Project 2 Motion detection via communication signals

张怡程　马文骁　陈嘉祺　高进

April 14, 2024

1 Introduction

1.1 Principles of Motion Detection

Moving objects reflect signals, causing time delays due to differences in distance, and frequency shifts due to the Doppler Effect on the radiation signal. The signal that directly comes from the illuminator is known as the reference signal, while the reflected one is termed the surveillance signal. By analyzing these two signals, we can determine the time delay and frequency shift caused by the object, thereby deriving its motion. Assume the transmitted signal to be $x(t)$, the reference signal to be $y_{\text{ref}}(t)$ and surveillance signal to be $y_{\text{sur}}(t)$. We can get

$$y_{\text{ref}}(t) = \alpha x\left(t-\tau_1\right), \quad y_{\text{sur}}(t) = \beta x\left(t-\tau_2\right) e^{\mathrm{j}2\pi f_d t} \tag{1}$$

Where α and β stand for the attenuation factor due to transmission, τ stands for the time delay. Due to Doppler Effect

$$\tau_i = \frac{s_i}{c}, \quad f_d = \frac{v f_c}{c} \tag{2}$$

Where f_d is the Doppler frequency shift.

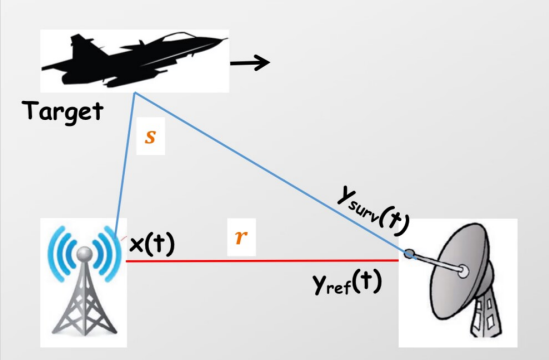


Figure 1: The basic principle of passive radar

1.2 Cross-Correlation (Ambiguity Function)

The ambiguity function is aimed at estimating the relation of two signals. Here is its definition:

$$\mathrm{Cor}(\tau, f_d) = \int_t^{t+T} y_{\mathrm{sur}}(t) y_{\mathrm{ref}}^*(t-\tau) e^{-\mathrm{j}2\pi \mathrm{f_d t}} \mathrm{dt} \tag{3}$$

where $y_{\mathrm{ref}}(t)$ represents the complex envelope of the signal received in the surveillance channel, $y_{\mathrm{sur}}(t)$ is a (cleaned) replica of the transmitted signal, τ denotes the potential Time Difference of Arrival (TDOA) of the target echo signal, and f_d indicates the potential corresponding bistatic Doppler. The maximum value of $\mathrm{Cor}(\tau, f)$, the corresponding variables of which are the kinematic information we get:

$$\left(\widehat{\tau}, \widehat{f_d}\right) = \arg\max_{\tau, f} \mathrm{Cor}(\tau, f_d) \tag{4}$$

1.3 Time Domain Processing of Signals

The signal is always transmitted at a higher frequency, necessitating the conversion of the target signal to a frequency centered around 0 Hz. This is followed by filtering out the high-frequency signal using a low-pass filter. The process of converting the signal to a frequency centered on 0 Hz is known as digital downconversion (DDC). Utilizing the properties of the Fourier transform, we can achieve the desired result through the application of the following formula.

$$y(t) = x(t) e^{-\mathrm{j}2\pi f_{\mathrm{ddc}} t} \tag{5}$$

where f_{ddc} is the central frequency of the target signal fragment. To filter out the high-frequency signal, we choose the 20-order Butterworth low-pass filter.

```
1 [b,a]=butter(20,bandwidth/(fs/2),'low');
2 x_filtered=filter(b,a,x);
```

Here, the LPF is verified. Draw the frequency response of the LPF system. From Figure2, it can be seen that the cutoff frequency of the LPF is approximately 9e6Hz, which is consistent with the requirements of practical operations.

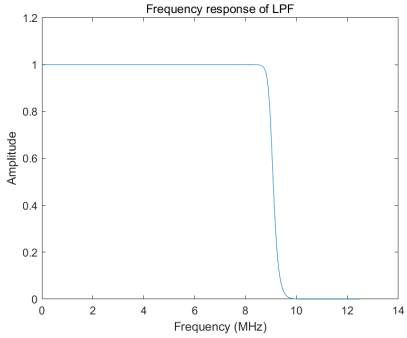


Figure 2: Frequency response of LPF

2 Result and Analysis

2.1 Task1

2.1.1 Restatement

Draw the time domain and frequency domain waveforms of the reference signal and the surveillance signal respectively in their original state, after undergoing DDC, and after passing through an LPF. Here, as an example, only the data from the first 0.01 seconds in data_1.mat is used for the illustration.

2.1.2 Analysis

First, read the file to obtain relevant information such as the reference signal, surveillance signal, and sampling frequency. Since there are two base stations, we will receive two segments of signals with bandwidths of 20MHz and 5MHz, respectively, which are 2110MHz∼2130MHz and 2130MHz∼2135MHz. Before processing the signals, we need to filter out the 2130MHz-2135MHz segment (5M bandwidth) and retain the 2110MHz∼2130MHz segment (20MHz). All signals obtained from the file reads have been preliminarily processed. By plotting the spectrum of the raw signal, we can see that f_ddc = -3e6 and bandwidth = 9e6.

Then we need to conduct DDC on the 2110MHz∼2130MHz signal so that the frequency is centered on 0Hz, and then plot the corresponding time-domain and frequency-domain graphs. The specific method for implementing DDC involves transforming the time-domain signal according to the formula

$$e^{j\omega_0 t}x(t) \longleftrightarrow X\left(j\left(\omega-\omega_0\right)\right) \tag{6}$$

where ω_0 = 3e6, followed by a time-frequency transformation using FFT.

Next, we use a low-pass filter with a bandwidth setting of 9e6. This allows us to retain the 2110MHz∼2130MHz signal while filtering out the 2130MHz∼2135MHz signal. Here, we set the order of the Butterworth filter to n=20 to minimize the bandwidth of the transition section.

Here, we take 'data1.mat' as an example to illustrate the time-domain and frequency-domain graphs of the reference signal and surveillance signal under different processing methods. The figures are shown in Figure3 and Figure4. The figures for the remaining data can be found in the appendix.

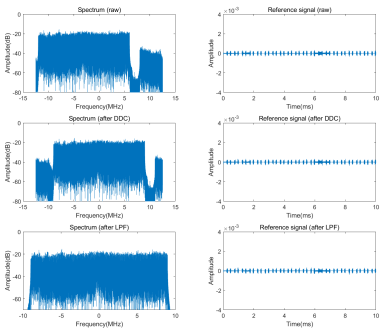


Figure 3: Reference Signal: 0∼0.5s

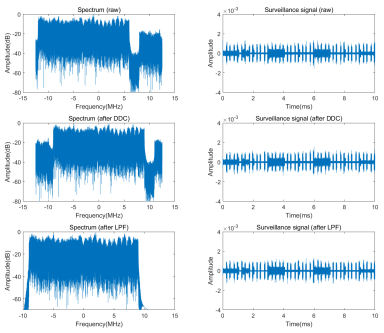


Figure 4: Surveillance Signal: 0∼0.5s

2.2 Task2

2.2.1 Restatement

After some trial, it was found that selecting a Doppler frequency range of -40Hz $\sim$ 40Hz and a distance range of 0 $\sim$ 72m between the signal transmitter and the moving object for traversal is quite appropriate. Based on this, draw the Range Doppler spectrum of the signal located at 0 $\sim$ 0.5s, 2 $\sim$ 2.5s, 5 $\sim$ 5.5s, and 7 $\sim$ 7.5s. The Range Doppler spectrum diagrams for the rest of the datasets are detailed in the appendix.

2.2.2 Analysis

Take the processed reference signal and surveillance signal from Task 1 and use them as inputs for the ambiguity function calculation. Iterate through different values of τ and f_D to compute the corresponding Cor(τ, f_D). Identify the values of τ and f_D that maximize Cor(τ, f_D), where the maximized tau corresponds to the time delay between the moving object and the transmission of the signal from the base station. This information can be used to qualitatively determine the relative position of the moving object with respect to the base station.

The value of f_D corresponds to the Doppler frequency. When $f_D > 0$, it indicates that the moving object is approaching the receiving tower, and when $f_D < 0$, it indicates that the moving object is moving away from the base station. This information can be used to determine the direction of motion of the object.

Based on the Figure5 generated from Task 2, it can be observed that during the time interval from 0 to 0.5 seconds, the approximate difference in distance

between the signal transmission end to the base station and the path from the moving object to the base station is about 12 meters. Additionally, the Doppler frequency is greater than 0 during this period, indicating that the moving object is approaching the base station. The analysis approach for the other three sets of Figure6, Figure7, Figure8 is the same.

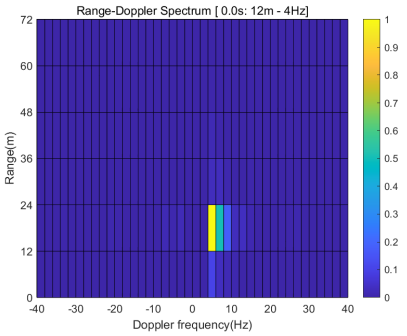


Figure 5: Reference Signal: 0∼0.5s

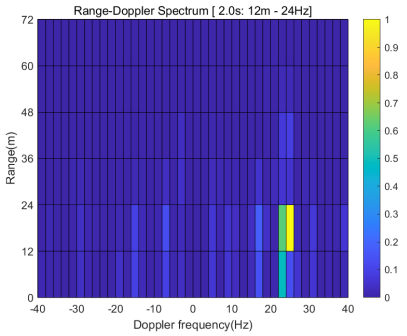


Figure 6: Surveillance Signal: 2.0∼2.5s

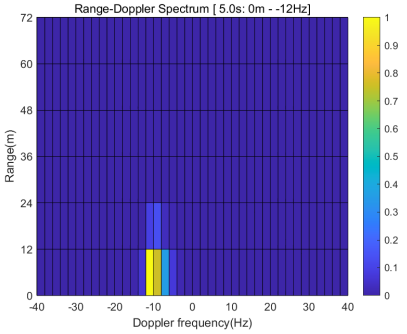


Figure 7: Reference Signal: 5∼5.5s

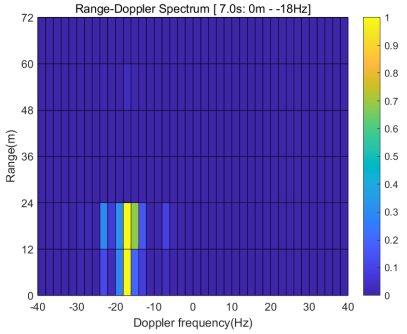


Figure 8: Surveillance Signal: 7∼7.5s

2.3 Task3

2.3.1 Restatement

Concatenate all datasets, and use a sliding window to select reference signals and surveillance signals from different time periods. Determine the Doppler frequency at the location of the maximum value of the Ambiguity Function. Draw the Time-Doppler spectrum of the signal.

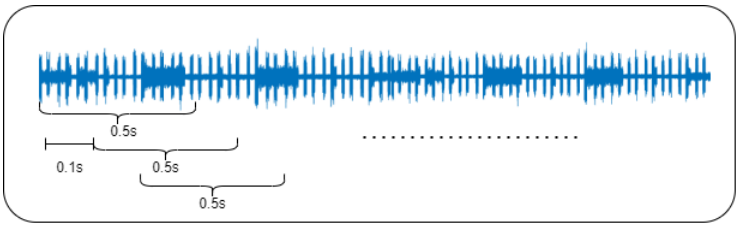


Figure 9: Method of signal segmentation (EX: 0.1s)

2.3.2 Analysis

Due to the fact that the motion of the target relative to the base station is in the same direction as the line connecting the two points, plus the unique configuration of the source, target, and reviewer, according to

$$f_D = \pm \frac{v}{\lambda_S} \tag{7}$$

it is evident that the magnitude of fD should be directly proportional to the velocity of the moving object, v. (Figure10)

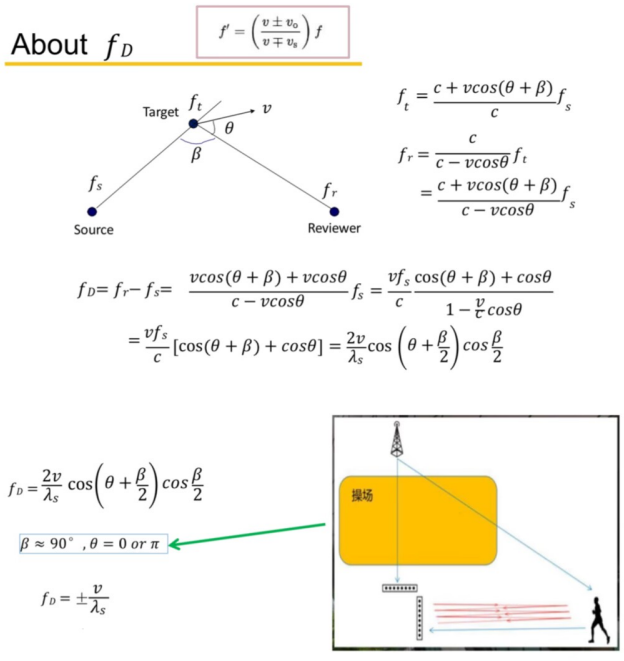


Figure 10: Principle to calculate velocity through Doppler frequency

Based on the Time-Doppler spectrum that has been plotted in Figure11, it can be observed that the motion of the object exhibits characteristics consistent with real-world scenarios. The entire motion process can be roughly divided into six stages:

① Approximately 0 to 1.5 seconds: During this phase, the object is moving towards the base station, and its velocity gradually increases.

② Approximately 1.5 to 3 seconds: The object continues to move towards the base station, and its velocity remains relatively constant.

③ Approximately 3 to 4.5 seconds: The object is still moving towards the base station, but its velocity gradually decreases, eventually reaching zero.

④ Approximately 4.5 to 5.5 seconds: During this stage, the object starts moving away from the base station, and its velocity gradually increases.

⑤ Approximately 5.5 to 8 seconds: The object continues to move away from the base station, and its velocity remains relatively constant.

⑥ Approximately 8.5 to 9.5 seconds: In this final phase, the object is moving away from the base station, and its velocity decreases until it reaches approximately zero.

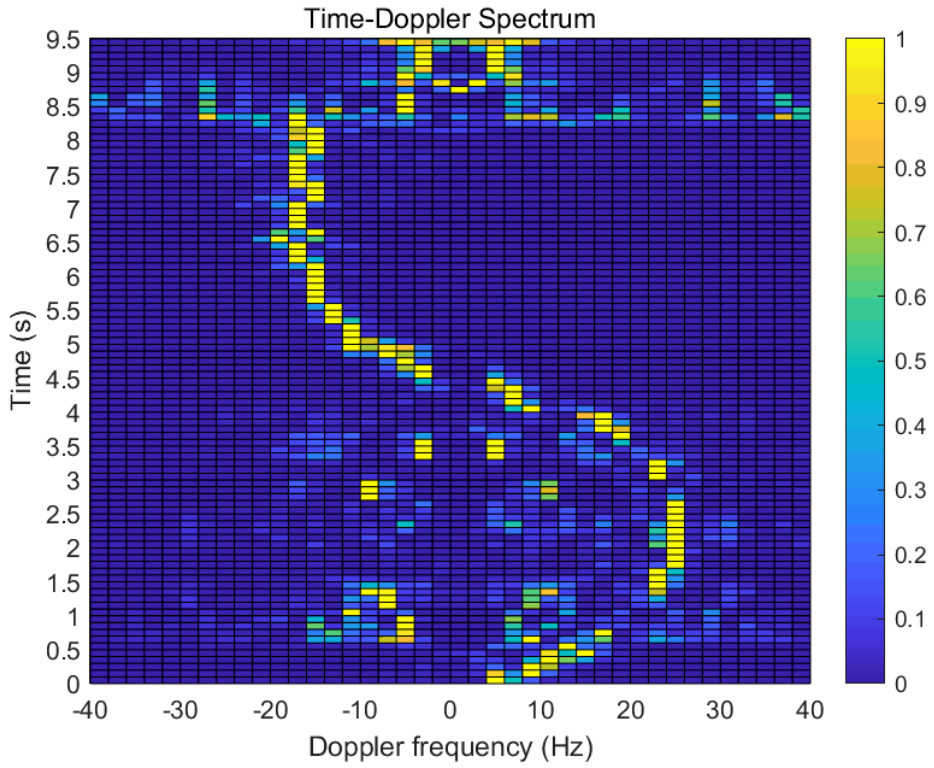


Figure 11: Time-Doppler spectrum with Δt = 0.1s, Δf_D = 2Hz

3 Extension

3.1 Further refinement of the Time-Doppler spectrum on the scale.

When plotting the Time-Doppler spectrum, further subdividing the time and frequency scale is beneficial for capturing finer details of the motion of the object. In Task3, we employed a time division of 0.1 seconds using a sliding window with the frequency as 2Hz. In the Extension section, we further refined the analysis by utilizing a 0.05-second sliding window or 1-Hz frquency division for the Time-Doppler spectrum plotting.

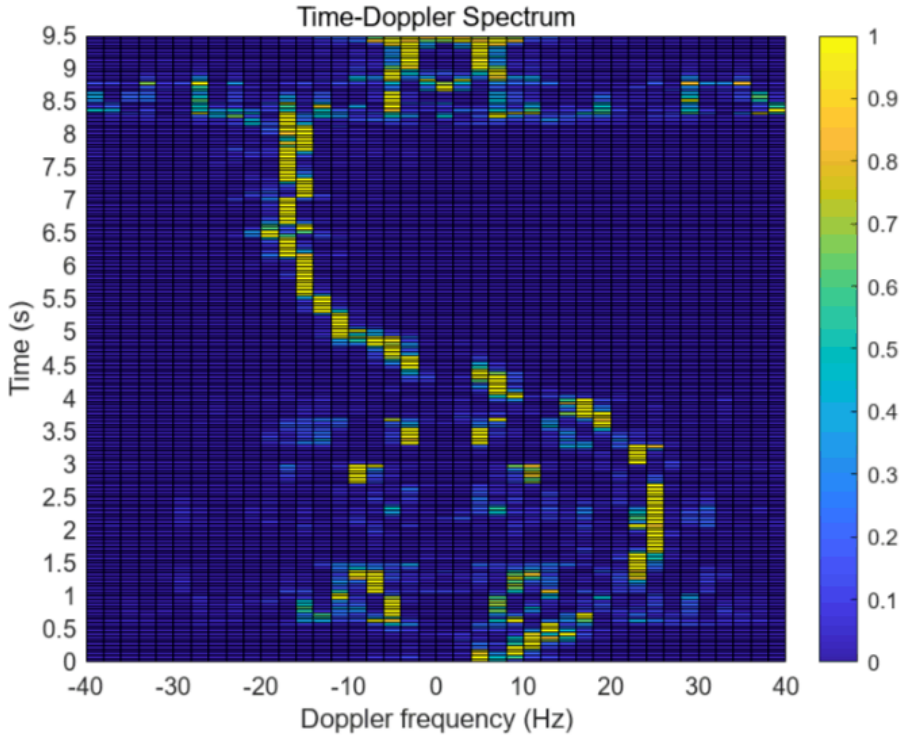


Figure 12: Time-Doppler spectrum with $\Delta t = 0.05s$

Based on the Figure12, it can be observed that a time scale of 0.1 seconds is already sufficient to effectively depict the motion of the object. Using a finer time granularity does not significantly enhance the characterization of the object's motion.

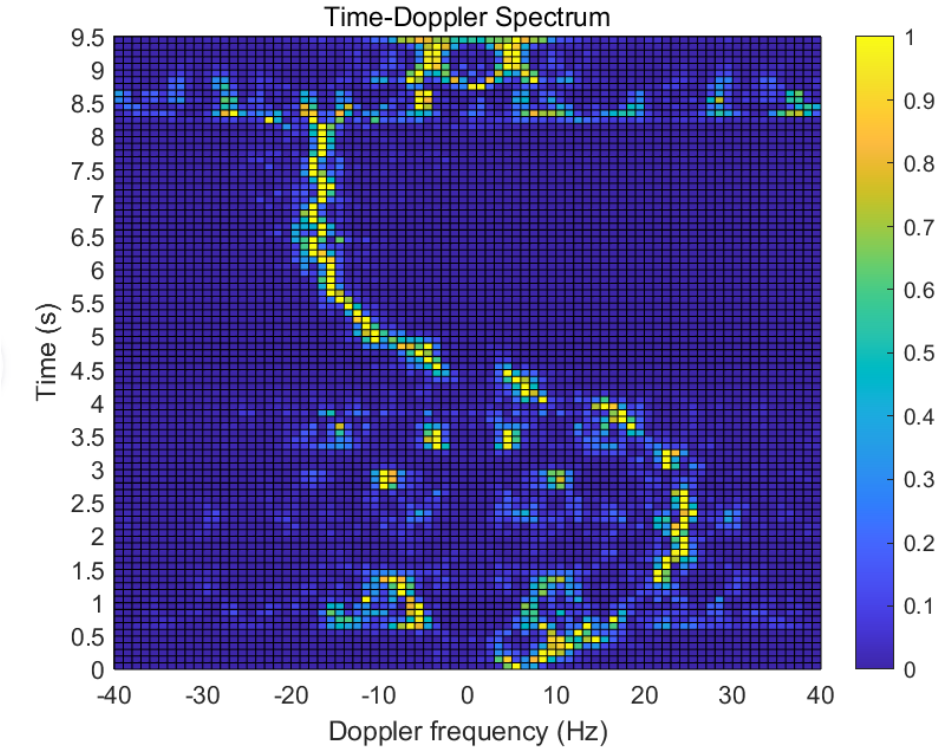


Figure 13: Time-Doppler spectrum with $\Delta f_D = 1\text{Hz}$

From Figure13, it can be seen that a frequency granularity of 1Hz, compared to 2Hz, does not significantly alter the motion trend. However, finer granularity is still advantageous for detecting the finer motion details of the moving object.

3.2 Optimization of the Algorithm

Since the data are too large, the optimization is crucial. At first, our team made use of the matrix operations recommended in MATLAB, which contains:

```
1 for j=1:41
2     corMat(i,j)=abs(sum(ys.*conj(yr).*exp_component(length(ys),Fd(j),fs
          ))));
3 end
```

However, the time cost is too large. So, we try accelerate the calculation. Define:

$$x[n] := y_{surv}[nT_s]\, y^*_{ref}[nT_s - \tau] \tag{8}$$

Note that now the ambiguity function looks like Fourier transform of $x[n]$:

$$\mathrm{Cor}\,(\tau, f_D) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi f_D n T_s} \tag{9}$$

Recall that the k-th term of Fourier Transform is defined as:

$$\sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{10}$$

When

$$-j2\pi f_D n T_s = -j2\pi kn/N \Longleftrightarrow k = f_D T_s N \tag{11}$$

they are the same.

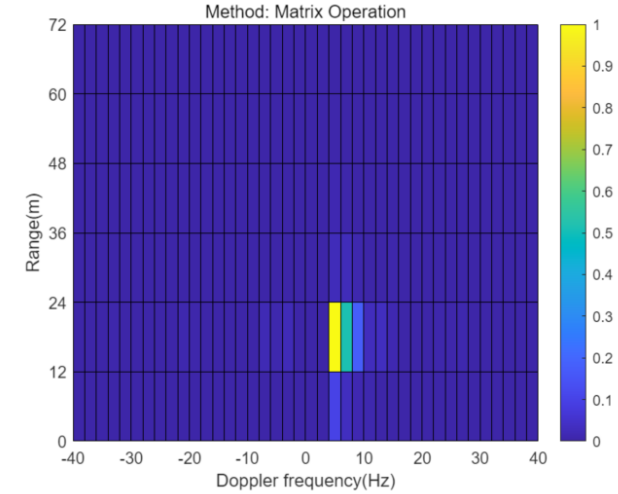
Therefore, we can use FFT to accelerate the calculation.

```
Y=fft(ys.*conj(yr));
corMat(i,:)=fftshift(abs(Y(fix(Fd.*length(ys)/fs)+21)));
```

Note that fftshift and +20 is used to make the index of negative frequencies legal, and +1 is used because the index of MATLAB starts from 1.

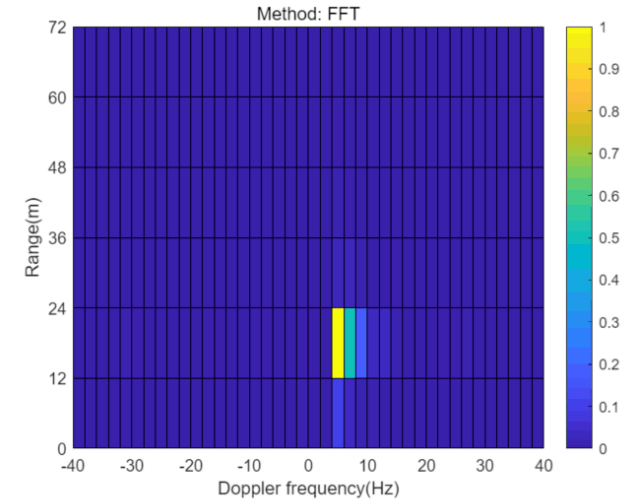
It's also a good idea to examine how efficient does MATLAB optimize the matrix operations. We try to use the for loop to calculate the ambiguity function:

```
for j=1:41
    for k=1:length(ys)
        corMat(i,j)=corMat(i,j)+ys(k).*conj(yr(k)).*exp(-2j*pi*Fd(j
            )*k/fs);
    end
    corMat(i,j)=abs(corMat(i,j));
end
```




```
tau = 4.0000e-08
fd = 4
历时 66.813195 秒。
```

Figure 14: Ambiguity function calculation via matrix operation (Time elapsed: 66.813195 seconds)




```
tau = 4.0000e-08
fd = 4
历时 9.899303 秒。
```

Figure 15: Ambiguity function calculation via FFT (Time elapsed: 9.899303 seconds)

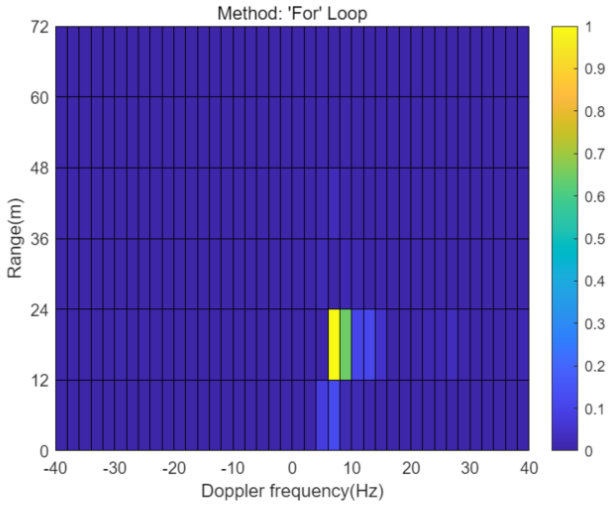


Figure 16: Ambiguity function calculation via "for" loop (Time elapsed: 627.714547 seconds)

From the comparison, we can see that MATLAB is quite efficient in matrix operations, and FFT is a good way to accelerate the calculation. We can also notice minor differences among the figures, which might be caused by errors in float number calculations.

3.3 Recovery of object's movement

From Task 3, we have obtained the Time-Doppler Spectrum, where Doppler frequecies of all times are obtained. We can get the velocity from them:

$$v = f_D \cdot c / f_C, f_C = 2123e6 \tag{12}$$

Assume the original displacement $x(0) = 0$

$$x(t) = \int_0^t v(\tau) d\tau \tag{13}$$

The $v - t$ graph and $x - t$ graph can be obtained from them.

Since we have obtained the spectra of different precisions, we can draw different graphs as follows:

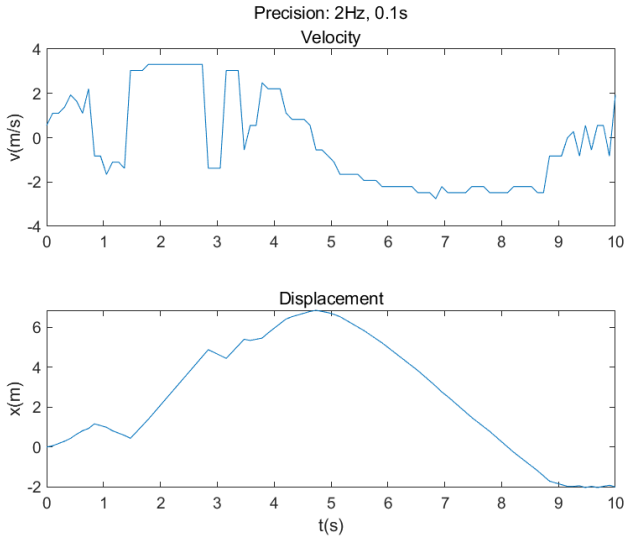


Figure 17: Motion state reconstruction (Temporal precision: 0.1s, Frequency precision: 2Hz)

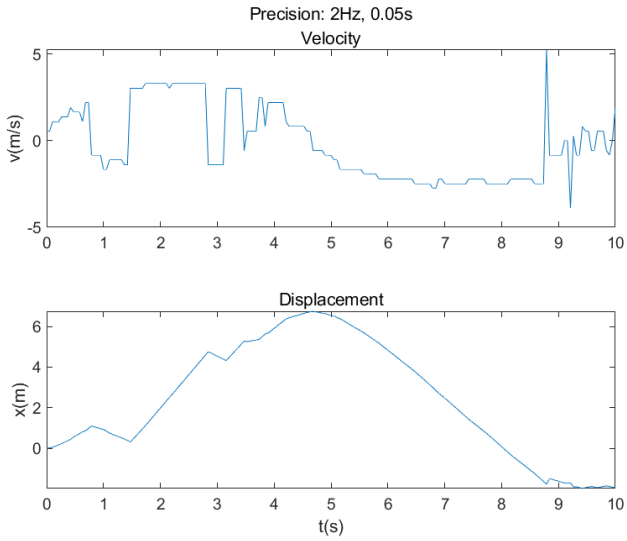


Figure 18: Motion state reconstruction (Temporal precision: 0.05s, Frequency precision: 2Hz)

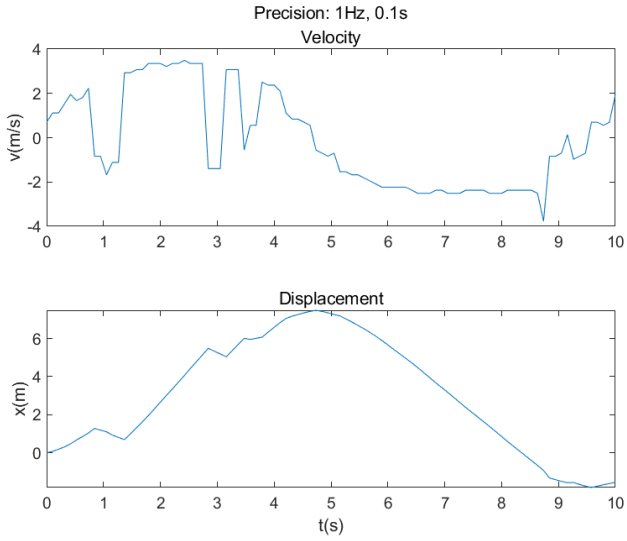


Figure 19: Motion state reconstruction (Temporal precision: 0.1s, Frequency precision: 1Hz)

Note that the differences among them are very small. We believe that the precision of 2Hz, 0.1s is acceptable.

However, the recovery of motion states using the Time-Doppler Spectrum has certain limitations. Firstly, there must be specific relative positions among the source, target, and reviewer. Additionally, there are certain requirements for the motion direction of the target.

3.4 Coding habits

Most codes in our project are encapsulated into functions. The varargin is used for flexibility. The following examples can help you understand the usage of our functions:

```
$different methods
Cor=corVar(xs,xr)
Cor=corVar(xs,xr,'for')$for_loop
$different operations
[tau,fd]=corEsti(xs,xr)
[tau,fd]=corEsti(xs,xr,'plot','Plot 1')
[tau,fd]=corEsti(xs,xr,'save','Plot 2','Plot_2.png')
```

This helps improve the readability, maintainability and reusability of our codes, and helps with our team collaboration. Furthermore, this project is maintained by Git.

4 Experience

In this project, we learned how to detect moving objects by using the reference signal and surveillance signal, combined with principles such as digital downconversion, ambiguity function, and Doppler frequency shift."

In this project, the biggest challenge we faced was the vast amount of data involved, which imposed high demands on computation time and memory. This, on the one hand, prompted us to explore algorithm optimization, and on the other hand, led us to improve our coding practices. Due to the lengthy computation time required for each task, small typos that prevented the timely generation of result graphs and necessitated rerunning the code proved to be a significant waste of time and effort. Through this project, we developed the habit of promptly saving the results of matrix computations. We believe this will foster good coding practices in our future work.

We found that the scientific division of labor can hardly be achieved without the usage of Git. We learn to manage our files appropriately in this project, and develop the habits of 'divide and conquer' and encapsulation.

5 Score

张怡程 99
马文骁 99
陈嘉祺 99
高进 99

6 Appendix

6.1 Task1

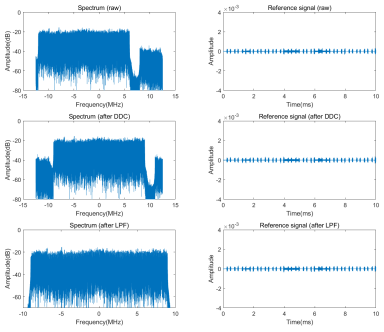


Figure 20: Reference Signal: 0.5∼1.0s

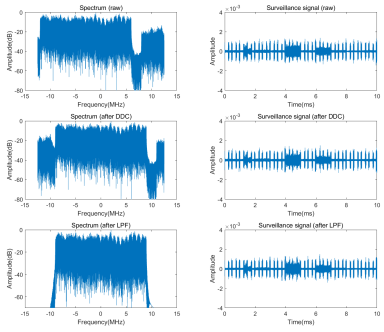


Figure 21: Surveillance Signal: 0.5∼1.0s

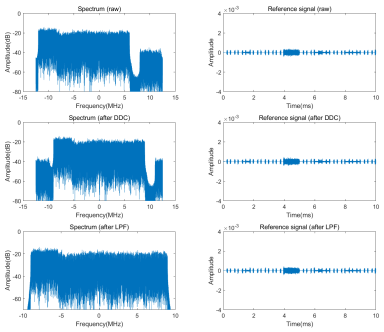


Figure 22: Reference Signal: 1.00∼1.5s

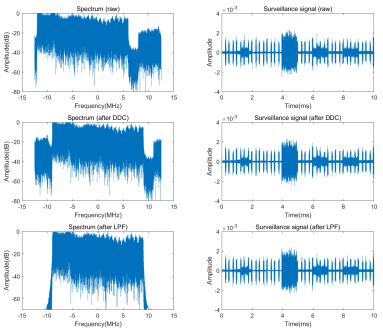


Figure 23: Surveillance Signal: 1.0∼1.5s

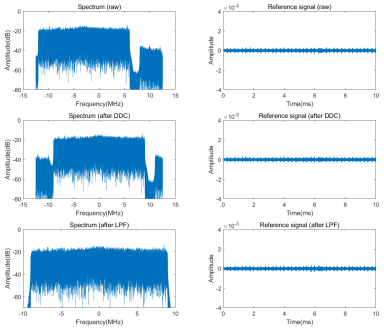


Figure 24: Reference Signal: 1.5∼2.0s

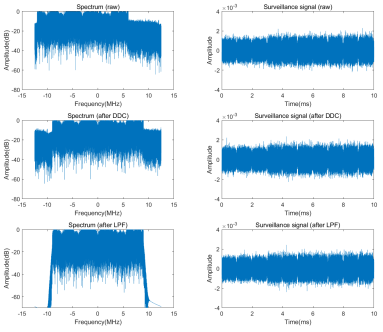


Figure 25: Surveillance Signal: 1.5∼2.0s

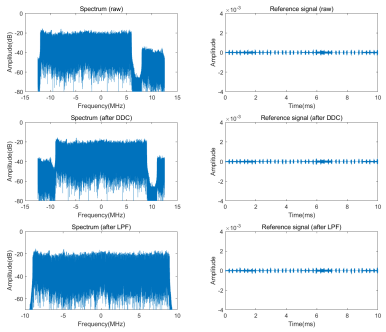


Figure 26: Reference Signal: 2.0∼2.5s

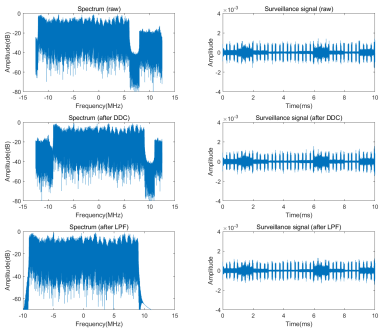


Figure 27: Surveillance Signal: 2.0∼2.5s

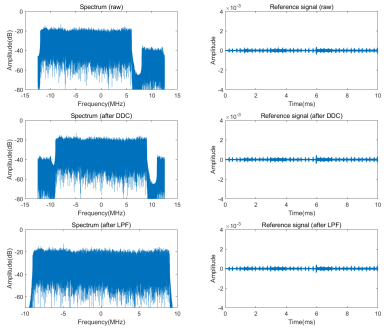


Figure 28: Reference Signal: 2.5∼3.0s

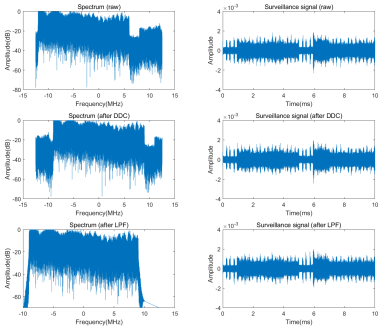


Figure 29: Surveillance Signal: 2.5∼3.0s

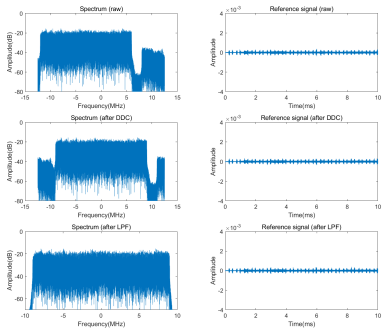


Figure 30: Reference Signal: 3.0∼3.5s

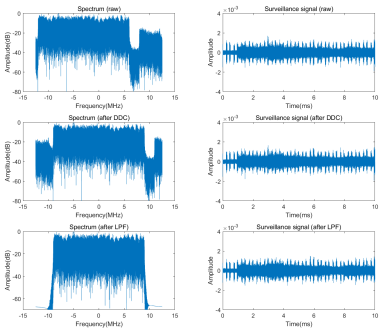


Figure 31: Surveillance Signal: 3.0∼3.5s

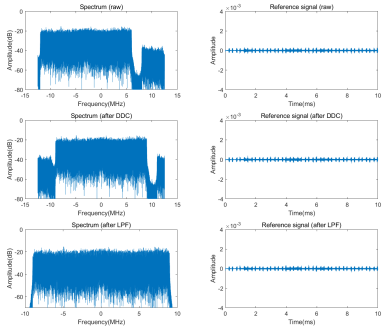


Figure 32: Reference Signal: 3.5∼4.0s

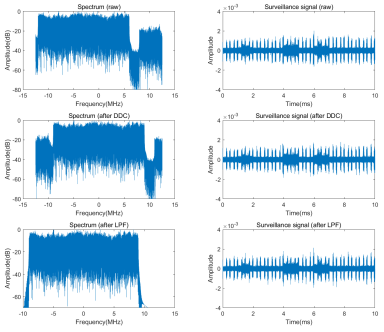


Figure 33: Surveillance Signal: 3.5∼4.0s

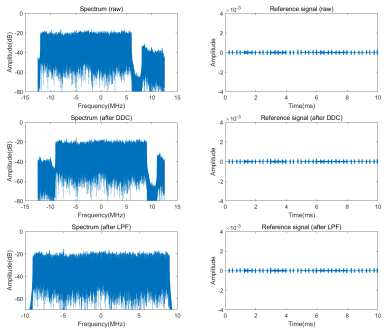


Figure 34: Reference Signal: 4.0∼4.5s

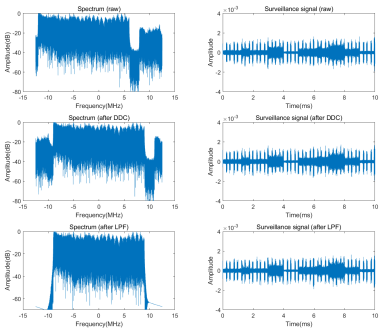


Figure 35: Surveillance Signal: 4.0∼4.5s

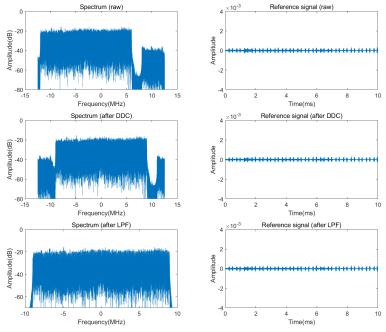


Figure 36: Reference Signal: 4.5∼5.0s

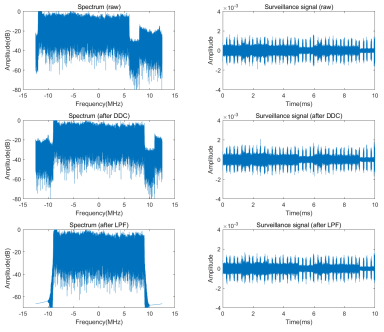


Figure 37: Surveillance Signal: 4.5∼5.0s

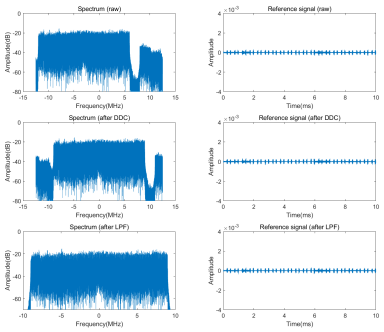


Figure 38: Reference Signal: 5.0∼5.5s

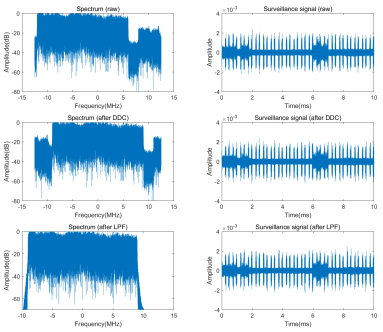


Figure 39: Surveillance Signal: 5.0∼5.5s

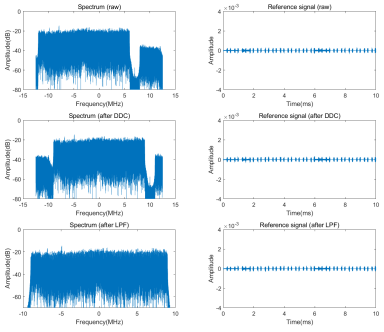


Figure 40: Reference Signal: 5.5∼6.0s

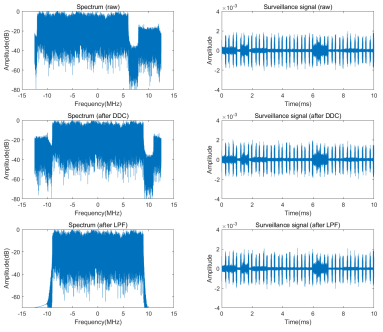


Figure 41: Surveillance Signal: 5.5∼6.0s

6.2 Task2

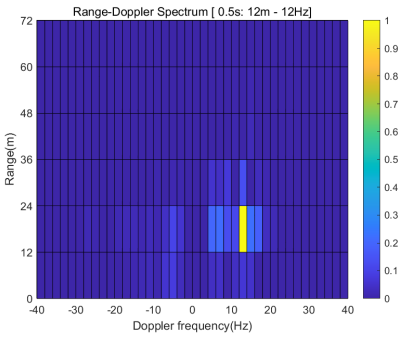


Figure 42: Reference Signal: 0.5∼1.0s

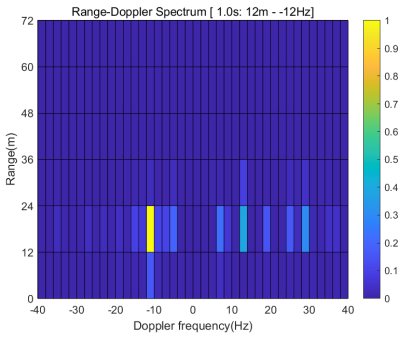


Figure 43: Surveillance Signal: 1.0∼1.5s

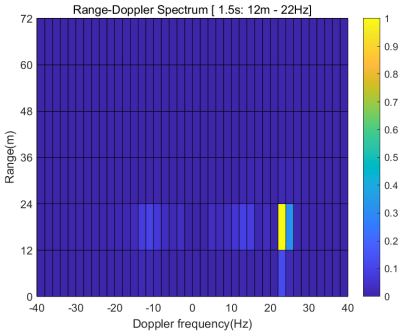


Figure 44: Reference Signal: 1.5∼2.0s

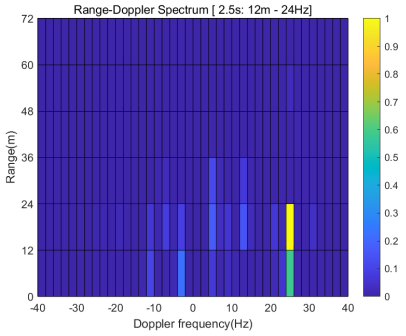


Figure 45: Surveillance Signal: 2.5∼3.0s

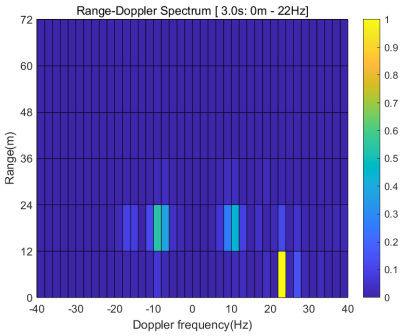


Figure 46: Reference Signal: 3.0∼3.5s

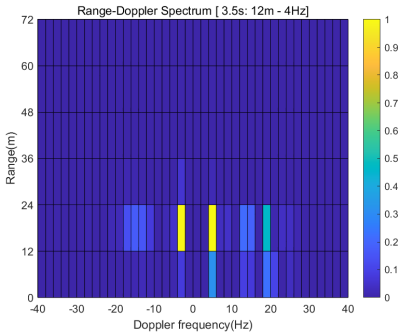


Figure 47: Surveillance Signal: 3.5∼4.0s

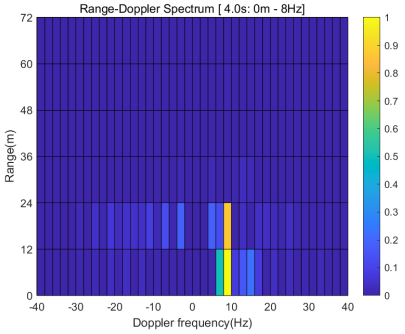


Figure 48: Reference Signal: 4.0∼4.5s

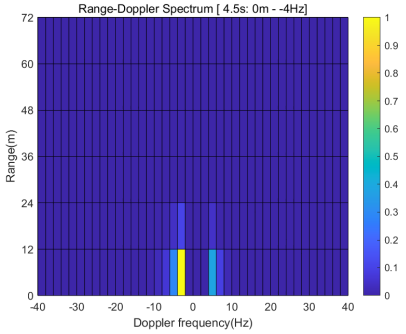


Figure 49: Surveillance Signal: 4.5∼5.0s

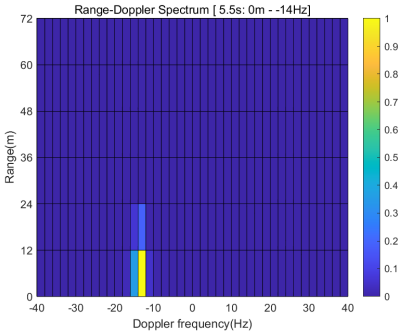


Figure 50: Reference Signal: 5.5∼6.0s

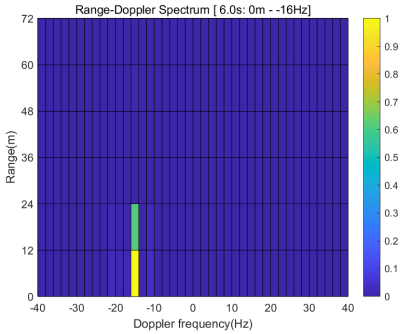


Figure 51: Surveillance Signal: 6.0∼6.5s

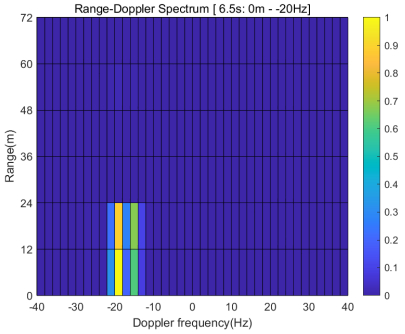


Figure 52: Reference Signal: 6.5∼7.0s

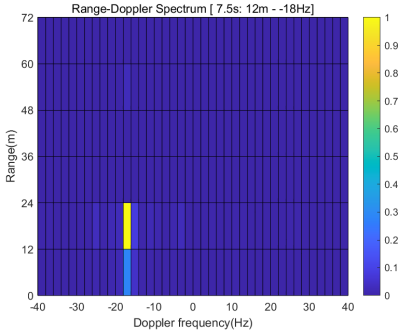


Figure 53: Surveillance Signal: 7.5∼8.0s

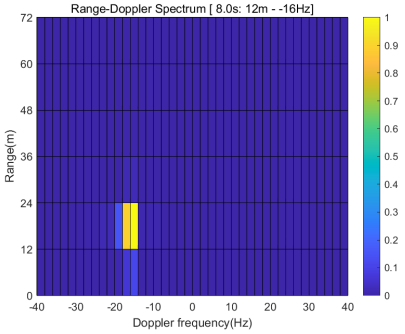


Figure 54: Reference Signal: 8.0∼8.5s

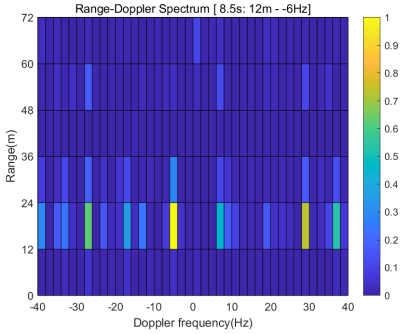


Figure 55: Surveillance Signal: 8.5∼9.0s

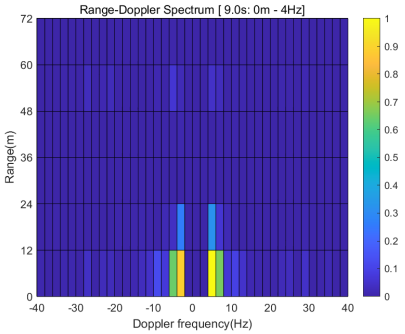


Figure 56: Reference Signal: 9.0∼9.5s

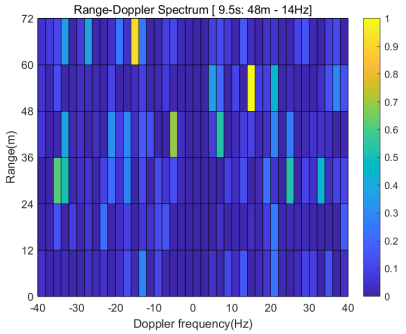


Figure 57: Surveillance Signal: 9.5∼10.0s

7 Code

7.1 Basic code for Task1-3

```
for data_index = 1:20
    folderPath = sprintf('C:\\Users\\Lenovo\\Desktop\\Project 2\\
        rawData\\data_%d\\', data_index);
    addpath(folderPath);
    filename = sprintf('data_%d.mat', data_index);
    load(filename);

%Task1
    %暂时先只处理0.01s
    seq_sur1=seq_sur(1:(length(seq_sur)/50));
    seq_ref1=seq_ref(1:(length(seq_sur)/50));
    duration1=0.01;

    %reference signal:freq-domain
    drawSpectrum(seq_ref1,f_s)
    drawDDCSpectrum(seq_ref1,f_s)
    AfterLPFSpectrum(seq_ref1,f_s);

    %reference signal:time-domain
    TimePlotting(seq_ref1,f_s,"Reference signal (raw)")
    DDC_TimePlotting(seq_ref1,f_s,"Reference signal (after DDC)")
    AfterLPF_TimePlotting(seq_ref1,f_s,"Reference signal (after LPF)",
        "Reference Signal",data_index)

    %surveillance signal:freq-domain
    drawSpectrum(seq_sur1,f_s)
    drawDDCSpectrum(seq_sur1,f_s)
    AfterLPFSpectrum(seq_sur1,f_s);

    %surveillance signal:time-domain
    TimePlotting(seq_sur1,f_s,"Surveillance signal (raw)")
    DDC_TimePlotting(seq_sur1,f_s,"Surveillance signal (after DDC)")
    AfterLPF_TimePlotting(seq_sur1,f_s,"Surveillance signal (after LPF)
        ", "Surveillance Signal",data_index)

%Task2
    seq_ref_after_LPF=AfterLPF_Timeseries(seq_ref,f_s);
    seq_sur_after_LPF=AfterLPF_Timeseries(seq_sur,f_s);

    cor_matrix=zeros([7,41]);
    N_tau=0:6;
    f_D=-40:2:40;
    for i=1:1:7
        for j=1:1:41
            seq_sur_cor=[seq_sur_after_LPF,zeros(1,N_tau(i))];
            seq_ref_cor=[zeros(1,N_tau(i)),seq_ref_after_LPF];
            n_spaced=0:length(seq_sur_cor)-1;
            cor_matrix(i,j)=abs(sum(seq_sur_cor.*conj(seq_ref_cor).*exp
                ((-2i*pi*f_D(j)/f_s)*n_spaced)));
        end
    end

    minVal = min(cor_matrix(:));
```

```
50      maxVal = max(cor_matrix(:));
51
52      % 进行 0-1 标准化,3次方加强效果
53      matrix_normalized = ((cor_matrix - minVal) / (maxVal - minVal)).^3
54
55      % 使用max函数找到矩阵中的最大元素及其位置
56      [maxValue, linearIndex] = max(matrix_normalized(:));
57      % 将线性索引转换为行列索引
58      disp("max value index")
59      [row, col] = ind2sub(size(matrix_normalized), linearIndex)
60      cor_max_f_D=(col-1)*2-40
61      cor_max_tau=(row-1)*3e8/f_s
62
63      % 保存矩阵
64      dataname=sprintf("matrix_normalized data%d.mat', data_index)
65      save(dataname, "matrix_normalized")
66
67      figure
68      pcolor(f_D,N_tau*3e8/f_s,matrix_normalized)
69      colorbar;
70      xlabel("Doppler frequency(Hz)"),ylabel("Range(m)"),yticks
            ([0,12,24,36,48,60,72]);
71      Title = sprintf('Range-Doppler Spectrum [%4.1fs: %dm - %dHz]', 0.5*
            data_index-0.5, cor_max_tau, cor_max_f_D);
72      title(Title)
73
74      % 保存图形
75      filename = sprintf('Range-Doppler Spectrum data%d.png', data_index)
76      saveas(gcf, filename);
77
78  end
79
80  %prepare for Task3
81  %{
82  ref_concatenated=zeros([1,12500000*20]);
83  sur_concatenated=zeros([1,12500000*20]);
84
85  for data_index = 1:20
86      folderPath = sprintf('rawData\\data_%d\\', data_index);
87      addpath(folderPath);
88
89      filename = sprintf('data_%d.mat', data_index);
90
91      load(filename);
92
93      seq_ref_after_LPF=AfterLPF_Timeseries(seq_ref,f_s);
94      seq_sur_after_LPF=AfterLPF_Timeseries(seq_sur,f_s);
95      disp("ref_concatenated")
96      ref_concatenated((data_index-1)*12500000+1:data_index*12500000)=
            seq_ref_after_LPF;
97      sur_concatenated((data_index-1)*12500000+1:data_index*12500000)=
            seq_sur_after_LPF;
98  end
99
100 % 使用 class 函数获取变量的类
101 variableClass = class(ref_concatenated);
102 % 显示变量的类
```

```
disp(['变量的类是: ', variableClass]);

ref_concatenated = double(ref_concatenated);
sur_concatenated = double(sur_concatenated);

save("ref_concatenated.mat","ref_concatenated")
save("sur_concatenated.mat","sur_concatenated")
%}

%Task3
ref_concatenated=load("ref_concatenated.mat")
sur_concatenated=load("sur_concatenated.mat")

ref_concatenated = struct2array(ref_concatenated);
sur_concatenated = struct2array(sur_concatenated);

% 使用 class 函数获取变量的类
variableClass = class(ref_concatenated);
% 显示变量的类
disp(['变量的类是: ', variableClass]);

time_segment=0.1
time_sliding_window=0.5
sliding_window_len=0.5*f_s;
num_sliding_windows=round((length(ref_concatenated)/f_s-
    time_sliding_window)/time_segment+1)

Time_Doppler_matrix=zeros([num_sliding_windows,41]);
Time=(0:num_sliding_windows-1)*time_segment;
N_tau=0:6;
f_D=-40:2:40;

disp("Start calculating")
k=0
for window_index=1:num_sliding_windows
    index1=round(((window_index-1)*time_segment)*f_s+1)
    index2=round((((window_index-1)*time_segment+time_sliding_window)*
        f_s));
    ref_segment=ref_concatenated(index1:index2);
    sur_segment=sur_concatenated(index1:index2);

    cor_matrix=zeros([7,41]);
    for i=1:1:7
        for j=1:1:41
            seq_sur_cor=[sur_segment,zeros(1,N_tau(i))];
            seq_ref_cor=[zeros(1,N_tau(i)),ref_segment];
            n_spaced=0:length(seq_sur_cor)-1;
            cor_matrix(i,j)=abs(sum(seq_sur_cor.*conj(seq_ref_cor).*exp
                ((-2i*pi*f_D(j)/f_s)*n_spaced)));
        end
    end
    minVal = min(cor_matrix(:));
    maxVal = max(cor_matrix(:));
    matrix_normalized = ((cor_matrix - minVal) / (maxVal - minVal)).^3;
    [maxValue, linearIndex] = max(matrix_normalized(:));
    [row, col] = ind2sub(size(matrix_normalized), linearIndex);
    Time_Doppler_matrix(window_index,:)=matrix_normalized(row,:);
```

```
157     k=k+1
158 end
159
160 % 保存矩阵
161 dataname="Time-Doppler Matrix.mat"
162 save(dataname, "Time_Doppler_matrix")
163
164 figure
165 pcolor(f_D,Time,Time_Doppler_matrix)
166 colorbar;
167 xlabel("Doppler frequency (Hz)"),ylabel("Time (s)")
168 yticks(0:0.5:9.5);
169 title("Time-Doppler Spectrum")
170
171 save("Time_Doppler_matrix.mat","Time_Doppler_matrix")
172
173 % 保存图形
174 filename='Time-Doppler Spectrum.png'
175 saveas(gcf, filename);
176
177 %function
178 function drawSpectrum(x,fs)
179     N=length(x);
180     H=fftshift(fft(x));
181     f=(-N/2:N/2-1)*fs/N;
182     figure,set(gcf, 'Position', [100 100 1000 800])
183     subplot(3,2,1),plot(f./10^6,20*log10(abs(H)))
184     xlabel("Frequency(MHz)"),ylabel('Amplitude(dB)'),ylim([-80,0])
185     title('Spectrum (raw)')
186 end
187
188 function [H,H_filtered]=SpectrumExtraction(x,fs)
189     f_ddc=-3e6;
190     bandwidth=9e6;
191     x=x.*exp(-1j*2*pi*f_ddc*xTime(x,fs));
192
193     %DDC
194     H=fftshift(fft(x));
195
196     %LPF
197     [b,a]=butter(20,bandwidth/(fs/2),'low');
198     x_filtered=filter(b,a,x);
199     H_filtered=fftshift(fft(x_filtered));
200 end
201
202 function drawDDCSpectrum(x,fs)
203     [H,~]=SpectrumExtraction(x,fs);
204
205     N=length(x);
206     f=(-N/2:N/2-1)*fs/N;
207     subplot(3,2,3),plot(f./10^6,20*log10(abs(H)))
208     xlabel("Frequency(MHz)"),ylabel('Amplitude(dB)'),ylim([-80,0])
209     title('Spectrum (after DDC)')
210 end
211
212 function AfterLPFSpectrum(x,fs)
213     [~,H]=SpectrumExtraction(x,fs);
```

```
214
215     N=length(x);
216     f=(-N/2:N/2-1)*fs/N;
217     subplot(3,2,5),plot(f./10^6,20*log10(abs(H)))
218     ylabel('Amplitude(dB)'),ylim([-70,0]),xlabel('Frequency(MHz)')
219     title('Spectrum (after LPF)')
220 end
221
222 function t=xTime(x,fs)
223     t=(0:length(x)-1)./fs;%将x轴用时间表示
224 end
225
226 function TimePlotting(x,fs,Title)
227     t=xTime(x,fs);
228     subplot(3,2,2),ylim([-4*1e3,4*1e3]),plot(t*1e3,x)
229     ylabel("Amplitude"),xlabel("Time(ms)"),ylim([-4*1e-3,4*1e-3])
230     title(Title)
231 end
232
233 function DDC_TimePlotting(x,fs,Title)
234     f_ddc=-3e6;
235
236     t=xTime(x,fs);
237     x=x.*exp(-1j*2*pi*f_ddc*xTime(x,fs));
238     subplot(3,2,4),plot(t*1e3,x)
239     ylabel("Amplitude"),xlabel("Time(ms)"),ylim([-4*1e-3,4*1e-3]),title
            (Title)
240 end
241
242 function AfterLPF_TimePlotting(x,fs,Title,Filename,data_index)
243     f_ddc=-3e6;
244     bandwidth=9e6;
245     t=xTime(x,fs);
246
247     x=x.*exp(-1j*2*pi*f_ddc*xTime(x,fs));
248     [b,a]=butter(20,bandwidth/(fs/2),'low');
249     x_filtered=filter(b,a,x);
250     subplot(3,2,6),plot(t*1e3,x_filtered);
251     ylabel("Amplitude"),xlabel("Time(ms)"),ylim([-4*1e-3,4*1e-3]),title
            (Title)
252
253     filename=sprintf('%s data%d.png', Filename, data_index);
254     saveas(gcf, filename);
255 end
256
257 function x_filtered=AfterLPF_Timeseries(x,fs)
258     f_ddc=-3e6;
259     bandwidth=9e6;
260     t=xTime(x,fs);
261
262     x=x.*exp(-1j*2*pi*f_ddc*xTime(x,fs));
263     [b,a]=butter(20,bandwidth/(fs/2),'low');
264     x_filtered=filter(b,a,x);
265 end
```

7.2 Improvement on code

In this section, we will improve the code based on the coding habits of Extension4 and complete Extension2-3 based on the improved code.

```
%Extension2: Optomization of algorithm
load("data_1.mat");
xr=seq_ref;
xs=seq_sur;

[tau,fd]=corEsti(xs,xr,'save','Method: FFT','Method_FFT.png')

[tau,fd]=corEsti(xs,xr,'plot','Method: Matrix Operation')

tau,fd
tic
[~,~]=corEsti(xs,xr,'plot','Method: ''For'' Loop')
toc

load("data_2.mat");
xr=seq_ref;
xs=seq_sur;figure
tic
[~,~]=corEsti(xs,xr,'plot','2')
toc



load("data_3.mat");
xr=seq_ref;
xs=seq_sur;figure
tic
[~,~]=corEsti(xs,xr,'plot','3')
toc

load("data_4.mat");
xr=seq_ref;
xs=seq_sur;figure
tic
[~,~]=corEsti(xs,xr,'plot','4')
toc

load("data_5.mat");
xr=seq_ref;
xs=seq_sur;figure
tic
[~,~]=corEsti(xs,xr,'plot','5')
toc

load("data_6.mat");
xr=seq_ref;
xs=seq_sur;figure
tic
[tau,fd]=corEsti(xs,xr,'plot','6')
toc

function [tau,fd]=corEsti(y_surv,y_ref,varargin)
    %可选参数：是否作图?'plot','save'
```

```
54      fs=25000000;
55      corMat=zeros(7,41);
56      Tau=linspace(0,1/fs*6,7);
57      Fd=linspace(-40,40,41);
58      for i=1:7
59          zero_compensation=zeros(1,fix(Tau(i)*fs));%时延后补长信号以对齐
60          ys=[y_surv,zero_compensation];
61          yr=[zero_compensation,y_ref];
62          %fft
63          Y=fft(ys.*conj(yr));
64          corMat(i,:)=fftshift(abs(Y(fix(Fd.*length(ys)/fs)+21)));
65
66          %dot
67          % for j=1:41
68          %     corMat(i,j)=abs(sum(ys.*conj(yr).*exp_component(length(ys
                ),Fd(j),fs)));
69          % end
70
71          %for
72          % for j=1:41
73          %     for k=1:length(ys)
74          %         corMat(i,j)=corMat(i,j)+ys(k).*conj(yr(k)).*exp(-2j*
                pi*Fd(j)*k/fs);
75          %     end
76          %     corMat(i,j)=abs(corMat(i,j));
77          % end
78      end
79      [~,I]=max(corMat(:));%寻找corMat中最大值索引
80      [m,n]=ind2sub(size(corMat),I);%索引转为横纵索引
81
82      tau=Tau(m);
83      fd=Fd(n);
84
85      %是否作图
86      numVar=nargin-2;
87
88      if isempty(varargin)
89          return
90      elseif strcmp(varargin{1},'plot')||strcmp(varargin{1},'save')
91          if numVar<2
92              disp('No enough arguments. Check title and/or filename.')
93              return
94          end
95          h=figure;
96
97          minVal = min(corMat(:));
98          maxVal = max(corMat(:));
99
100         % 进行 0-1 标准化,3次方加强效果
101         corMat=((corMat - minVal) / (maxVal - minVal)).^3;
102
103         pcolor(Fd,Tau.*3e8,corMat);colorbar
104
105         xlabel("Doppler frequency(Hz)"),ylabel("Range(m)"),yticks
                (0:12:72)
106         title(varargin{2})
107
```

```
108             if strcmp(varargin{1},'save')
109                 if numVar<3
110                     disp('Filename not detected!')
111                     return
112                 end
113                 saveas(h,varargin{3},'png')
114             end
115         else
116             disp('Wrong input.')
117         end
118 end
119
120 function y=exp_component(n,fd,fs)%dot法要调用的函数
121     y=exp((-2i*pi*fd/fs).*linspace(0,n-1,n));
122 end
123
124
125 %Extension3: Recovery of the movement
126 load('Time_Doppler_matrix.mat');
127 mat1=Time_Doppler_matrix;
128 load('Time_Doppler_matrix_1_Hz.mat')
129 mat2=Time_Doppler_matrix;
130 load('Time_Doppler_matrix_005s.mat')
131 mat3=Time_Doppler_matrix;
132 clear Time_Doppler_matrix
133
134 drawMovement(mat1,'Precision: 2Hz, 0.1s','2Hz_0.1s.png')
135
136 drawMovement(mat2,'Precision: 1Hz, 0.1s','1Hz_0.1s.png')
137
138 drawMovement(mat3,'Precision: 2Hz, 0.05s','2Hz_0.05s.png')
139
140 function drawMovement(Mat,Title,varargin)
141     Nt=length(Mat);
142     Nf=length(Mat(1,:));
143
144     speed=zeros(1,length(Mat));
145     for k=1:length(Mat)
146         [~,speed(k)]=max(Mat(k,:));
147         %现在speed是索引
148     end
149
150     %t=(1:length(speed))./(length(Mat)/9.5);
151     t=linspace(0,10,Nt);
152
153     %speed=(speed-21).*2;%索引是1:41, 修正为-20:2:20
154     speed=(speed-(Nf+1)/2);
155     %索引由1:N变为-N/2:N/2
156     speed=speed.*(80/Nf);
157     speed=speed.*(3e8/2123e6);
158
159     h=figure;subplot(2,1,1),plot(t,speed),subtitle('Velocity'),ylabel('
            v(m/s)'),title(Title)
160     loc=zeros(1,length(Mat));
161     for k=2:length(Mat)
162         loc(k)=loc(k-1)+speed(k-1)*(t(2)-t(1));%t(2)-t(1)是数据点间的时
                间间隔
```

```
163     end
164
165     subplot(2,1,2),plot(t,loc),subtitle('Displacement'),ylabel('x(m)'),
            xlabel('t(s)')
166
167     if isempty(varargin)
168         return
169     else
170         saveas(h,varargin{1},'png')
171     end
172 end
```