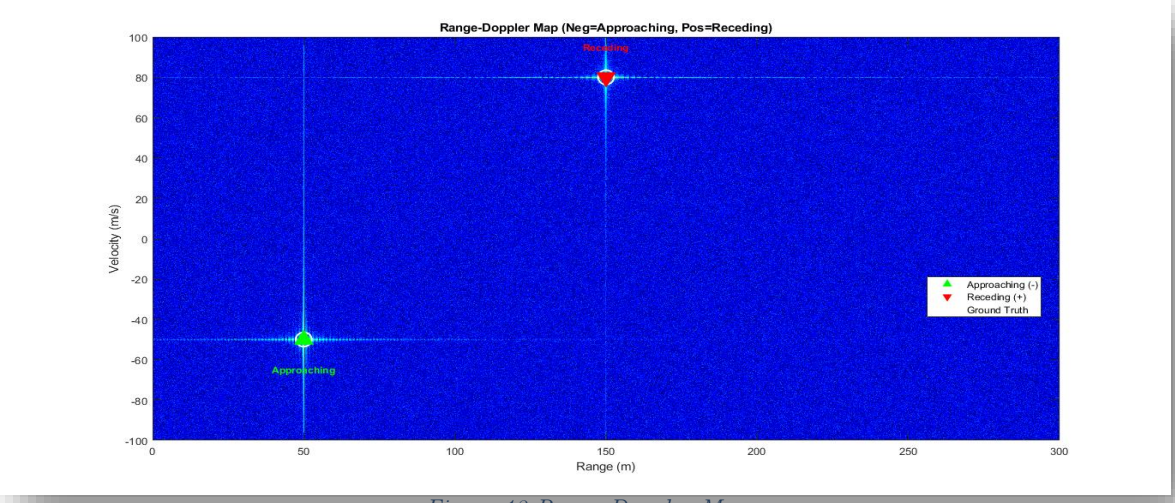

Figure 12 Range-Doppler Map

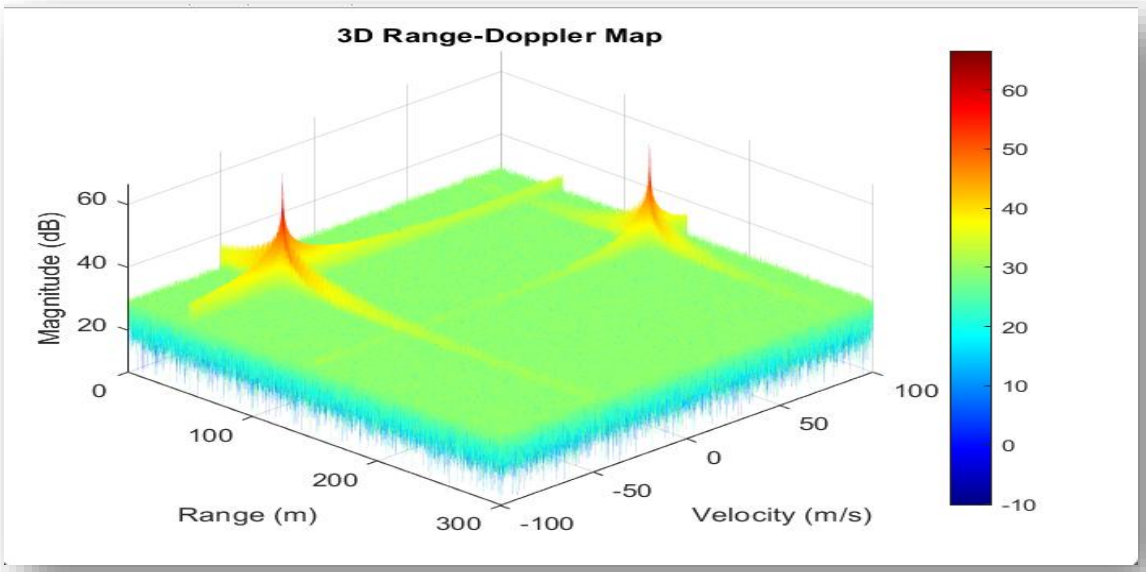

Figure 13 3D Range Doppler Map

Error Estimating for detected and the true Targets:

```
Generating Signals for 2 Fixed Targets...
RANGE DETECTION RESULTS
Table 1 : Processing Time: 2.7723 seconds
True R (m)    | Det R (m)     | Error (m)
50.00         | 50.03         | 0.0262
150.00        | 150.00        | 0.0018
 Table 2 : VELOCITY DETECTION RESULTS
True V        | Det V         | Error       | Direction
-50.00        | -50.04        | -0.0377     | Approaching (-)
80.00         | 79.96         | -0.0389     | Receding (+)
```

Figure 13 (SNR_ dB=5)

```
Generating Signals for 2 Fixed Targets...
RANGE DETECTION RESULTS
Table 1 : Processing Time: 2.1167 seconds
True R (m)    | Det R (m)     | Error (m)
50.00         | 50.03         | 0.0262
150.00        | 150.00        | 0.0018
 Table 2 : VELOCITY DETECTION RESULTS
True V        | Det V         | Error       | Direction
-50.00        | -50.04        | -0.0377     | Approaching (-)
80.00         | 79.96         | -0.0389     | Receding (+)
```

Figure 14 (SNR_ dB=10)

```
Generating Signals for 2 Fixed Targets...
RANGE DETECTION RESULTS
Table 1 : Processing Time: 2.0321 seconds
True R (m)    | Det R (m)     | Error (m)
50.00         | 50.03         | 0.0262
150.00        | 150.00        | 0.0018
 Table 2 : VELOCITY DETECTION RESULTS
True V        | Det V         | Error       | Direction
-50.00        | -50.04        | -0.0377     | Approaching (-)
80.00         | 79.96         | -0.0389     | Receding (+)
```

Figure 14 (SNR_ dB=15)

Range-Velocity Map Analysis the Range-Velocity maps remain remarkably consistent across different SNR levels because the 2D FFT effectively combines processing gains from

both range and Doppler dimensions. While noise is distributed across the entire map, target energy is focused into specific range-velocity bins, keeping the signal distinct.

At a lower SNR of 5 dB, the background noise is naturally higher, yet the targets remain clearly distinguishable. As the SNR increases, the noise floor drops, which enhances visual clarity but does not alter the target locations. This confirms that as long as the radar operates above the minimum detectable threshold, increasing the SNR primarily serves to reduce background noise rather than shifting detection coordinates or significantly changing estimation accuracy.

Conclusion:

In this project, a complete FMCW radar signal model and processing chain was developed and evaluated, starting from waveform transmission and target echo simulation and ending with target detection and parameter estimation. The radar scenario included 2 targets with different ranges and velocities, and effects such as round-trip propagation delay, Doppler-induced phase shifts, and additive noise were incorporated to closely represent real-world operation.

The received signals were processed using standard FMCW radar techniques. Dechirping through conjugate mixing converted time delays into beat frequencies, enabling extraction of range information through fast-time Fourier transforms. The slow-time Fourier transforms across pulses allowed Doppler frequencies to be calculated, producing a two-dimensional range–Doppler map that clearly separated targets in both distance and velocity. Physical range and velocity axes were derived from system parameters.

A peak-based detection approach was applied to identify targets in range and to estimate their corresponding velocities from the range–Doppler map. The results demonstrate that the implemented processing chain successfully detects multiple targets and accurately estimates their ranges and radial velocities under noisy conditions. The measured processing time also provides insight into the computational cost of the algorithm, which is relevant for real-time radar applications.

Overall, this project validates the fundamental principles of FMCW radar operation and demonstrates how waveform design, signal processing, and detection techniques work together to extract target information.

Full Code & GitHub:

https://github.com/a7med57/FMCW-Radar.git

```
% Parameters
c = 3e8; % Speed of light
fc = 76.5e9; % Carrier frequency = 76.5 GHz
Tch = 2.1e-6;
PRI = 8.4e-6;
PRF = 1/PRI;
fmin = -0.5e9;
fmax = 0.5e9;
BW = fmax - fmin;
s = (fmax - fmin)/Tch;
N_pulses = 1880;
Fs = 2e9;
dt = 1/Fs;
t = 0:dt:PRI-dt;


%% Targets ------> [ positive = Going away -----------  negative =
Approaching ]
```

```
R_true = [50, 150];
V_true = [-50, 80];
A_true = [0.8, 0.5];  % Amplitude of targets
L = length(R_true);

%% Transmitted Pulse
P = exp(1j * 2*pi*( fmin*t + 0.5*s*t.^2 )) .* (t<=Tch);
Tx = repmat(P, N_pulses, 1);


%% Received Signal
Rx = zeros(N_pulses, length(t));
SNR_dB = 15;
noise_power = 10^(-SNR_dB/10);

fprintf('Generating Signals for %d Fixed Targets...\n', L);

for n = 1:N_pulses
    pulse_echo = zeros(size(t));
    for k = 1:L
        RTT = 2*R_true(k)/c;
        Ns = round(RTT/dt);
        if Ns+1 <= length(t)
            fd = 2*V_true(k)*fc/c;
            doppler_phase = exp(1j*2*pi*fd*n*PRI);
            shifted = [zeros(1,Ns), P(1:end-Ns)] * A_true(k);
            pulse_echo = pulse_echo + shifted*doppler_phase;
        end
    end
    Rx(n,:) = pulse_echo + sqrt(noise_power/2)*(randn(size(t)) +
1j*randn(size(t)));
end

% Signal Processing & Detection
timer_start = tic;
% Mixing
Mix = Tx .* conj(Rx); % The Most Important
% Range FFT
N_samples_active = round(Tch * Fs);
Mix_active = Mix(:, 1:N_samples_active);
n_fft_range = 2^nextpow2(N_samples_active) * 4;
range_spectrum_matrix = fft(Mix_active, n_fft_range, 2);
range_spectrum_matrix = range_spectrum_matrix(:, 1:n_fft_range/2);
% Doppler FFT
RDM = fftshift(fft(range_spectrum_matrix, N_pulses, 1), 1);
RDM_dB = 10*log10(abs(RDM));
processing_time = toc(timer_start);
% Axis Generation
freq_range = (0:n_fft_range/2-1) * (Fs / n_fft_range);
```

```
rng_axis = (freq_range * c) / (2 * s);
lambda = c / fc;
freq_doppler = -(-N_pulses/2 : N_pulses/2 - 1) * (PRF / N_pulses);
vel_axis = (freq_doppler * lambda) / 2;
% PEAK DETECTION
% Detect Range Peaks
first_pulse_fft = abs(range_spectrum_matrix(1, :));
threshold = max(first_pulse_fft) * 0.1;
range_resolution = rng_axis(2) - rng_axis(1);
min_dist_meters = 10;
min_dist_indices = round(min_dist_meters / range_resolution);
[pks, locs] = findpeaks(first_pulse_fft, ...
                        'MinPeakHeight', threshold, ...
                        'MinPeakDistance', min_dist_indices);
detected_ranges = rng_axis(locs);
% Detect Velocity
detected_velocities = zeros(1, length(detected_ranges));
for i = 1:length(detected_ranges)
    dop_cut = abs(RDM(:, locs(i)));
    [~, v_idx] = max(dop_cut);
    detected_velocities(i) = vel_axis(v_idx);
end


%%  Plotting Section
N_show = 3;
Tx_show = reshape(Tx(1:N_show, :).', 1, []);
Rx_show = reshape(Rx(1:N_show, :).', 1, []);
t_show  = linspace(0, N_show*PRI, length(Tx_show));
% Frequency Sawtooth
t_one_pulse = linspace(0, PRI, 1000);
freq_one_pulse = zeros(size(t_one_pulse));
active_idx = t_one_pulse <= Tch;
freq_one_pulse(active_idx) = linspace(fmin, fmax, sum(active_idx));
t_sawtooth = linspace(0, N_show*PRI, 1000*N_show);
freq_sawtooth = repmat(freq_one_pulse, 1, N_show);


% Transmitted Signal
figure;
plot(t_show*1e6, real(Tx_show));
title(sprintf('2. Transmitted Signal (Real Part) - Showing %d Pulses',
N_show));
ylabel('Amplitude');
xlabel('Time (\mus)');
xlim([0, N_show*PRI*1e6]);
grid on;
figure('Name', 'FMCW Sawtooth', 'Color', 'w');
plot(t_sawtooth*1e6, freq_sawtooth/1e9, 'b', 'LineWidth', 2);
```

```
117
118  title('FMCW Sawtooth Waveform (Frequency vs Time)');
119  ylabel('Frequency (GHz)');
120  xlabel('Time (\mus)');
121  grid on;
122  xlim([0, N_show*PRI*1e6]);
123
124  % Add Annotations
125  text(Tch*1e6/2, 0, 'Active Chirp', 'HorizontalAlignment', 'center', ...
126      'BackgroundColor', 'w', 'EdgeColor', 'k');
127  text(PRI*1e6 - 2, 0, 'Idle Time', 'HorizontalAlignment', 'center', ...
128      'BackgroundColor', 'w', 'EdgeColor', 'k');
129  % Received Signal
130  figure;
131  plot(t_show*1e6, real(Rx_show));
132  title(sprintf('3. Received Signal (Real Part) - Showing %d Pulses',
133  N_show));
134  ylabel('Amplitude');
135  xlabel('Time (\mus)');
136  xlim([0, N_show*PRI*1e6]);
137  grid on;
138
139  % Figure 2: Range FFT Detection
140  figure('Name', 'Range Detection', 'Color', 'w');
141  plot(rng_axis, 20*log10(abs(range_spectrum_matrix(1,:)))); hold on;
142  plot(detected_ranges, 20*log10(pks), 'rv', 'MarkerFaceColor', 'r');
143  yline(20*log10(threshold), '--k', 'Threshold');
144  title('Range FFT');
145  xlabel('Range (m)'); ylabel('dB'); grid on; xlim([0, 300]);
146
147  % Figure 3: Doppler Spectrum
148  figure('Name', 'Doppler Spectrum', 'Color', 'w');
149  num_plots = length(detected_ranges);
150  for i = 1:num_plots
151      subplot(num_plots, 1, i);
152      dop_cut_dB = RDM_dB(:, locs(i));
153      plot(vel_axis, dop_cut_dB, 'LineWidth', 1.5); hold on;
154      [min_val, true_idx] = min(abs(R_true - detected_ranges(i)));
155      if min_val < 10
156          xline(V_true(true_idx), '--g', 'LineWidth', 2);
157      end
158
159      title(sprintf('Target detected at %.1f m', detected_ranges(i)));
160      xlabel('Velocity (m/s)'); ylabel('dB');
161      grid on; xlim([-100, 100]);
162  end
163
164  %  FIGURE 4: Range-Doppler Map
165  figure('Name', 'Range-Doppler Map (Direction)', 'Color', 'w');
```

```
imagesc(rng_axis, vel_axis, RDM_dB);
axis xy; colormap('jet');
clim([max(RDM_dB(:))-45, max(RDM_dB(:))]);

hold on;
plot(R_true, V_true, 'wo', 'MarkerSize', 14, 'LineWidth', 1.5);

for m = 1:length(detected_ranges)
    r_val = detected_ranges(m);
    v_val = detected_velocities(m);

    if v_val < -0.5
        % NEGATIVE = APPROACHING
        plot(r_val, v_val, '^', 'Color', 'g', 'MarkerSize', 12, ...
            'LineWidth', 2, 'MarkerFaceColor', 'g');
        text(r_val, v_val - 15, 'Approaching', 'Color', 'g', ...
            'FontWeight', 'bold', 'FontSize', 9, 'HorizontalAlignment',
'center');

    elseif v_val > 0.5
        % POSITIVE = RECEDING
        plot(r_val, v_val, 'v', 'Color', 'r', 'MarkerSize', 12, ...
            'LineWidth', 2, 'MarkerFaceColor', 'r');
        text(r_val, v_val + 15, 'Receding', 'Color', 'r', ...
            'FontWeight', 'bold', 'FontSize', 9, 'HorizontalAlignment',
'center');
    end
end

title('Range-Doppler Map (Neg=Approaching, Pos=Receding)');
xlabel('Range (m)');
ylabel('Velocity (m/s)');
xlim([0, 300]);
ylim([-100, 100]);
h1 = plot(NaN,NaN,'^g', 'MarkerFaceColor','g');
h2 = plot(NaN,NaN,'vr', 'MarkerFaceColor','r');
h3 = plot(NaN,NaN,'wo', 'LineWidth', 1.5);
legend([h1, h2, h3], 'Approaching (-)', 'Receding (+)', 'Ground Truth',
'Location', 'best');

%  FIGURE 5: 3D Range-Doppler Map
figure('Name', '3D Range-Doppler', 'Color', 'w');
[R_grid, V_grid] = meshgrid(rng_axis, vel_axis);
% Plot Surface
surf(R_grid, V_grid, RDM_dB, 'EdgeColor', 'none');
colormap('jet'); colorbar;
view(45, 45);
xlim([0, 300]); ylim([-100, 100]);
zlim([max(RDM_dB(:))-60, max(RDM_dB(:))]);
```

```
title('3D Range-Doppler Map');
xlabel('Range (m)'); ylabel('Velocity (m/s)'); zlabel('Magnitude (dB)');


%% Printing in Command Window
fprintf('RANGE DETECTION RESULTS\n');
fprintf('Table 1 : Processing Time: %.4f seconds\n', processing_time);
fprintf('%-12s | %-12s | %-10s\n', 'True R (m)', 'Det R (m)', 'Error
(m)');
[R_true_sorted, sort_idx] = sort(R_true);
V_true_sorted = V_true(sort_idx);

for k = 1:L
   [val, idx] = min(abs(detected_ranges - R_true_sorted(k)));
   if val < 10
       det_r = detected_ranges(idx);
       err_r = det_r - R_true_sorted(k);
       fprintf('%-12.2f | %-12.2f | %-10.4f\n', R_true_sorted(k), det_r,
err_r);
   else
       fprintf('%-12.2f | %-12s | %-10s\n', R_true_sorted(k), 'MISSED',
'-');
   end
end

fprintf(' Table 2 : VELOCITY DETECTION RESULTS \n');
fprintf('%-12s | %-12s | %-10s | %-15s\n', 'True V', 'Det V', 'Error',
'Direction');
for k = 1:L
   [val, idx] = min(abs(detected_ranges - R_true_sorted(k)));
   if val < 10
       det_v = detected_velocities(idx);
       err_v = det_v - V_true_sorted(k);
       if det_v < -0.1
           dir_str = 'Approaching (-)';
       elseif det_v > 0.1
           dir_str = 'Receding (+)';
       else
           dir_str = 'Static';
       end

       fprintf('%-12.2f | %-12.2f | %-10.4f | %-15s\n',
V_true_sorted(k), det_v, err_v, dir_str);
   else
       fprintf('%-12.2f | %-12s | %-10s | %-15s\n', V_true_sorted(k),
'MISSED', '-', '-');
   end
end
```

References:

- Dr Mohammed Explanation Video.
- https://youtube.com/playlist?list=PLPQrZqhah1qFoVZyZB_QQ9iNSmiJyaHp7&si=0XnYpIN_9RrB4-8k
- MathWorks Radar Basics