Project 2: Motion Detection via Communication Signals

Author	Student Name & ID: 万芃（12011615）、王卓扬（12112907）、刘谦益（12011812）、冯彦捷（12010825）

Introduction

Background Information

Multipath Propagation

Signals reaching the receiving antenna by two or more paths. Because of the existence of buildings and the mountain, signals may be reflected, thus the multipath propagation formed.

Multipath propagation led to attenuation and limit the broadband and transmitting speed. The signal received by the receiver is the superposition of the direct wave and many reflected waves.

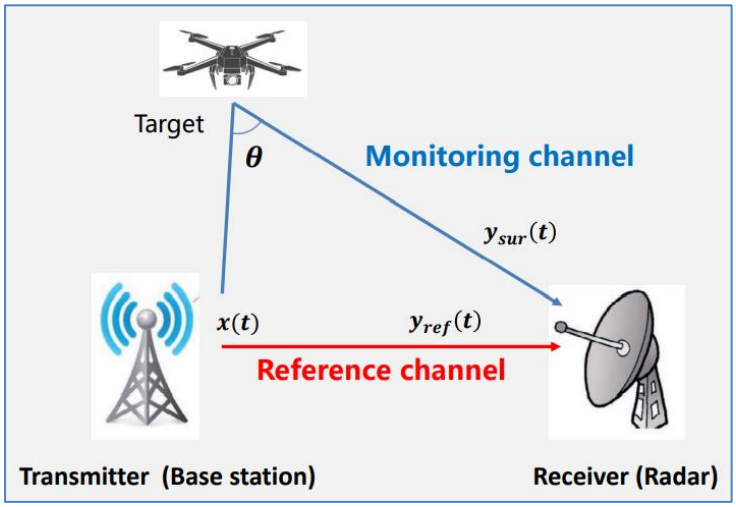


Figure 1: A Simple Model for Multipath Propagation

Denote the signal as $x(t)$, we can define that

$$y_{ref}(t) = \alpha x(t - \tau_r) \quad (1)$$

$$y_{sur}(t) = \beta x(t - \tau_s) e^{j2\pi f_d t} \quad (2)$$

Doppler Shift

When the source and observer of a signal are moving relatively, the frequency will change. Increase when approaching (which is called a “blue shift”), decreasing when move apart (which is called a “red shift”).

The wave source will send out a wave after vibration, and the receiver will receive different frequency due to relative movement.

$$f' = \left(\frac{v \pm v_0}{v \mp v_s}\right) f \tag{3}$$

Radar

The device which can analyze high frequency radio waves reflected from surfaces of objects to determining their position and velocity.

There are active radars: Both transmitter and receiver are radar. Transmitter and receiver located nearly. And Passive radars: Base station serve as transmitter and radar is the receiver. Receive a reference signal from a transmitter through a direct path. Then receive a scattered signal from the target. Calculating the time difference and frequency difference between the samples of the two signals.

Objective

The objective of this project is that we have a runner (point B) on the platform of the *Run Yang Gymnasium*, running back and forward. There is a working base station (point A) perpendicular to the running route directly opposite. We set up two antennas at the foot of perpendicular (point O). These are all illustrated in *Figure 2*.

Now we want to know how the runner's motion is like, only according to the signals we received.

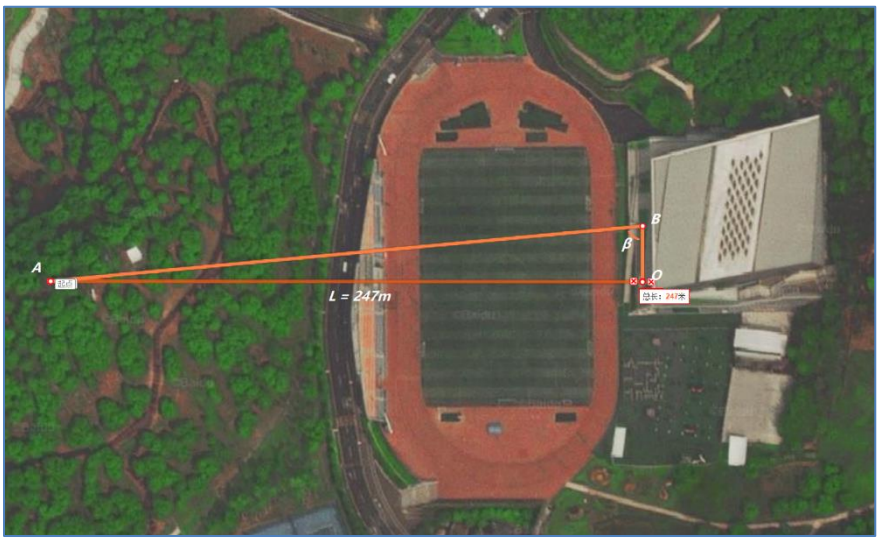


Figure 2: The Top View of the Site

Methodology

Motion Analysis

In the model described in section *Multipath Propagation (Figure 1)*, we defined $y_{ref}(t)$ and $y_{sur}(t)$. What we need to do is to figure out two important things:

(1) $\Delta\tau = |\tau_r - \tau_s|$. The time shift between two signals. We can calculate the distance difference ΔL by multiplying Δt with the light speed c.

$$\Delta \mathrm{L} = \Delta t \cdot c \tag{4}$$

(2) f_d. The doppler frequency shift.

When we know the two parameters, we can know the runner's velocity at time t using the *Formula 5*, λ is the center wavelength of the signal.

$$v = \frac{\lambda \cdot f_d}{2\cos\frac{\beta}{2}} \tag{5}$$

To get the value of β, We used *Baidu Map*, and measured that $L = |OA|$ is about *247 meters*. Denote displacement of the runner is $x(t) = |OB|$, $x(t) > 0$. And using some geometry knowledge we have $|AB| = \sqrt{L^2 + x^2(t)}$. The distance difference of two signals is $|AB| + |BO| - |AO|$, so we have the relationship

$$\sqrt{L^2 + x^2(t)} + x(t) - L = \Delta L = c \cdot \Delta t \tag{6}$$

And in the triangle, we have

$$\beta = tan^{-1}\frac{L}{x(t)} \tag{7}$$

All the above, we can now use Δt and f_d to figure out $v(t)$. To obtain a smoother $x(t)$ we can integrate $v(t)$ again to achieve that. And we think that repeat this process for several times will make the result be stable. The whole process is like: $\Delta t \rightarrow \Delta L \rightarrow x(t) \rightarrow \beta \rightarrow v(t) \rightarrow x_{smooth}(t)$.

Ambiguity Function Process

The next question is, how to figure out Δt and f_d? To solve this problem, we use the *Cross-correlation Function* (CCF), which is related to the self-correlation Function we used in echo cancellation. We define the ambiguity function as:

$$\begin{aligned} Cor(c,d) &= \int_t^{t+T} y_{sur}(t+c)\left[y_{ref}(t)e^{j2\pi dt}\right]^* \\ &= \int_t^{t+T} y_{sur}(t+c)y_{ref}^*(t)e^{-j2\pi dt} \end{aligned} \tag{8}$$

This function describes the correlation of the signals of the two channels after correction for a time shift of c and a frequency shift of d. Which means, when (c,d) is approaching the real value $(\Delta t, f_d)$, the magnitude of the function gets larger.

So, we can just obtain Δt and f_d by maximizing the ambiguity function:

$$(\Delta t, f_d) = \underset{(c,d)}{arg\max}\, Cor(c,d) \tag{9}$$

That all about the methodology in this project.

Lab Results & Analysis

Task 1 Plotting the Spectrums

Step 1. Load Data

The radar receives the base band signals distributed majorly in two frequency bands: 2110 ~ 2130 MHz (20 MHz bandwidth) and 2130 ~ 2135 MHz (5 MHz bandwidth).

In this step, we need to read data from the package provided as *.mat* files, from *data_1.mat* to *data_20.mat*, using the command *load*.

Step 2. Digital Down Convert

Digital Down Convert (DDC) can eliminate imbalance-related distortion created by an analog IF mixer and it avoids phase distortion from analog filters. After the DDC, the sample rate is significantly reduced, and we can have a more efficient implementation of the DSP routines that further process the data.

Which we need is the information in the stronger band, i.e., in this case is the 2110 ~ 2130 MHz band. So, in this step we use DDC to move the center of the 2110 ~ 2130 MHz band to the origin point in the frequency domain. This is the preparation for low pass filtering later.

Step 3. Low Pass Filtering

Next, we use LPF to extract the information in the 2110 ~ 2130 MHz band. Then, the base station signals of 2130 ~ 2135MHz (5M bandwidth) shall be filtered, while signals of 2110 ~ 2130 MHz (20M bandwidth) are going to be retained.

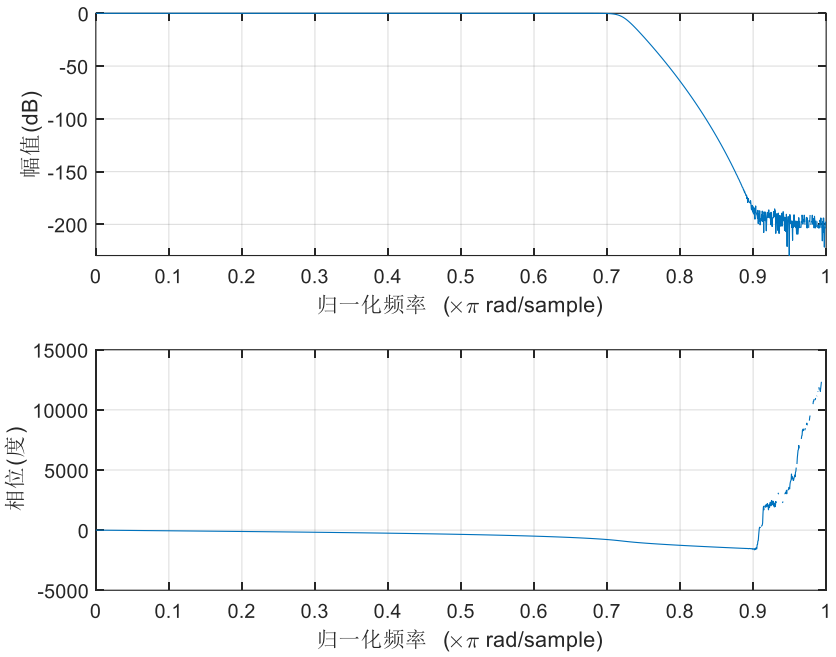


Figure 3: The Frequency Response of the LPF

Now, we gain the figures of the reference signals and surveillance signals in both time domain and frequency domain. We plotted all of them for all the 20 data files. **And we will show one of them here (*Figure 4*), the other will be shown in the *Appendix*:**

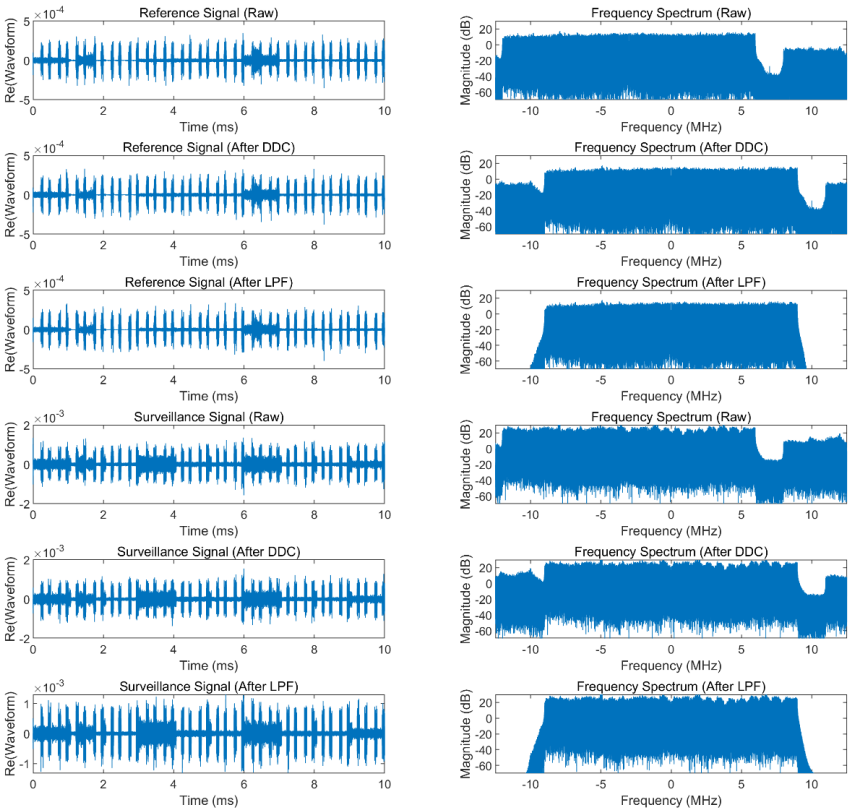


Figure 4: Example – Spectrums for Data 01

Task 2 Plotting the Heatmaps

Step 1. Ambiguity Function Process

From the section *Methodology* we know how to calculate Δt and f_d. In this step we will implement this procedure.

For each data file we take 6 sampling points in time shift (0 ~ 5 sampling intervals) and 40 sampling points in frequency shift (-40 ~ 38 Hz). Then we iterate through all the above points and calculate the ambiguity function values. Do this for all 20 data files.

(To save time, we used *gpuArray* to accelerate the process.)

Step 2. Plot the Heatmaps for The Ambiguity Function

Then we get a small heatmap for this data file, and here's some examples (*Figure 5 ~ 9,* ***other figures are shown in Appendix***).

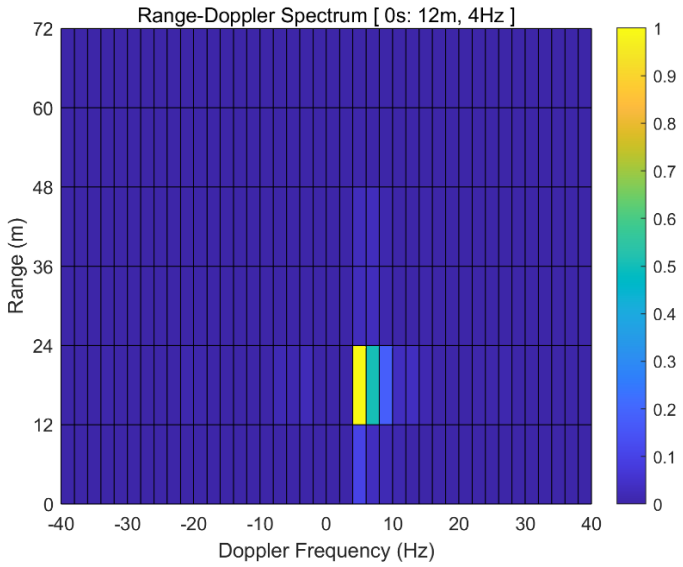


Figure 5: Example – Heatmap for Ambiguity Function for Data 01

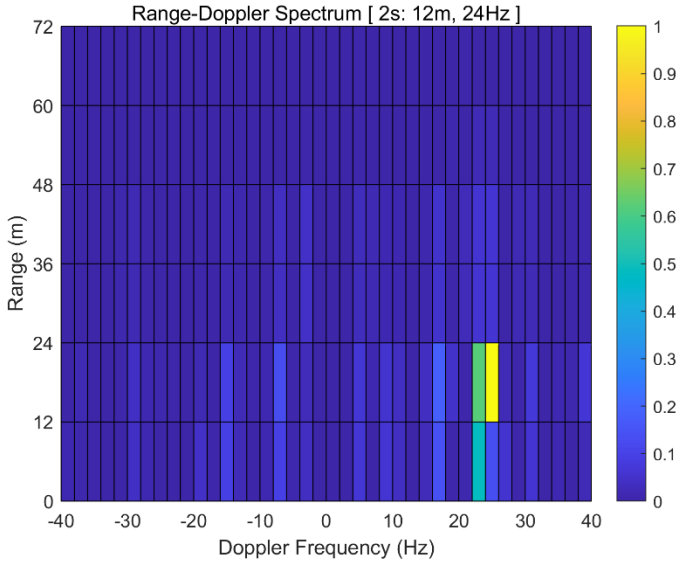


Figure 6: Example – Heatmap for Ambiguity Function for Data 05

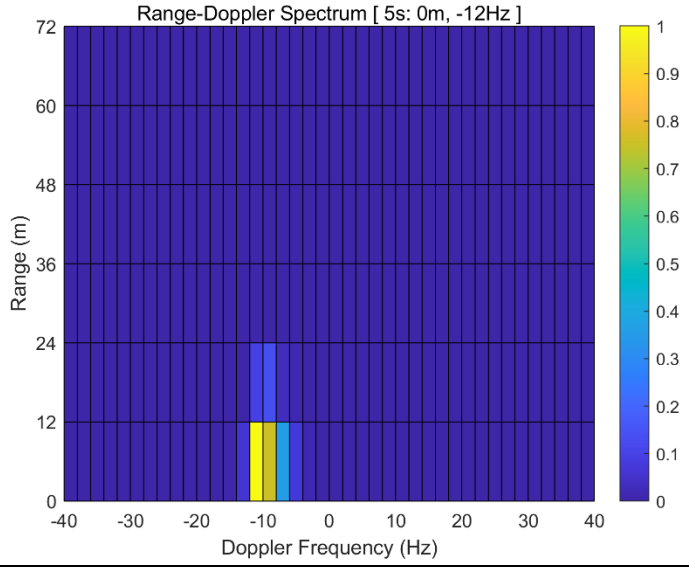


Figure 7: Example – Heatmap for Ambiguity Function for Data 11

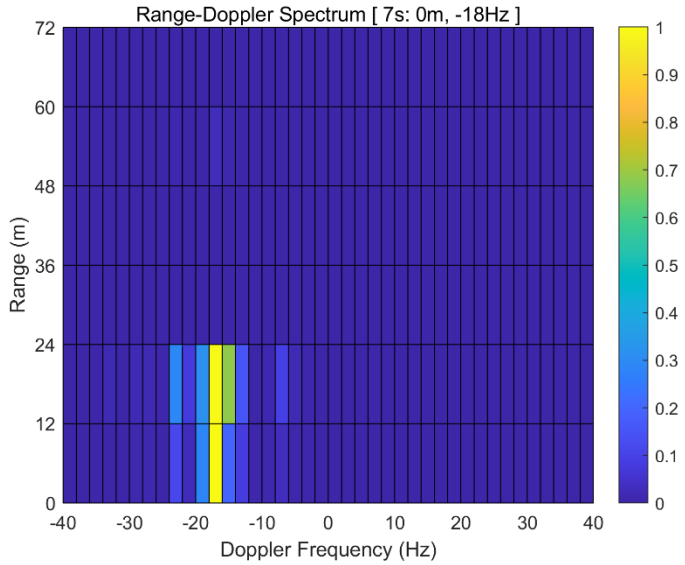


Figure 8: Example – Heatmap for Ambiguity Function for Data 15

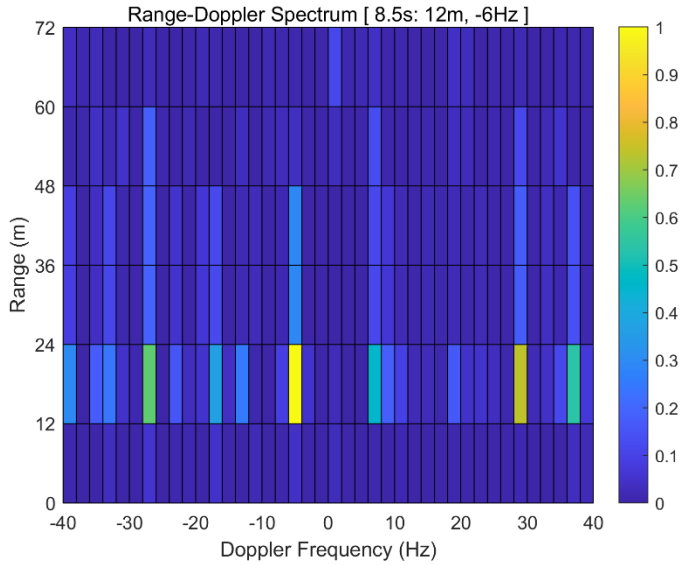


Figure 9: Example – Heatmap for Ambiguity Function for Data 18

We did some transform on the function values to make the figure clearer, such as normalizing the values into 0 ~ 1, and calculating the cubes of them.

Step 3. Plot the Doppler Frequency – Time Relation

Then we selected more sampling points in timeline and plotted a big heatmap for the Doppler Frequency – Time Relation (*Figure 10*).

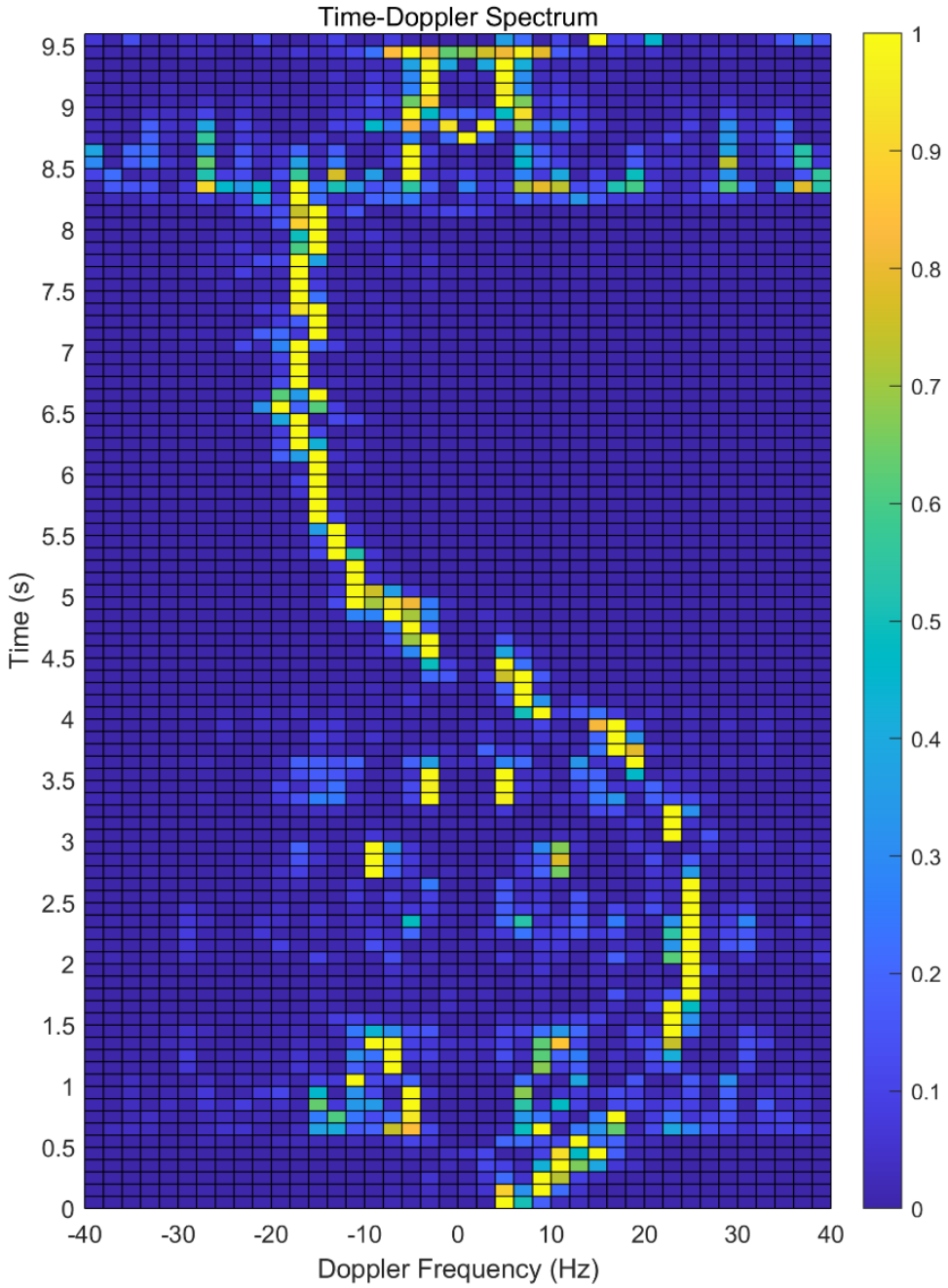


Figure 10: Doppler Frequency – Time Relation

In addition, we respectively took more sampling points in timeline and doppler shift. They are shown in *Figure 11*.

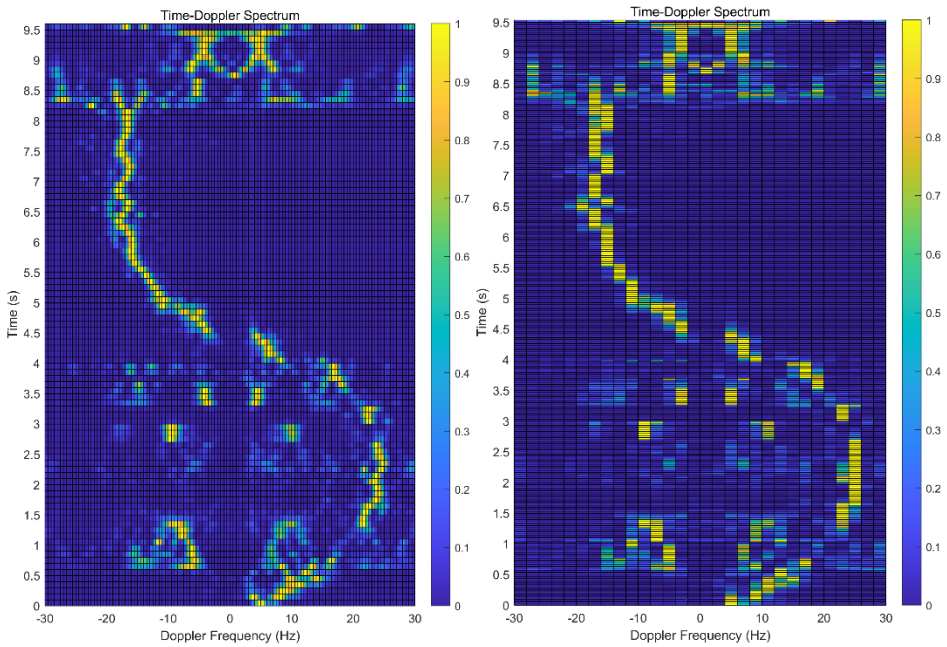


Figure 11: More Accurate Figures

Task 3 Calculating $x(t)$ and $v(t)$

Use the method in section *Methodology*, we can calculate the velocity and displacement of the runner (*Figure 12*).

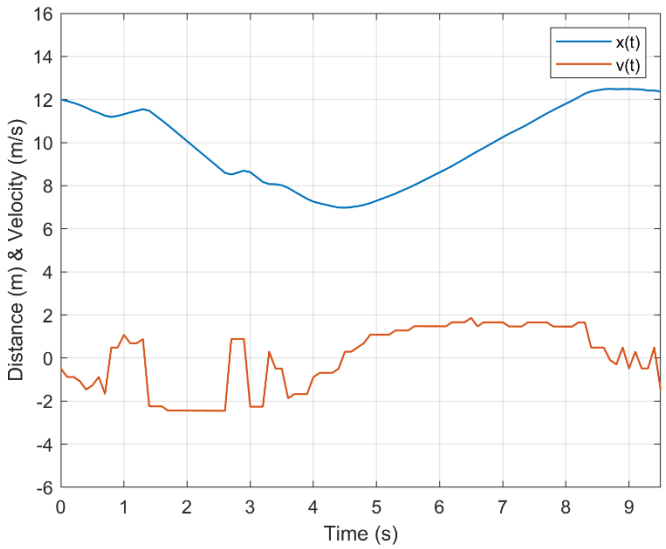


Figure 12: The Velocity and Displacement of the Runner

Extension

Radar Applications

Radar can only receive signals from one direction, so when we need to search target of all other directions, we need to build more radars toward different direction.

Micro Doppler mm-wave Radar

When electron wave hit at the surface of relatively moving object, the frequency of reflection wave will change. Moreover, vibration or rotation of parts object will lead to micro doppler effect. Doppler effect will be used to determine the speed of the object and micro doppler effect will be used to determine the character of the object.

$$f_{doppler} = \frac{2 \cdot V_e}{\lambda} \cdot cos\theta$$

$f_{doppler}$ is the doppler frequency, λ is the wavelength, V_e is the velocity, θ is the arrival angle. This formula is similar to *Formula 5*.

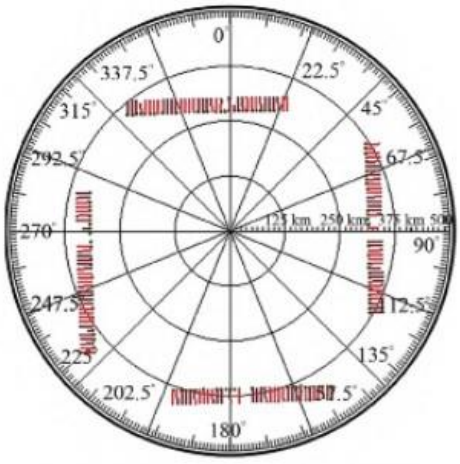


Figure 13: The Appearance of Radar

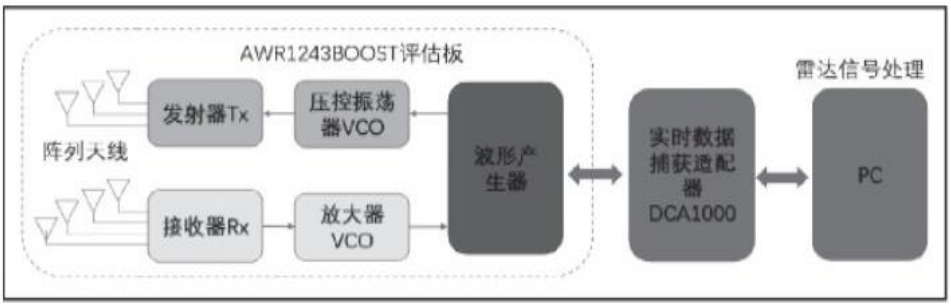


Figure 14: A Simple Model for the System

Experience

Contributors

Introduction – Liu Qianyi;

Objective, Methodology – Wang Zhuoyang;

Task 1 – Feng Yanjie;

Task 2 – Wan Peng, Liu Qianyi, Wang Zhuoyang;

Task 3 – Wang Zhuoyang;

Extension – Wan Peng;

Experience – Liu Qianyi;

Slides – Feng Yanjie, Wang Zhuoyang.

Experience
In this project, we learned about the working principle of passive radar, especially for non-electronic information major students, which greatly enriched our knowledge. In addition, the core of project is the processing of radar signals and understand the meaning of data is the core. The sampling, filtering and frequency conversion are all mentioned previously by teacher in class. The work of project makes us more skilled in this. One of the difficulties in this project is the understanding of the fuzzy function, and its ultimate purpose is to amplify the desired signal. We are also the first time to contact drawing heatmaps, which can allow us to represent our data more vividly in the future.

Score	99

Appendix

Programs

main.m

```
%% Passive Radar Simulation
% clear;
addpath("data");
coder.gpu.kernelfun;

tfMap = true;
heatMap = true;
resetCorDt = true;
resaveCorDt = true;
useBandLow = true;

dataIndexSet = 1 : 20;      % 数据集
order_lpf = 20;             % 低通滤波器阶数
order_rmv = 3;              % 归一化后降噪的幂

if useBandLow
    f_ddc = -3e6;           % 中心频率
    f_w = 9e6;              % 带宽
else
    f_ddc = 9.5e6;
    f_w = 2e6;
end

c_Set = 0 : 6;              % 时延采样间隔点
d_Set = -40 : 2 : 40;       % 频移采样间隔点值
c_idxSet = 1 : length(c_Set) - 1; % 时延采样左值点索引
d_idxSet = 1 : length(d_Set) - 1; % 频移采样左值点索引
```

```
t0_s = 5;                    % 时间-频移热图单个数据集内时间轴上的时间采样个数

if resetCorDt
    corDt = gpuArray(zeros(t0_s * dataIndexSet(end), c_idxSet(end),
d_idxSet(end)));
end

for dataIndex = dataIndexSet
    % 读取当前数据子集
    load("data_" + num2str(dataIndex) + ".mat");
    disp("Data_" + num2str(dataIndex));

    % 基本量
    dt = 1 / f_s;     % 原信号采样间隔
    lambda = 3e8 / f_c;     % 中心波长

    r_Set = c_Set / f_s * 3e8;  % 路程差
    c_mSet = c_Set(1 : end - 1) / f_s;              % 时延采样左值点
    d_mSet = d_Set(1 : end - 1);                    % 频移采样左值点

    t0_vOffset = cur_time;
    t0_idxOffset = (dataIndex - 1) * t0_s;

    t0_vSet = linspace(0, duration, t0_s + 1);
    t0_vSet = t0_vSet(1 : end - 1) + t0_vOffset;     % 时间-频移热图的绝对时间
采样点
    t0_idxSet = (1 : t0_s) + t0_idxOffset;          % 时间-频移热图的绝对时间采
样点索引

    l_ref = length(seq_ref);    % 参考信号采样点个数
    l_sur = length(seq_sur);    % 监视信号采样点个数
    t_ref = linspace(0, duration - dt, l_ref);  % 参考信号时域向量
    t_sur = linspace(0, duration - dt, l_sur);  % 监视信号时域向量

    timeToPoint = @(T) floor(T / dt + 1);

    % 数字下变频
    disp("  DDC.");
    fac_ref = exp(-1j * 2 * pi * f_ddc * t_ref);
    fac_sur = exp(-1j * 2 * pi * f_ddc * t_sur);
    ref_ddc = seq_ref .* fac_ref;
    sur_ddc = seq_sur .* fac_sur;
```

```
    % 低通滤波
    disp("  LPF.");
    [b_lpf, a_lpf] = butter(order_lpf, f_w / (f_s / 2));
    ref_lpf = filter(b_lpf, a_lpf, ref_ddc);
    sur_lpf = filter(b_lpf, a_lpf, sur_ddc);

    ref = gpuArray([ref_lpf, zeros(1, c_Set(end) + 1)]);
    sur = gpuArray([sur_lpf, zeros(1, c_Set(end) + 1)]);

    % 时间轴分段模糊函数值表
    if heatMap
        disp("  AMBIGUITY.");
        hmap = gpuArray(zeros(c_idxSet(end) + 1, d_idxSet(end) + 1));

        for t0_idx = t0_idxSet                  % 绝对时间轴采样点索引
            t0 = t0_vSet(t0_idx - t0_idxOffset); % 绝对时间轴采样点
            disp("    t0 = " + num2str(t0));
            for c_idx = c_idxSet                % 时延采样点索引
                disp("      c_idx = " + num2str(c_idx));
                for d_idx = d_idxSet            % 频移采样点索引
                    c = c_mSet(c_idx);
                    d = d_mSet(d_idx);
                    % disp("      c_idx = " + num2str(c_idx) + ", d_idx = " +
num2str(d_idx));

                    T = duration / t0_s;
                    N1 = timeToPoint(t0 - t0_vOffset);
                    N2 = timeToPoint(t0 - t0_vOffset + T);
                    C = floor(timeToPoint(c)) - 1;
                    fac = gpuArray(exp(-1j * 2 * pi * d * linspace(t0, t0 + T,
N2 - N1 + 1)));
                    y_ref = gpuArray(ref(N1 : N2));
                    y_sur = gpuArray(sur((N1 : N2) + C));
                    y = y_sur .* conj(y_ref) .* fac;

                    % 积分
                    y = (y + circshift(y, 1)) / 2;
                    y = y(2 : end);
                    corDt(t0_idx, c_idx, d_idx) = sum(y) * dt;

                    % 当前数据子集下的完整模糊函数值表
                    hmap(c_idx, d_idx) = hmap(c_idx, d_idx) + corDt(t0_idx, c_idx,
d_idx);
                end
```

```
            end
        end
    end

    % 绘图与持久化存储
    if tfMap
        spectrumFigure = figure(1);
        set(gcf, 'position', [0, 0, 1000, 900]);
        subplot(6, 2, 1);
        plotTimeSpectrum_ms_Re(seq_ref, f_s, "Reference Signal (Raw)");
        subplot(6, 2, 2);
        plotFrequencySpectrum_MHz_dB(seq_ref, f_s, "Frequency Spectrum
(Raw)");
        subplot(6, 2, 7);
        plotTimeSpectrum_ms_Re(seq_sur, f_s, "Surveillance Signal (Raw)");
        subplot(6, 2, 8);
        plotFrequencySpectrum_MHz_dB(seq_sur, f_s, "Frequency Spectrum
(Raw)");
        subplot(6, 2, 3);
        plotTimeSpectrum_ms_Re(ref_ddc, f_s, "Reference Signal (After DDC)");
        subplot(6, 2, 4);
        plotFrequencySpectrum_MHz_dB(ref_ddc, f_s, "Frequency Spectrum (After
DDC)");
        subplot(6, 2, 9);
        plotTimeSpectrum_ms_Re(sur_ddc, f_s, "Surveillance Signal (After
DDC)");
        subplot(6, 2, 10);
        plotFrequencySpectrum_MHz_dB(sur_ddc, f_s, "Frequency Spectrum (After
DDC)");
        subplot(6, 2, 5);
        plotTimeSpectrum_ms_Re(ref_lpf, f_s, "Reference Signal (After LPF)");
        subplot(6, 2, 6);
        plotFrequencySpectrum_MHz_dB(ref_lpf, f_s, "Frequency Spectrum (After
LPF)");
        subplot(6, 2, 11);
        plotTimeSpectrum_ms_Re(sur_lpf, f_s, "Surveillance Signal (After
LPF)");
        subplot(6, 2, 12);
        plotFrequencySpectrum_MHz_dB(sur_lpf, f_s, "Frequency Spectrum (After
LPF)");
        print(spectrumFigure, "output/SpectrumFigure_Data_" +
num2str(dataIndex) + "_r300.png", "-dpng", "-r300");
    end
```

```
    if heatMap
        save("rec/hmap_from_" + num2str(cur_time) + "s.mat", "hmap");       %
持久化存储 hmap，便于修正结果图
        heatmapFigure = figure(2);
        [meshgrid_Doppler, meshgrid_Range] = meshgrid(d_Set, r_Set);
        hmap = abs(hmap);
        hmap_max = max(max(hmap));
        surf(meshgrid_Doppler, meshgrid_Range, (hmap / hmap_max) .^ order_rmv);
        view(0, 90);
        colorbar;
        xlim([d_Set(1), d_Set(end)]), xlabel("Doppler Frequency (Hz)");
        ylim([r_Set(1), r_Set(end)]), ylabel("Range (m)");
        set(gca, 'YTick', r_Set);
        set(gca, 'YTicklabel', num2str(r_Set'));
        [R_idx, D_idx] = find(hmap == hmap_max);
        R = r_Set(R_idx);
        D = d_Set(D_idx);
        title("Range-Doppler Spectrum [ " + num2str(t0_vSet(1)) + "s: " +
num2str(R) + "m, " + num2str(D) + "Hz ]");
        print(heatmapFigure, "output/HeatmapFigure_Data_" + num2str(dataIndex)
+ "_r300.png", "-dpng", "-r300");
    end
end

if resaveCorDt
    save("rec/corDt.mat", "corDt");
end

%% Functions
function plotTimeSpectrum_ms_Re(s, fs, tit)
    l_s = length(s);
    t_s = l_s / fs;
    p_t = linspace(0, t_s, l_s) * 1e3;
    plot(p_t, real(s)), xlim([0, 0.01] * 1e3);
    title(tit);
    xlabel("Time (ms)"), ylabel("Re(Waveform)");
end

function plotFrequencySpectrum_MHz_dB(s, fs, tit)
    l_s = length(s);
    t_s = l_s / fs;
    f_bound = (l_s - 1) / t_s / 2;
    p_f = linspace(-f_bound, f_bound, l_s) / 1e6;
    p_m = 20 * log10(abs(fftshift(fft(s))));
```

```
    plot(p_f, p_m), xlim([-f_bound, f_bound] / 1e6), ylim([-70, 30]);
    title(tit);
    xlabel("Frequency (MHz)"), ylabel("Magnitude (dB)");
end
```

timeFreqFigure.m

```
%% Calculating
L_corDt = 100;
N = L_corDt - t0_s + 1;
hmap_res = gpuArray(zeros(N + 1, d_idxSet(end) + 1));
t_Set = (0 : N) * duration / t0_s;

for T0 = 1 : N
    hmap_tmp = gpuArray(zeros(c_idxSet(end) + 1, d_idxSet(end) + 1));
    for I = 0 : (t0_s - 1)
        hmap_tmp(:, :) = hmap_tmp(:, :) + [squeeze(corDt(T0 + I, :, :)),
zeros(c_idxSet(end), 1); zeros(1, d_idxSet(end)), 0];
    end
    hmap_tmp = abs(hmap_tmp);
    hmap_tmp_max = max(max(hmap_tmp));
    [R_tmp_idx, D_tmp_idx] = find(hmap_tmp == hmap_tmp_max);
    hmap_tmp = (hmap_tmp / hmap_tmp_max) .^ order_rmv;
    hmap_res(T0, :) = hmap_tmp(R_tmp_idx, :);
    cd_res(T0, 1) = r_Set(R_tmp_idx) + 6;
    cd_res(T0, 2) = d_Set(D_tmp_idx) + 1;
end

%% Plotting
tfHeatmapFigure = figure(3);
set(gcf, "Position", [0 0 600 800]);
[meshgrid_Doppler, meshgrid_Time] = meshgrid(d_Set, t_Set);
surf(meshgrid_Doppler, meshgrid_Time, hmap_res);
view(0, 90);
colorbar;
xlim([d_Set(1), d_Set(end)]), xlabel("Doppler Frequency (Hz)");
ylim([t_Set(1), t_Set(end)]), ylabel("Time (s)");
y_tickSet = linspace(0, (dataIndex(end) - 1) * duration, dataIndex(end));
set(gca, 'YTick', y_tickSet);
set(gca, 'YTicklabel', num2str(y_tickSet'));
title("Time-Doppler Spectrum");
print(tfHeatmapFigure, "output/tfHeatmapFigure_r300.png", "-dpng", "-r300");
```

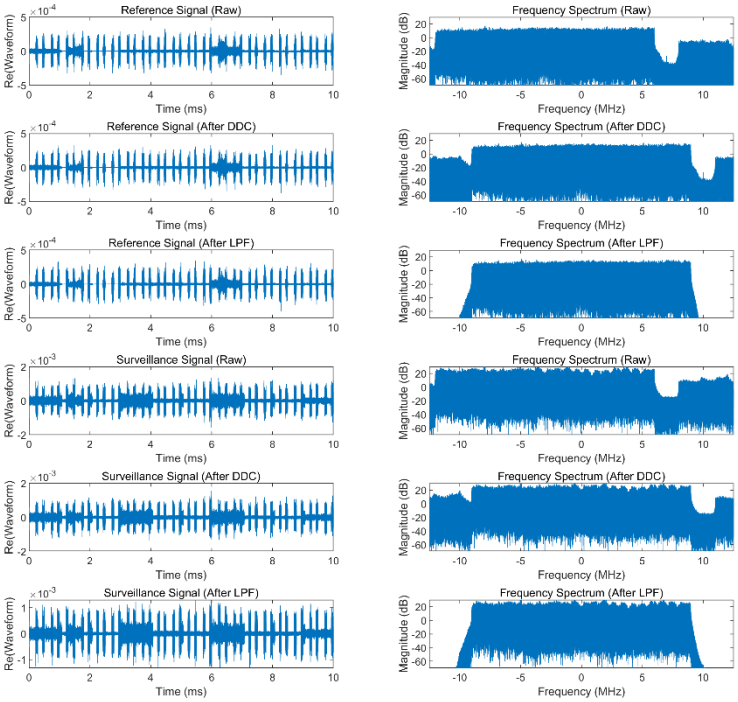
motionAnalysis.m

```
%% Calculating
```

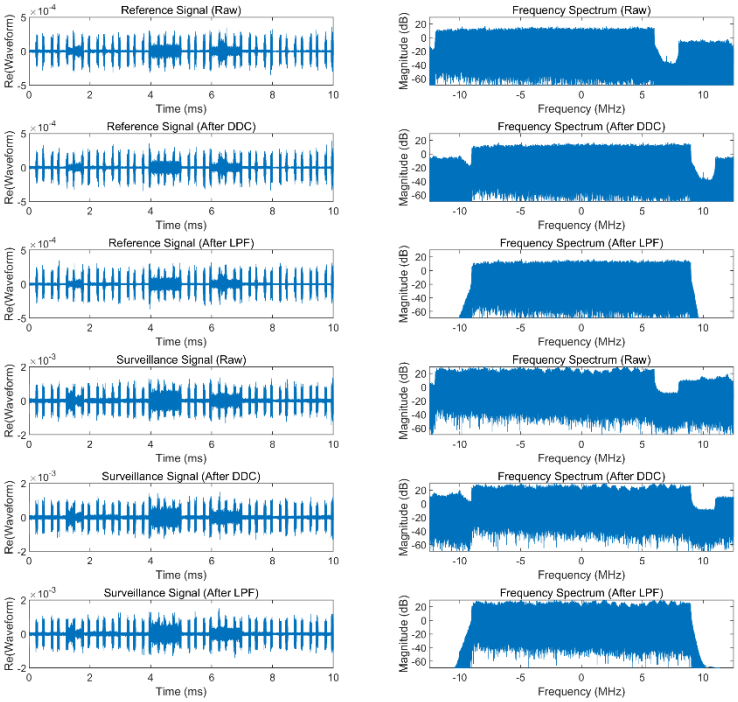
```
L = 230;
T = duration / t0_s;
REP = 10000;

c_t = cd_res(:, 1)';
d_t = cd_res(:, 2)';
x_t = c_t / 2 .* (c_t + 2 * L) ./ (c_t + L);

for K = 1 : REP
    b_t = x_t ./ sqrt(L * L + x_t);
    v_t = - lambda .* d_t ./ (sqrt((b_t + 1) ./ 2) .* 2);

    X_t = (v_t + circshift(v_t, 1)) / 2 * T;
    x_t(1) = 0;
    for J = 2 : length(v_t)
        x_t(J) = sum(X_t(2 : J));
    end
    x_t = x_t + 12;
end

%% Plotting - Remember to reset cd_res before timeFreqFigure!
vtFigure = figure(4);
plot(t_Set(1 : end - 1), x_t, "LineWidth", 1), hold on;
plot(t_Set(1 : end - 1), v_t, "LineWidth", 1);
grid on;
xlabel("Time (s)"), ylabel("Distance (m) & Velocity (m/s)");
xlim([t_Set(1) t_Set(end - 1)]), ylim([-6, 16]);
legend("x(t)", "v(t)");
print(vtFigure, "output/vtFigure_r300.png", "-dpng", "-r300");
```

All the Figures for the Results

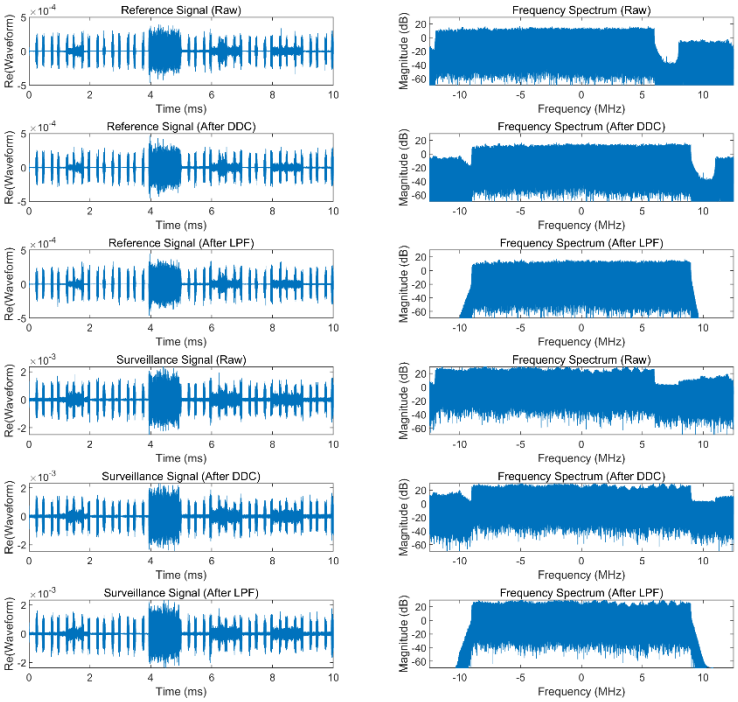
Task 1. Time and Frequency Spectrums



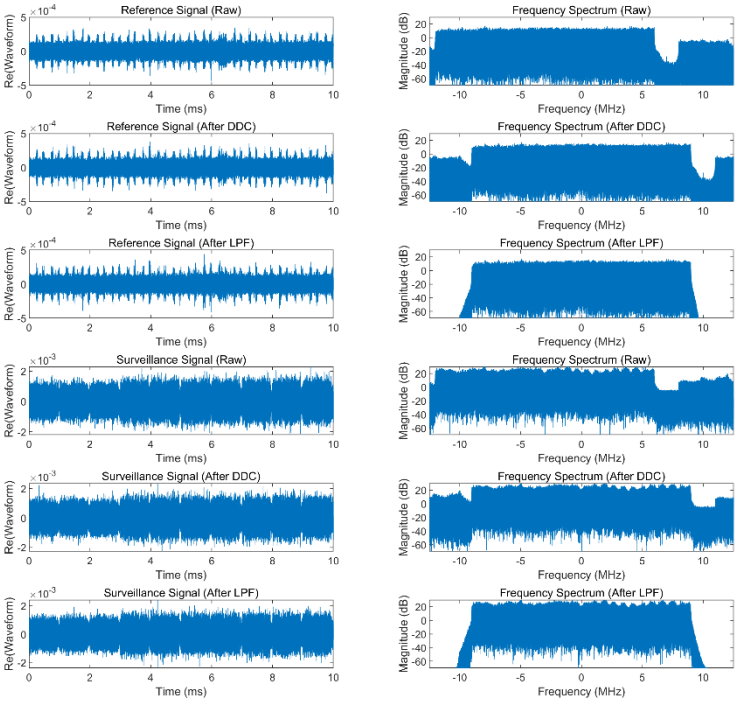
Data 01



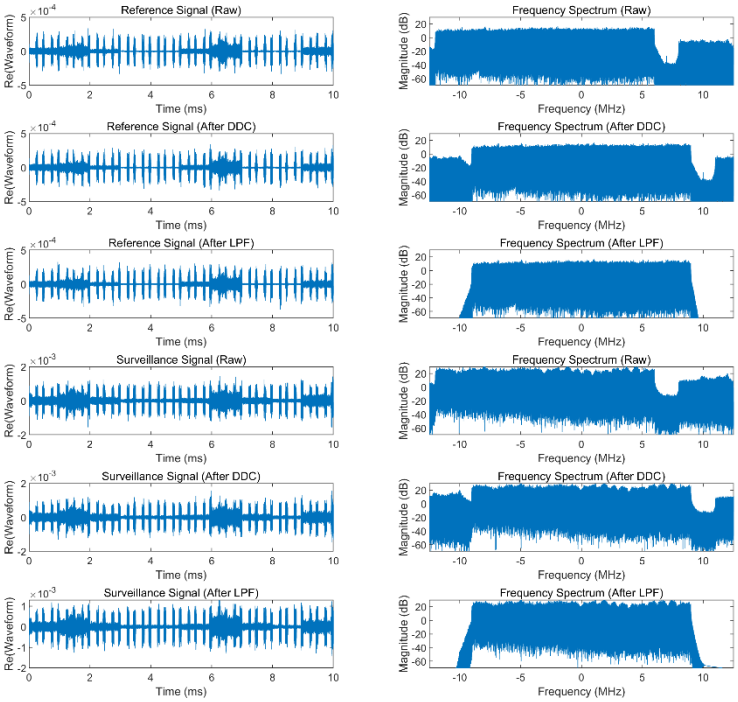
Data 02



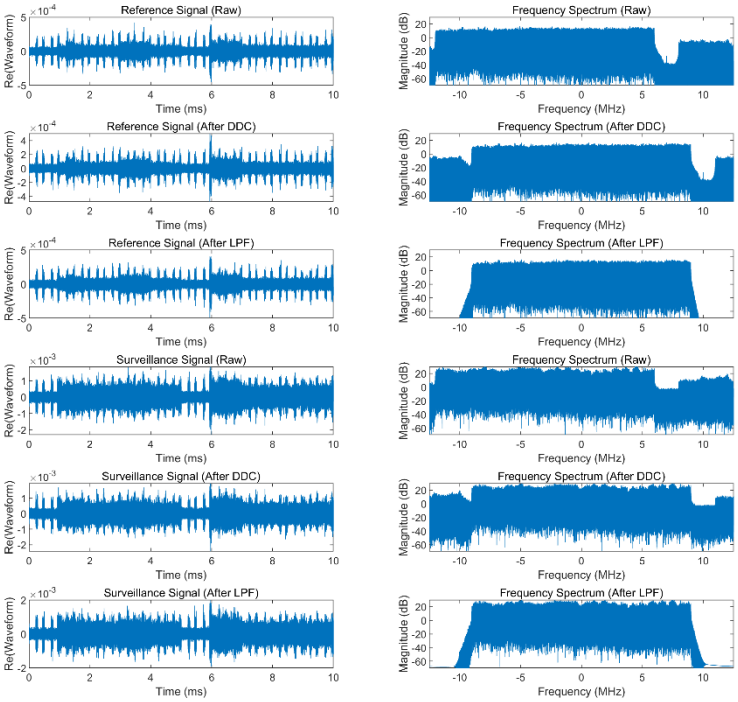
Data 03



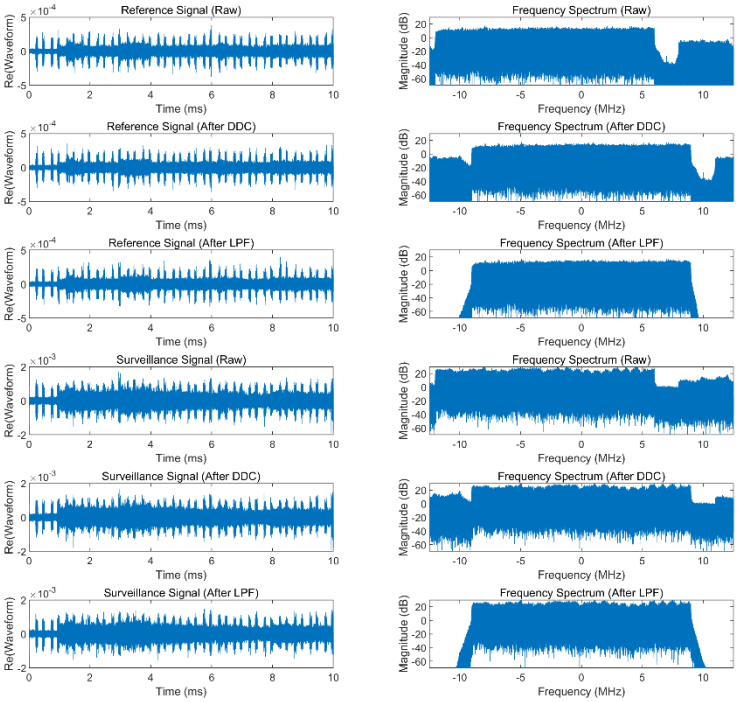
Data 04



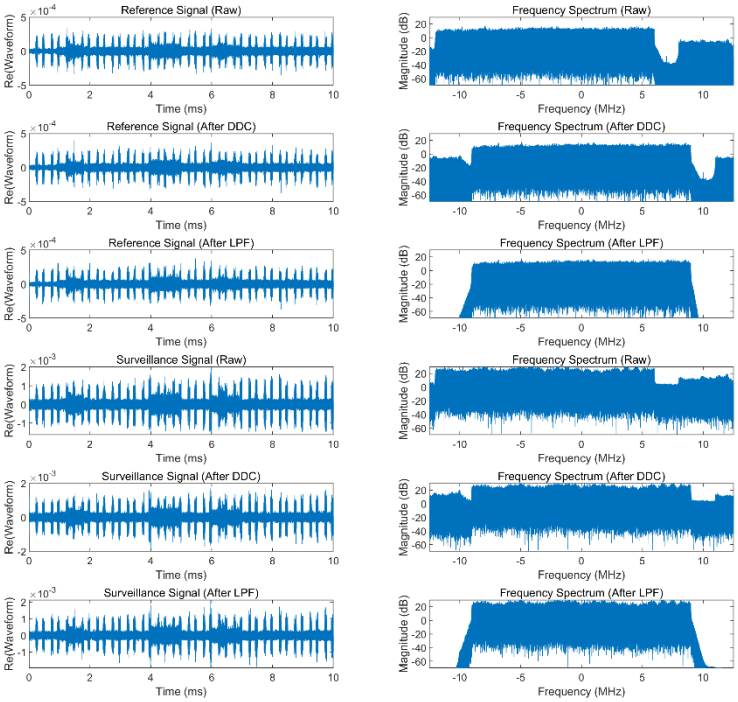
Data 05



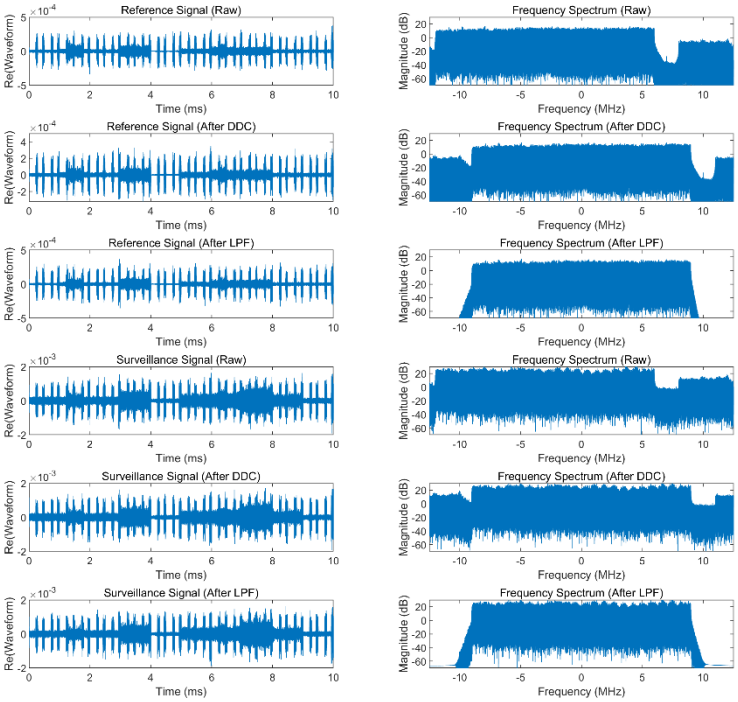
Data 06



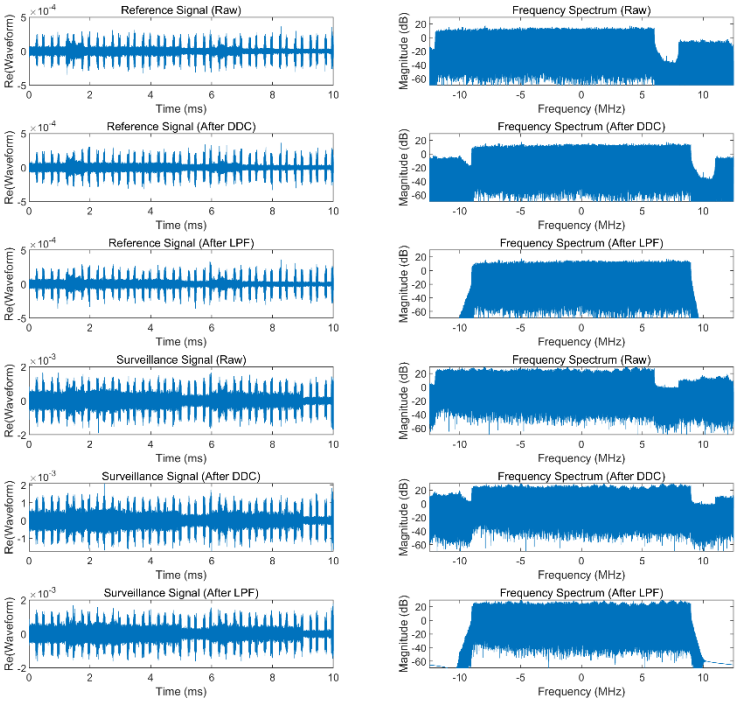
Data 07



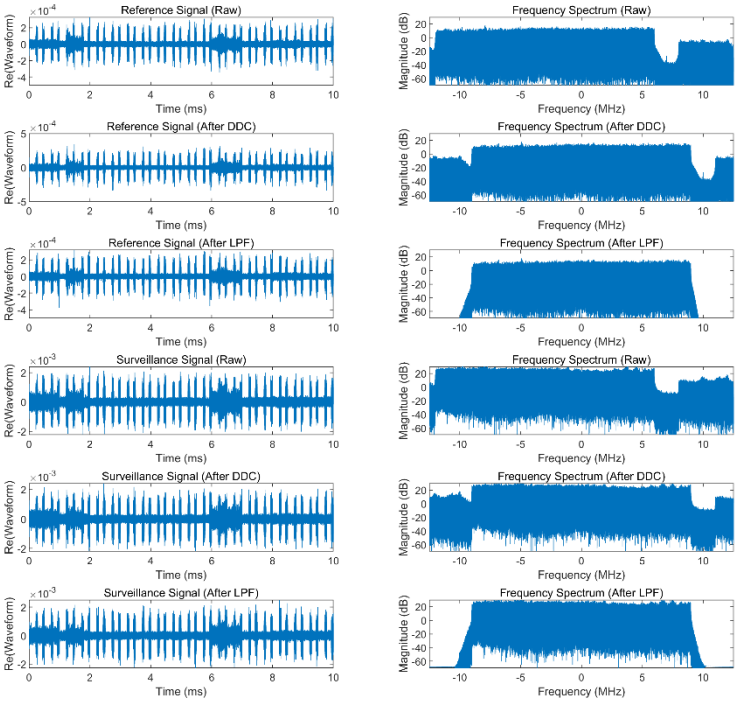
Data 08



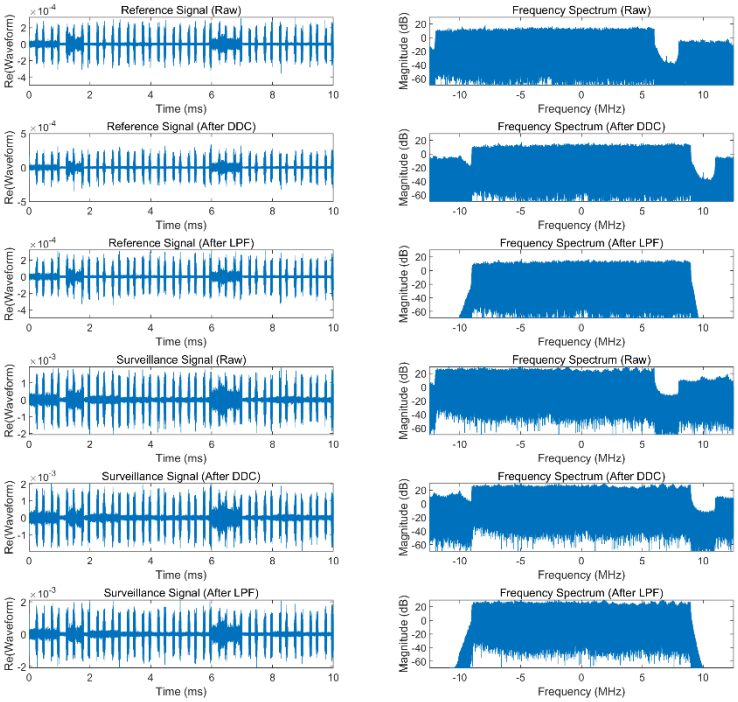
Data 09



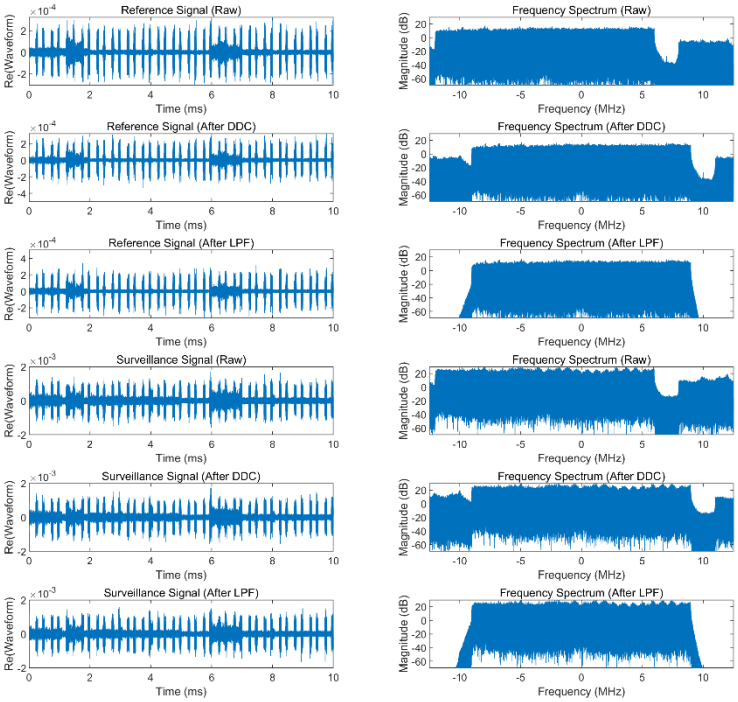
Data 10



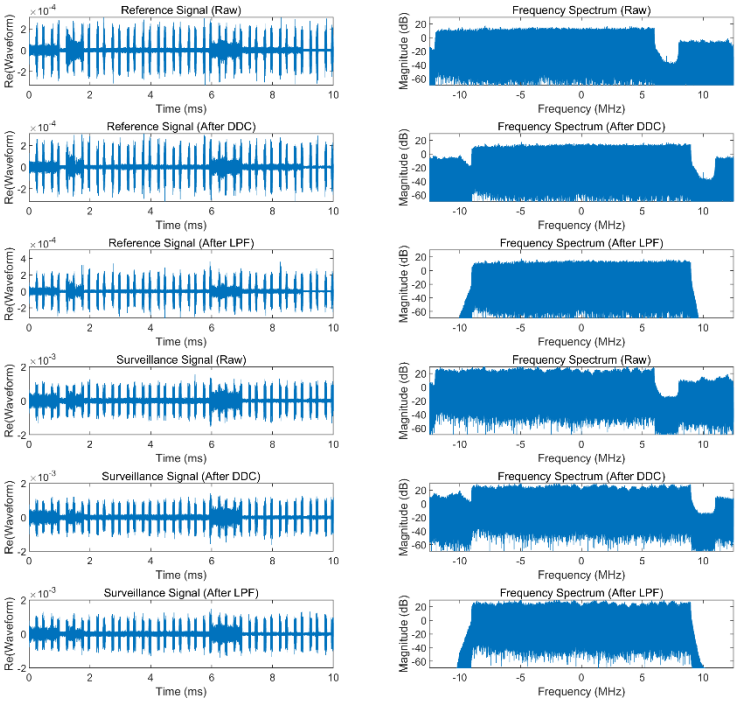
Data 11



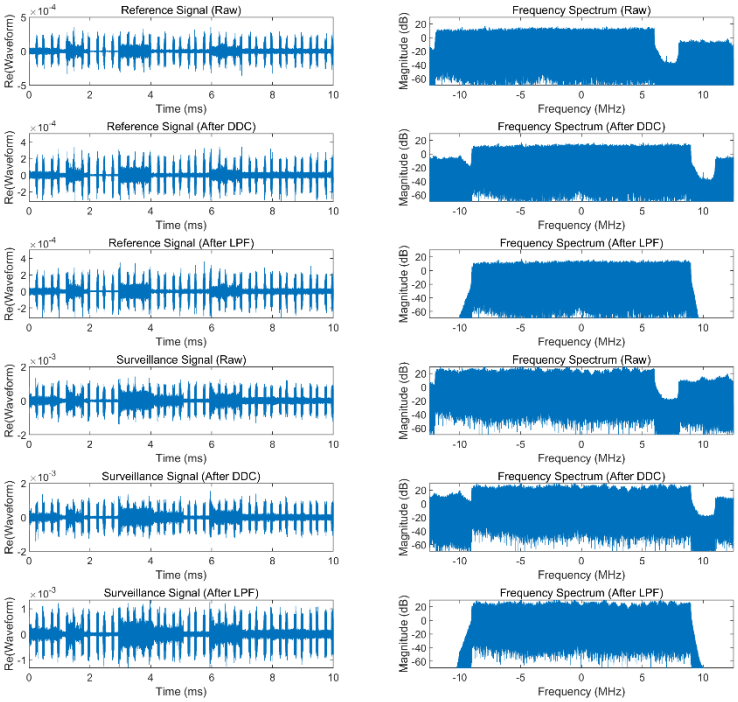
Data 12



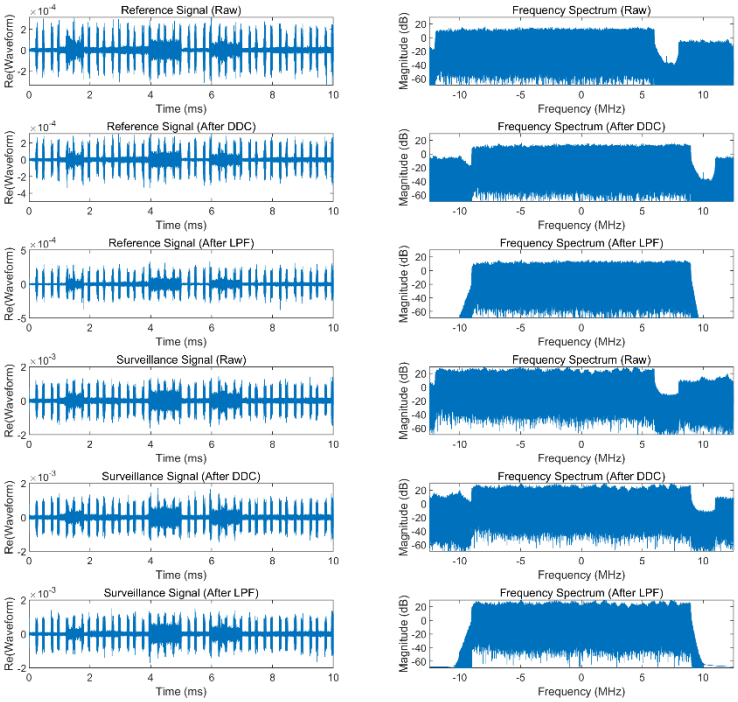
Data 13



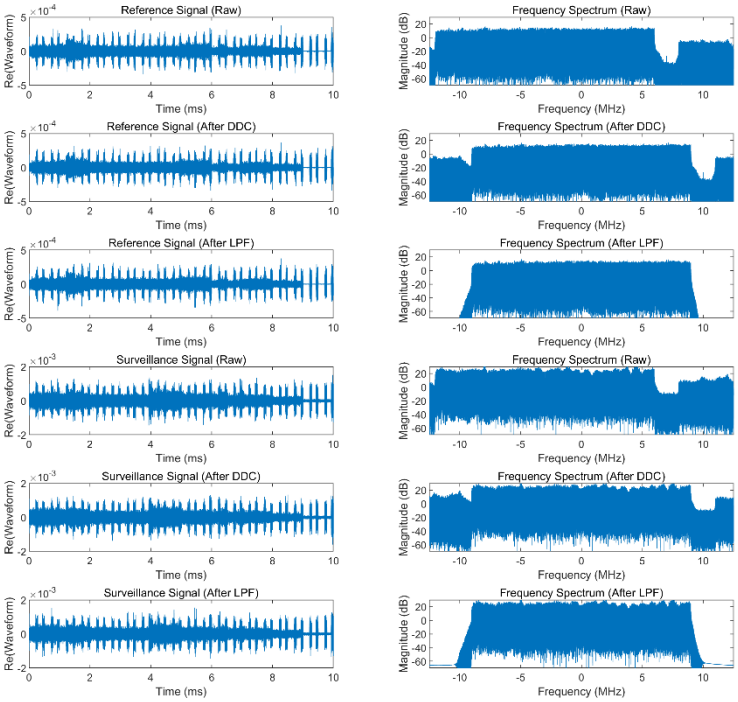
Data 14



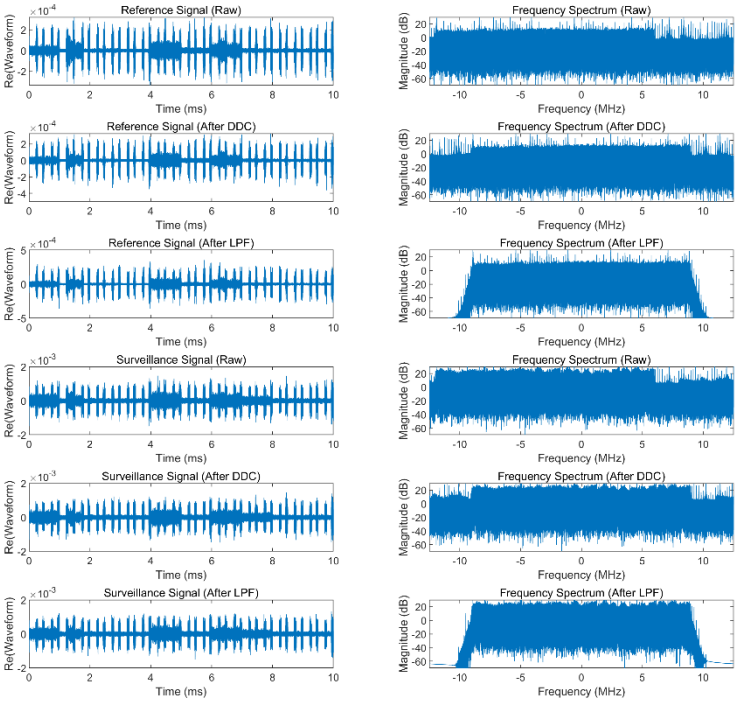
Data 15



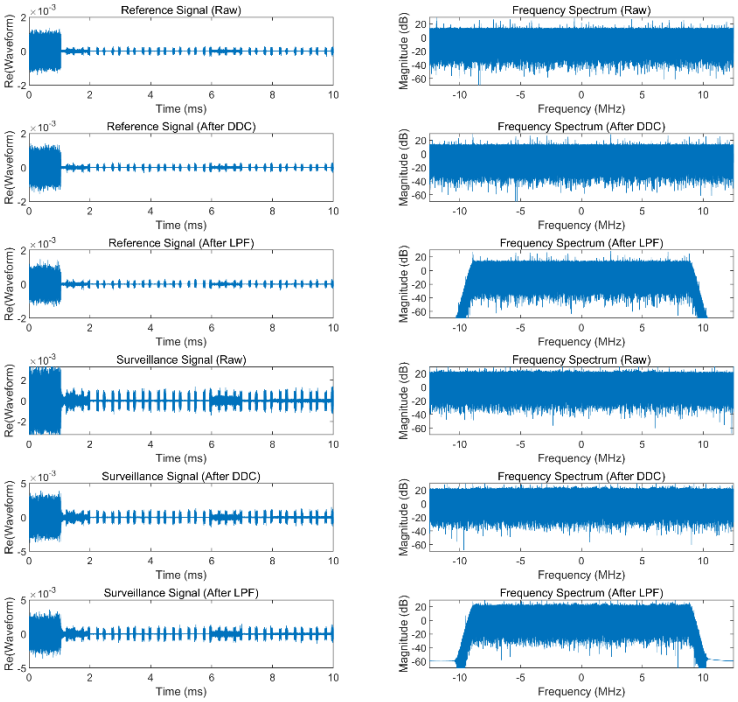
Data 16



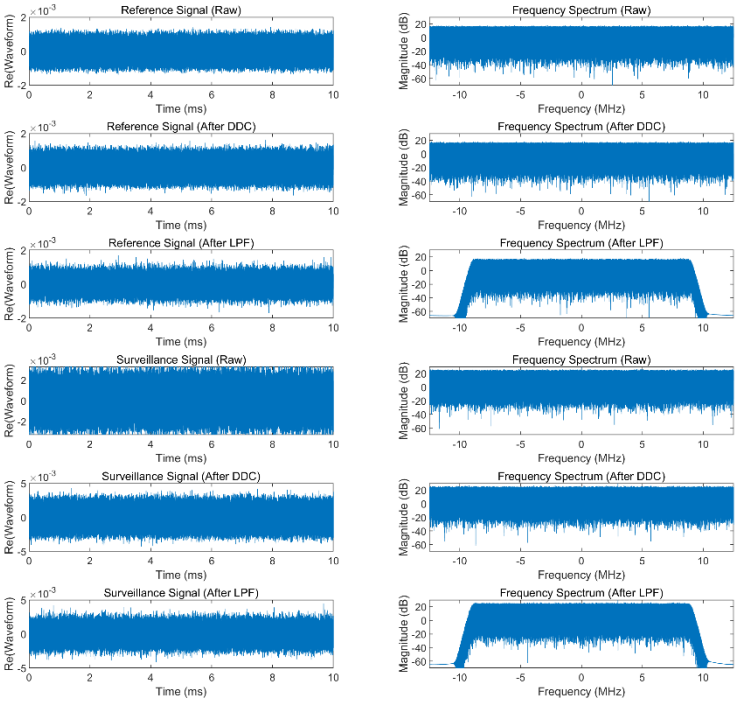
Data 17



Data 18

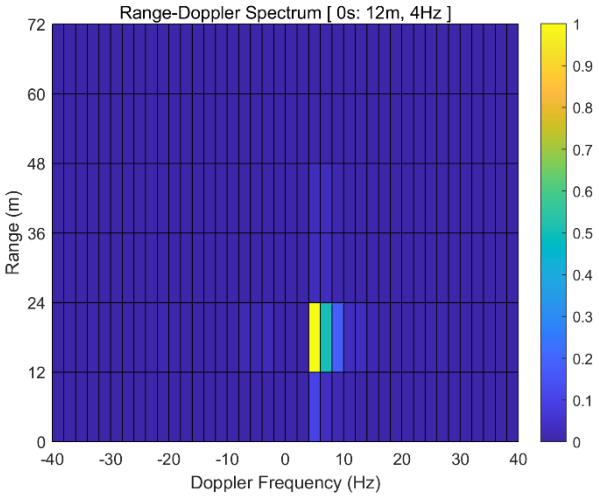


Data 19

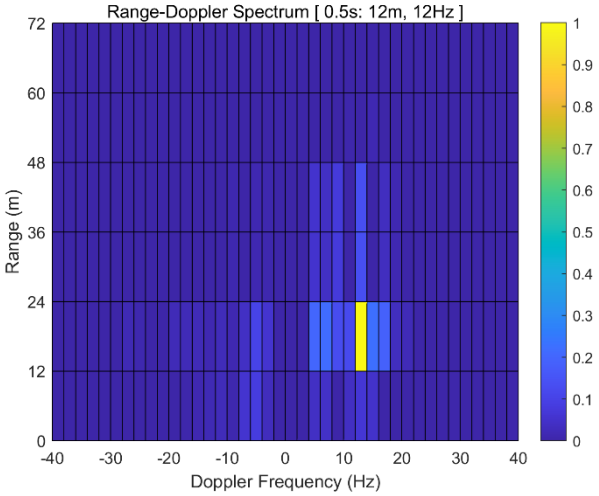


Data 20

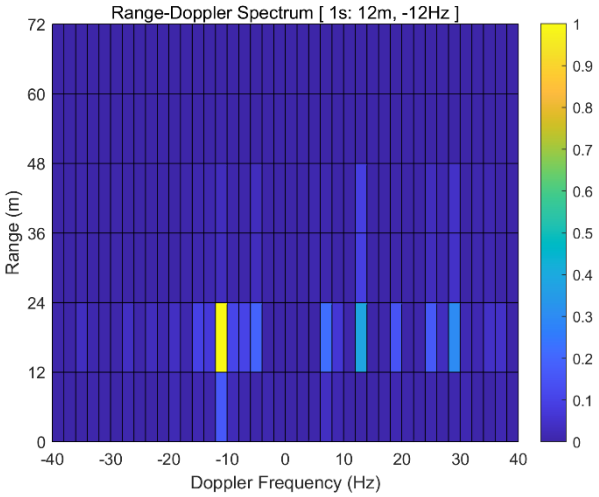
Task 2. Heatmaps for Ambiguity Function



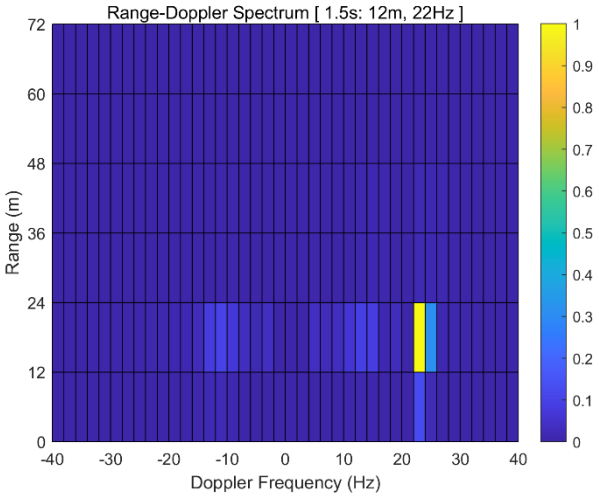
Data 01



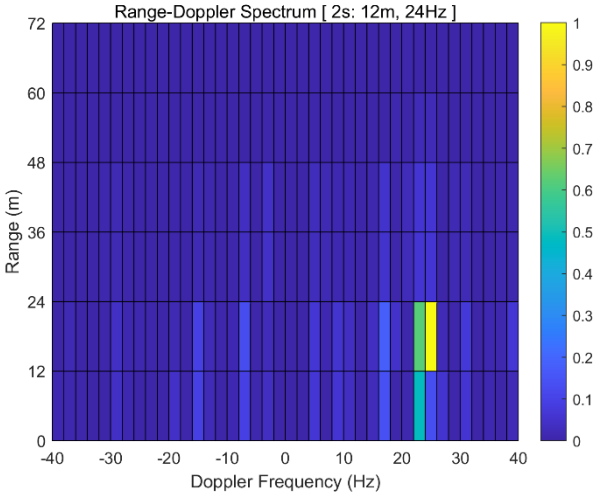
Data 02



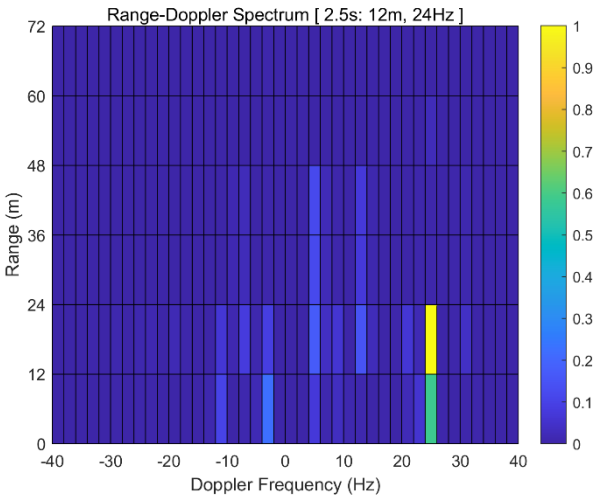
Data 03



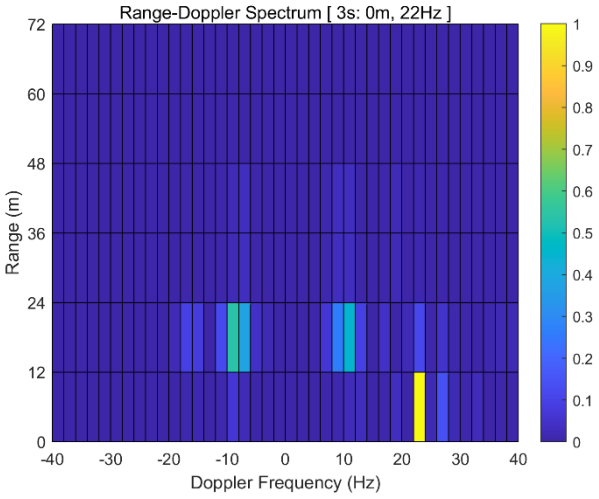
Data 04



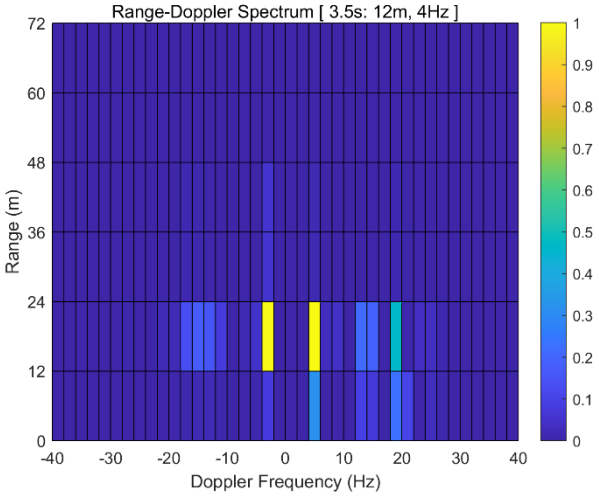
Data 05



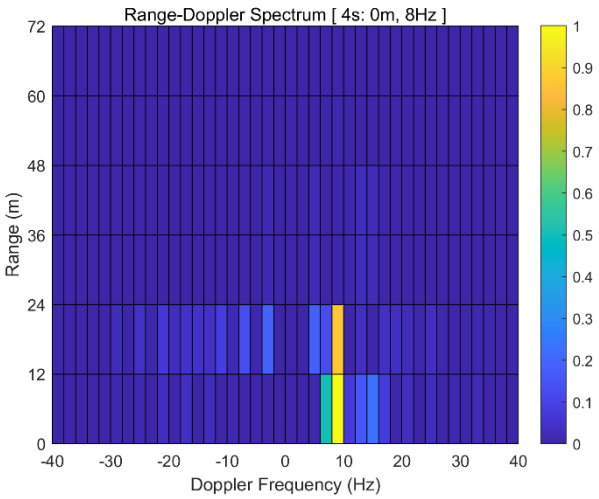
Data 06



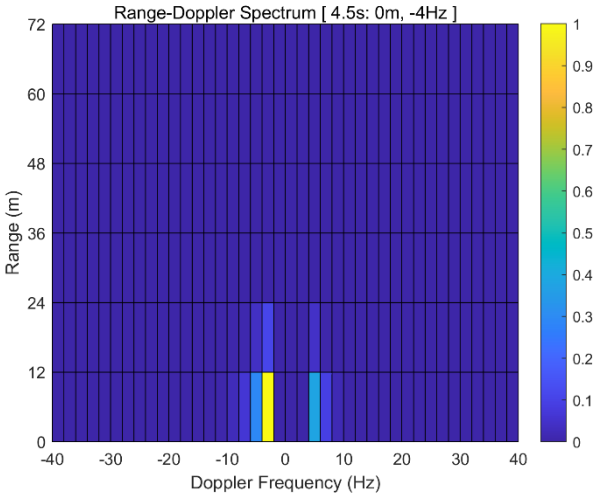
Data 07



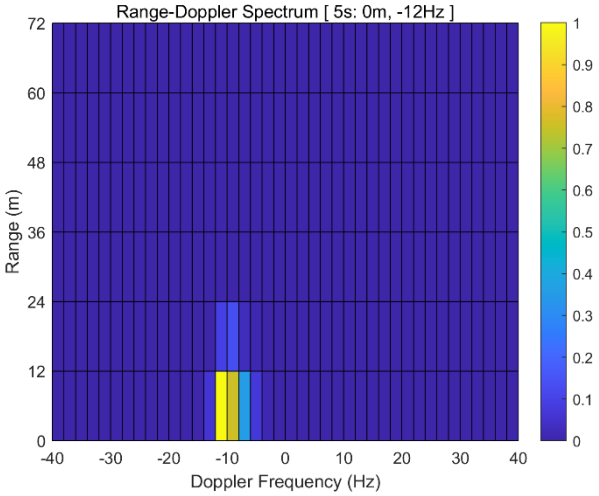
Data 08



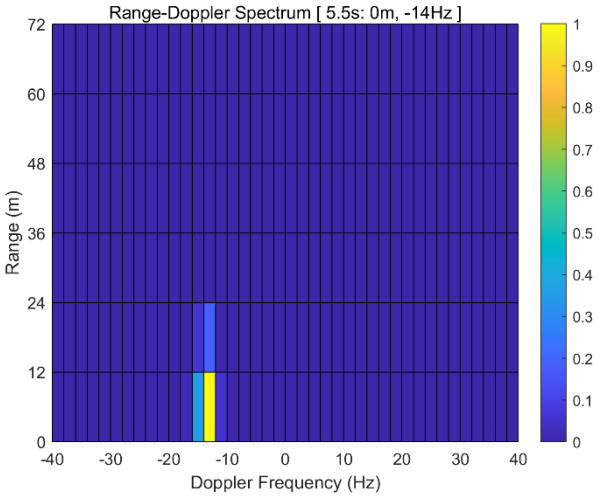
Data 09



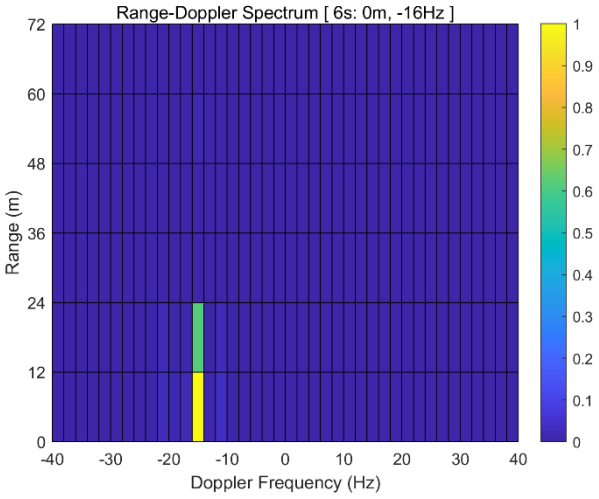
Data 10



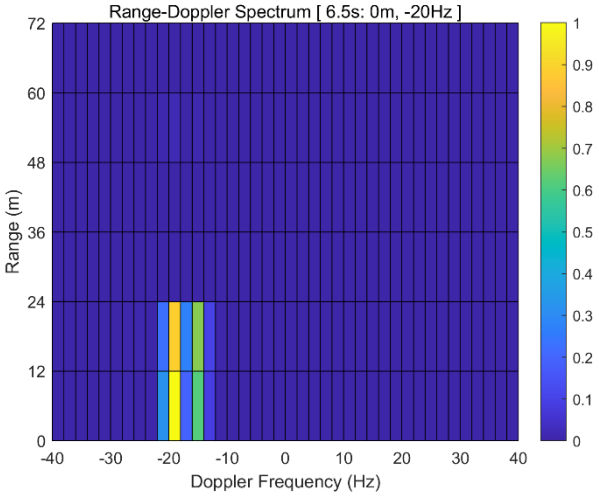
Data 11



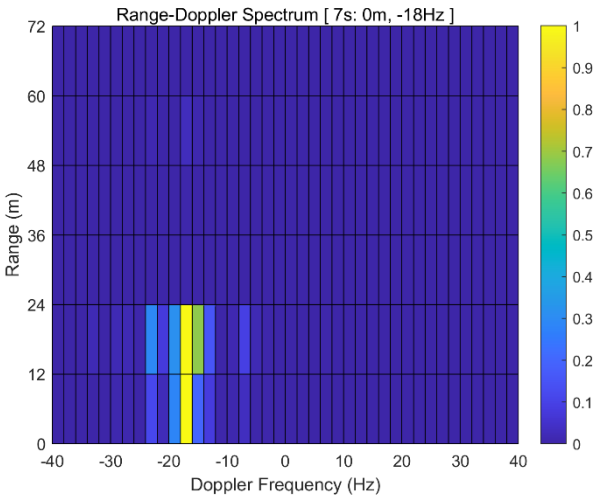
Data 12



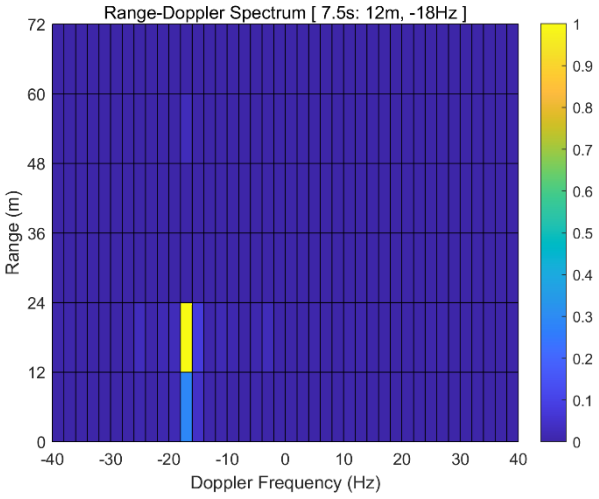
Data 13



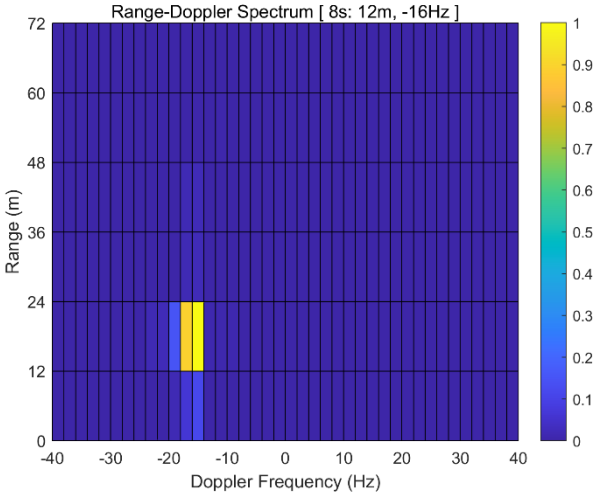
Data 14



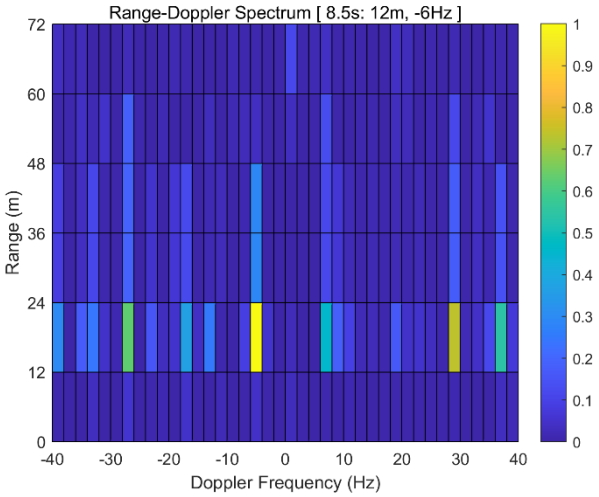
Data 15



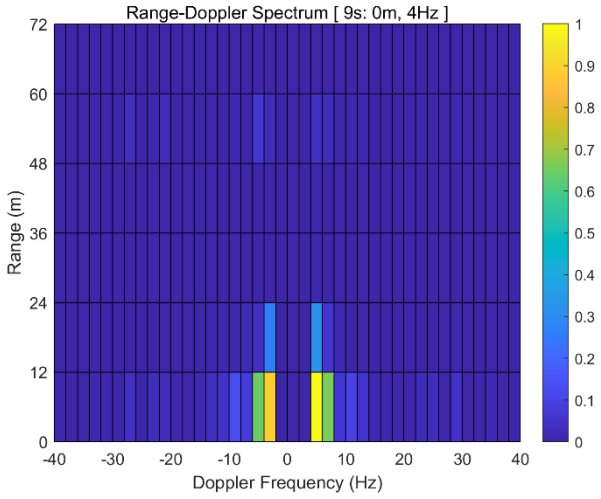
Data 16



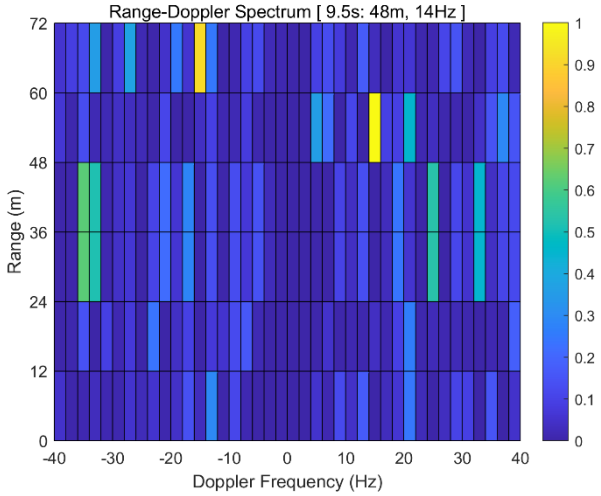
Data 17



Data 18

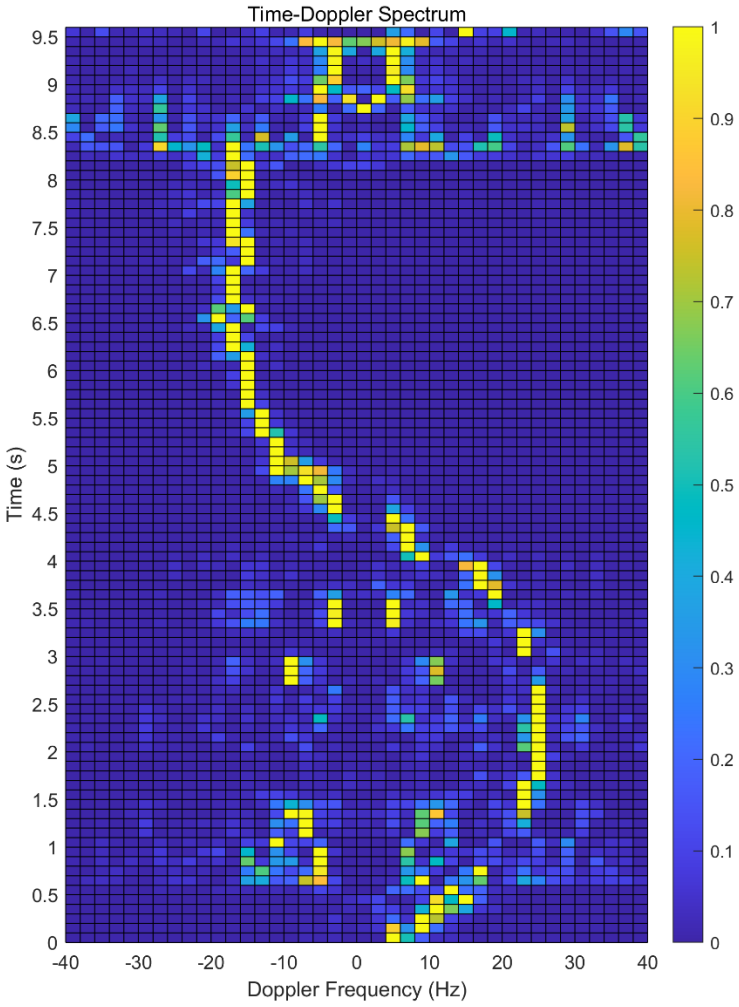


Data 19

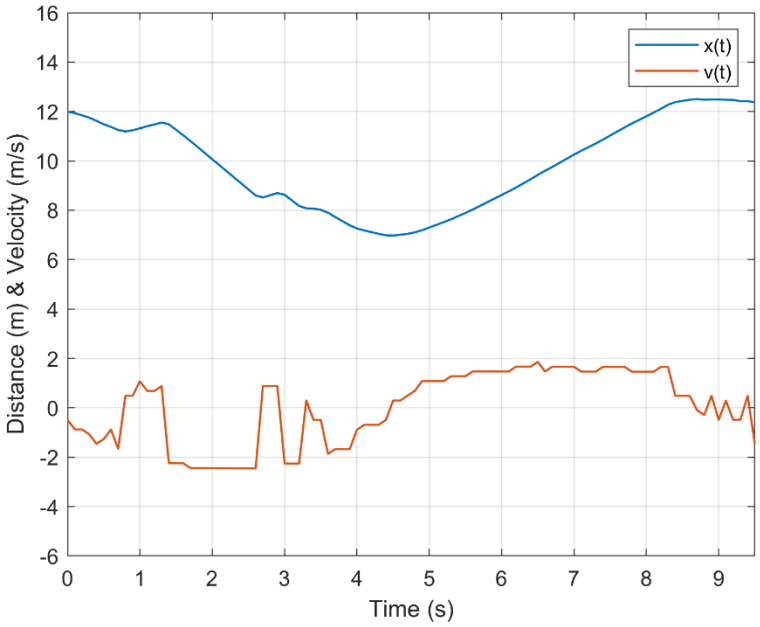


Data 20

Task 3. Doppler Frequency – Time Relation



Extra. Velocity and Displacement – Time Relation


x(t)
v(t)
Distance (m) & Velocity (m/s)
Time (s)
16
14
12
10
8
6
4
2
0
-2
-4
-6
0
1
2
3
4
5
6
7
8
9