

Project 2. 基于通信信号的运动检测

12212859 黎思婕 12212860 魏雨杉 12212861 黄姝颜

Introduction

1. 检测原理

Project2 旨在利用信号传递的多径效应检测人的运动，获得运动中的人不同时间的位置和速度信息。如图 1 所示，detector 可以接受直接从雷达传过来的参考信号 seq_{ref} 和雷达信号经过人再传回来的监测信号 seq_{sur} ，由于发射的雷达距离 detector 很远，所以近似两个信号传播路程之差就是人离 detector 的距离 x 。

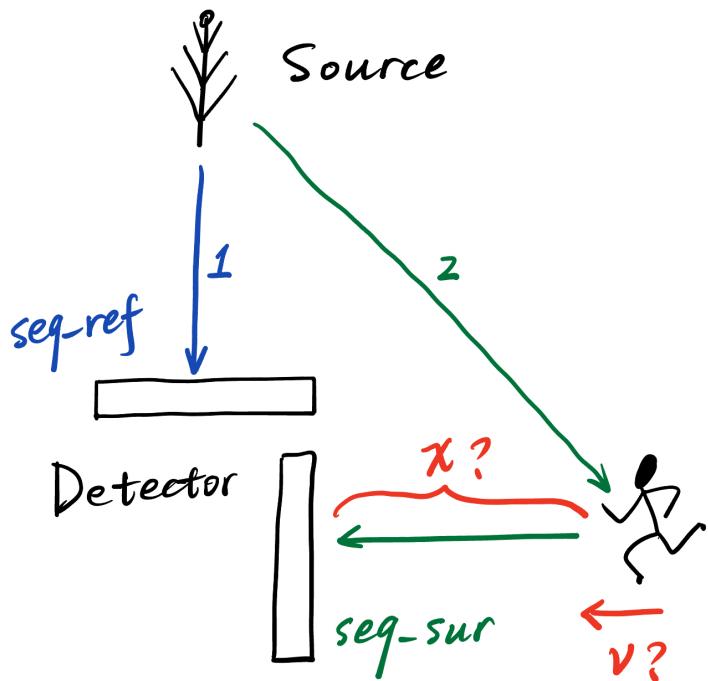


Figure 1. 运动检测原理

由速度位移和多普勒效应公式

$$x = c\tau$$

$$v = \frac{f_d c}{f_c}$$

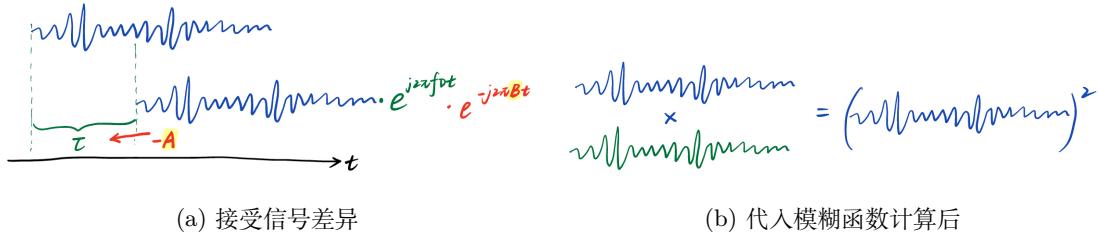
可知，我们只用求出 τ 和 f_d 即可得到位置和时间信息。其中光速 c 和载波中心频率 f_c 为已知量，而信号传播的时间差 τ 和多普勒频率 f_d 为待求量。

τ 和 f_d 的获取依靠认为构造的模糊函数寻找得到，即模糊函数（离散形式）

$$Cor(\tau, f_d) = \sum_{n=0}^{N-1} y_{surv}[nT_s] y_{ref}^*[nT_s - \tau] e^{-j2\pi f_d nT_s}$$

取得最大值的时候， τ 和 f_d 就是想要得到的时间差和多普勒频率。

模糊函数的理解 Detector 接收到的信号差异在于监测信号相较于参考信号多了一个时延和多普勒频率（如图 2a），代入模糊函数后，若模糊函数 τ 和 f_d 就是想要得到的时间差和多普勒频率，相当于将信号时移回去并将多普勒频率抵消（如图 2b），此时相当于信号本身做了平方，模糊函数达到最大值。



2. 信号处理

数据包里的信号包含了两种不同的信号，我们需要取出我们想要的信号部分代入模糊函数计算，因此需要先对信号进行处理。

DDC 一般滤波器都是低通滤波器，为了信号能够经过低通滤波器后取得特定的成分，我们需要先将目标成分的中心频率频移至 0Hz，进行数字下变频（DDC）的操作。信号在频域做平移，但是频域只做理解，实际对时域进行操作，相当于时域信号 $\times e^{j2\pi f_{ddc}t}$ ，其中，平移的频率量 f_{ddc} 已知。

LPF 然后将 DDC 处理后的信号经过低通滤波器（LPF）滤除其他信号成分。经探索，我们设计了 20 阶（滤波效果较好）的巴特沃斯滤波器，取已知带宽进行设计。由图 3 明显看出，滤波器能较好保留 9MHz 带宽的成分。

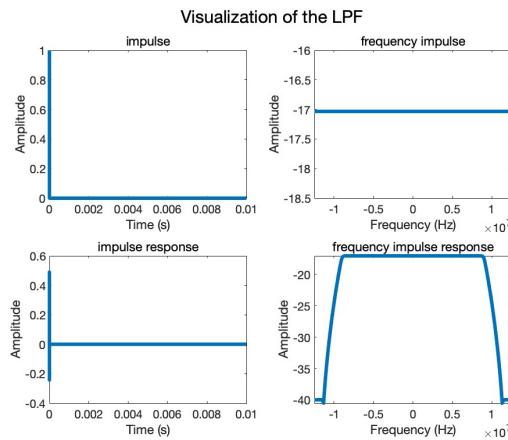


Figure 3. 低通滤波器的验证（图示为冲激响应）

注意 我们得到的信号是经过了去除噪声、移频、取特定方向信号等的处理的，所以引入了复数部分，并不完全是实信号，在处理的过程需要考虑到这一点。

3. 计算过程

为了找出使得模糊函数最大的 τ 和 f_d , 我们对这两个量进行了遍历, 并用循环计算出不同的 $\hat{\tau}$ 和 \hat{f}_d 代入模糊函数得到的值, 将所得并入一个矩阵并用三维图像画出, 从而在三维图像中直观看出两个变量的改变所对应的值的相对大小。得到了每段时间的速度和位置, 我们可以还原出人运动的大致动态过程。

τ 的遍历值 一方面, 采样率限制了遍历的最小时间间隔 $\Delta\tau_{min}$, 因为 $\Delta\tau_{min} = \frac{1}{f_s}$, 对应的, 由 $(\Delta x)_{min} = c \times \Delta\tau_{min} = c \times \frac{1}{f_s} = 12m$ 我们可以得知距离的精度为 12 米, 这是一个较低的精度, 这也是为什么我们不用 $\frac{\Delta x}{\Delta t}$ 来求速度。另一方面, 我们通过实际经验确定遍历范围, 因为人运动的范围大致在 $0 \sim 72m$ 之内, 通过 $x = c \times \tau$ 我们得知只需遍历 $\tau = 0 \sim 6\frac{1}{f_s}s$ 的情况。

f_d 的遍历值 一方面, 一次计算的样本点数量限制了最小多普勒频率间隔 $(\Delta f_d)_{min}$ 。推导傅立叶变换后作图的横坐标时我们有公式:

$$\omega_0 = 2\pi(\Delta f_d)_{min} = \frac{2\pi}{N\tau_0}$$

已知 $\tau_0 = \frac{1}{f_s}$, $N = \frac{\text{duration}}{\tau_0}$ (片段样本数), 计算可以得出, 当 $N = 12,500,000$ 时 (一个数据包的样本点个数) $(\Delta f_d)_{min} = 2Hz$ 。另一方面, 人运动的速度限制了 f_d 的遍历范围, 由公式 $v = \frac{f_d c}{f_c}$ 可以算得, 当人运动速度在 $-5.65m/s \sim 5.65m/s$ 时对应的频率范围为 $-40Hz \sim 40Hz$ 。

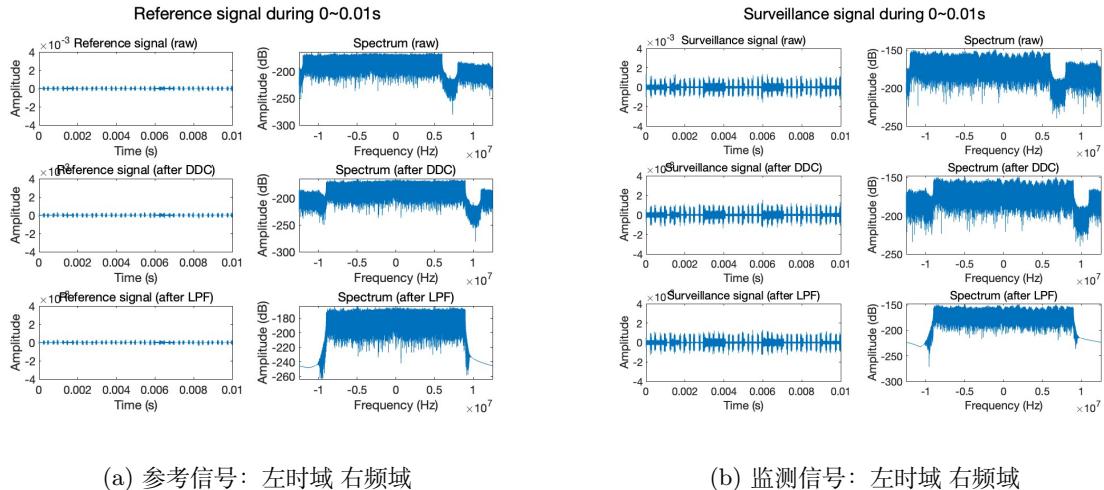
4. 技术应用

基于通信信号的运动检测技术利用现有的无线通信信号 (如 Wi-Fi、蜂窝网络信号等) 来检测和分析环境中的移动物体。该技术无需额外的传感器设备, 具有低成本、高隐蔽性和广泛覆盖的优势。在智能家居中, 基于通信信号的运动检测可以用于安防监控、老人跌倒检测和儿童活动监控等, 实现对家庭成员的安全保护。在智能交通系统中, 该技术可以用于车辆检测、交通流量监控和行人检测, 从而优化交通信号和提高道路安全。此外, 基于通信信号的运动检测在医疗健康、无人机监控、仓储物流等领域也有广泛的应用前景。通过分析通信信号的变化模式, 系统可以准确地识别和跟踪物体的运动轨迹, 为各类应用场景提供可靠的数据支持和智能化解决方案。

Result and Analysis

1. Task1

任务要求画出频域和时域中各个节点时 $0 \sim 0.01$ 秒信号的图像, 并按要求标注、设定横纵坐标。如图 4 所示, 从频域中很明显可以看到 DDC 后目标信号 (带宽更长的) 中心频率被平移至 $0Hz$, 经过低通滤波器之后大于 $9MHz$ 的成分基本被滤除。



(a) 参考信号: 左时域 右频域

(b) 监测信号: 左时域 右频域

Figure 4. 0~0.01s 信号处理过程展示

注意

- Matlab 作图消耗大，所以只画 0~0.01s 的部分。而画 0~0.01s 只处理这么长时间的来做图，因为傅立叶变换后无法对时间操作，无法取 0~0.01s 对应的频率成分做图；
- 原数据因为经过去噪声、取了特定方向的信号、频移的处理包含了复数部分，在作时域图是用 real();
- 我们做傅立叶变换的时候乘了 tau，但是只是想看相对频率的大小也可以不乘；
- 频域振幅取了对数是 dB, $20\log_{10}(\text{abs}(X))$;
- 所有频域图像均用 fftshift 处理，不翻折过来后面的图是错的。

2. Task2

题目要求作出 data1, 5, 11, 15 的 Range Doppler spectrum。我们暴力运用 for 循环遍历 τ 和 f_d 并用 surf() 函数作图。但是由于速度太慢，要做四个图，所以图 5 中的结果都是用后面拓展部分所讲的 fft() 函数计算得到的结果完成的（见拓展部分）。

图像锐化 我们为了突出最大值使得图像不那么花，利用了指数函数的非线性映射将小的值变得更小，大的值变得更大。一开始我们取了 1000 作为指数得到图 5 中左侧一列的图像；但是在老师的提醒下我们理解了，因为模糊函数是连续的，当最大值附近的点也较大时结果会更有可信度，所以我们进一步调整讲指数定为 3，保留了部分较大值，验证了最大值点的正确性。

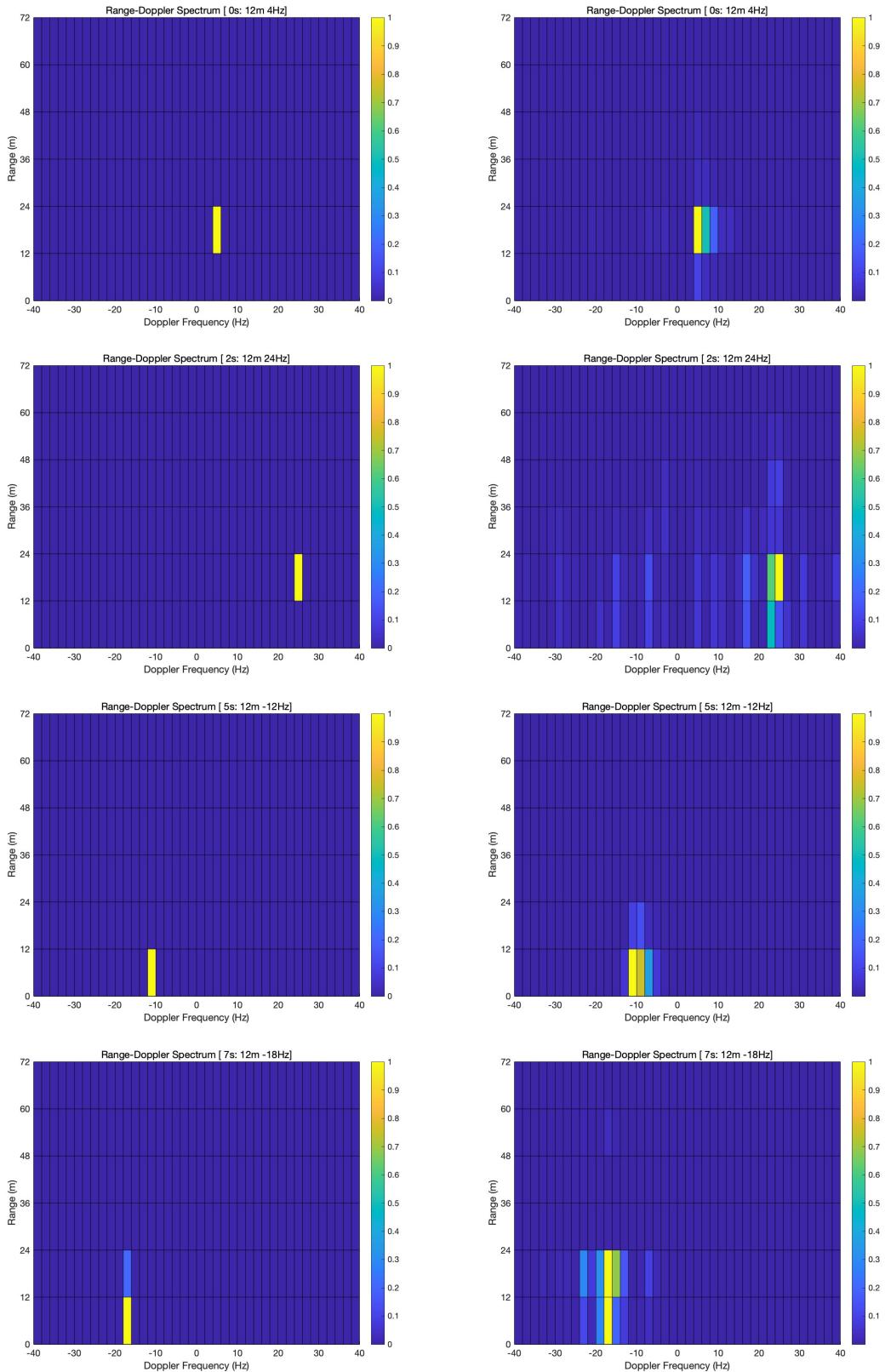


Figure 5. Range Doppler spectrum

也是因为调整对比度，我们才发现了最后几个数据包可能存在的异常，即除了最大值附近的值较大，比较远的地方也有一个极值点。如图 6 所示，如果锐化过大，像左边那一列那样，就无法看出区别异常。

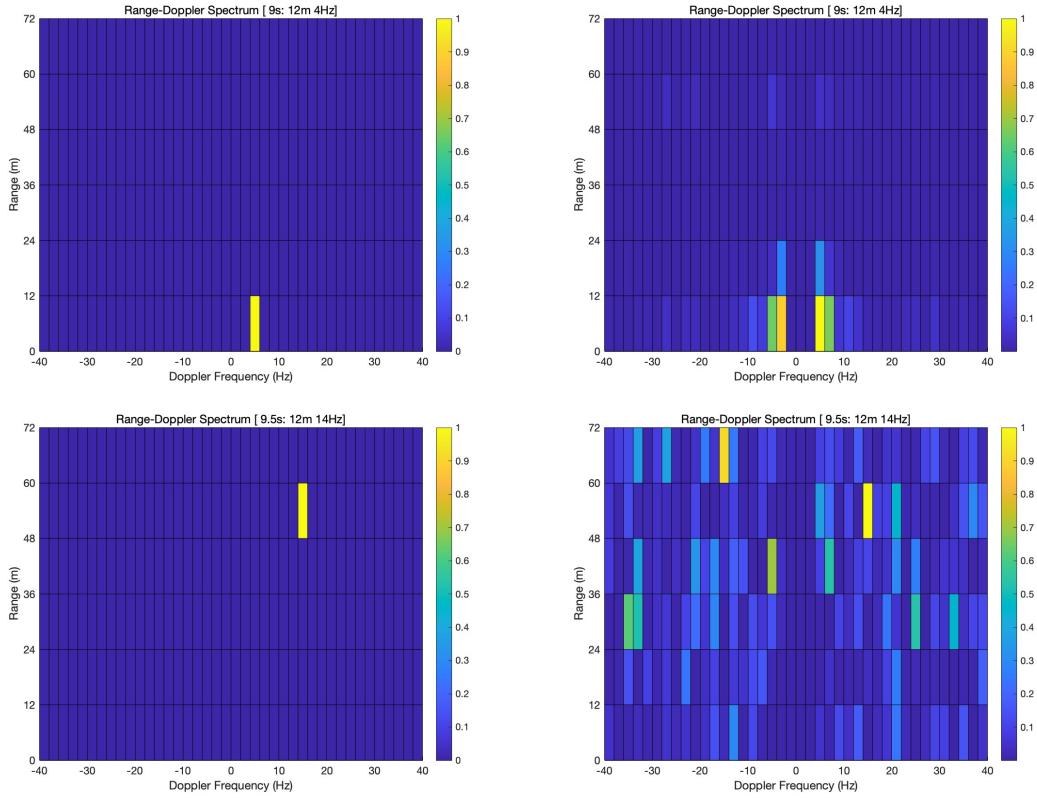


Figure 6. 异常数据包

3. Task3

题目要求画出 Time-Doppler spectrum，只需对 20 个已有的数据包进行 Task2 的处理，然后将最大值所在的行取出拼在一起即可。如图 7 所示，作图是只保留最大值结果，右图是保留了较大值的结果。

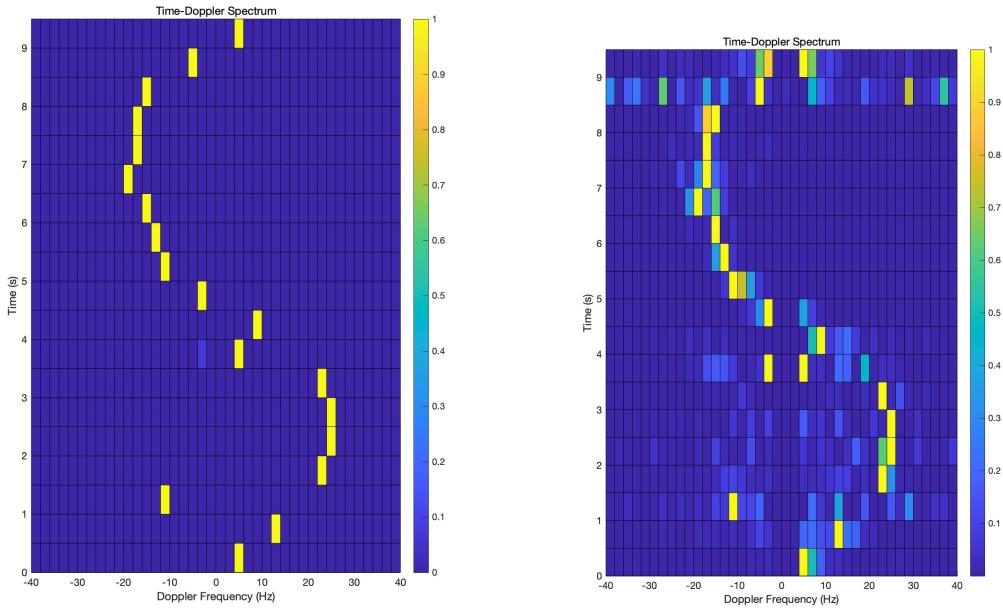


Figure 7. Time-Doppler spectrum

4. 拓展 1：利用 FFT 提高计算速度

我们找到了两种利用 FFT 提高速度的方式。

- 利用时域卷积等于频域相乘的性质，推导如 8 所示。

$$\begin{aligned}
 & \mathcal{F} \left\{ \sum_k (y_{surv}[k] e^{-j\frac{\omega}{2\pi B} k}) \cdot y_{ref}^*[A-k] \right\} \\
 & = Y_{sur}(e^{j(\omega - \frac{\omega}{2\pi B})}) Y_{ref}^*(e^{j\omega})
 \end{aligned}$$

Figure 8. 利用卷积性质

- 观察整体形式与傅立叶变换公式类似，直接将 $y[n]' = y_{ref}[nT_s] y_{sur}^*[nT_s - \tau]$ 看作整体，想得到的某个频率下模糊函数的值就是对 $y[n]'$ 进行傅立叶变换之后的结果，推导如 9 所示。

$$\begin{aligned}
 Cor(\tau, f_0) &= \sum_{n=0}^{N-1} y_{sur}[nT_s] y_{ref}^*[nT_s - \tau] e^{-j2\pi f_0 n T_s} \\
 Cor(e^{j\omega}) &= \sum_{n=0}^{N-1} y[n] e^{j\omega n}
 \end{aligned}$$

Figure 9. 利用与傅立叶变换相似的形式

与 for 循环的计算方法相比，两种利用 FFT 计算得到的计算速度均显著提升，其中，因为利用卷积性质准换的过程中需要用到 2 次 FFT 和 1 次 ifft，因此计算速度不如利用整体形式相似简化的方法块。图 10 是分别用三种方法计算 task2 中 Data1 时用 tic toc 计时得到的程序运行时间。

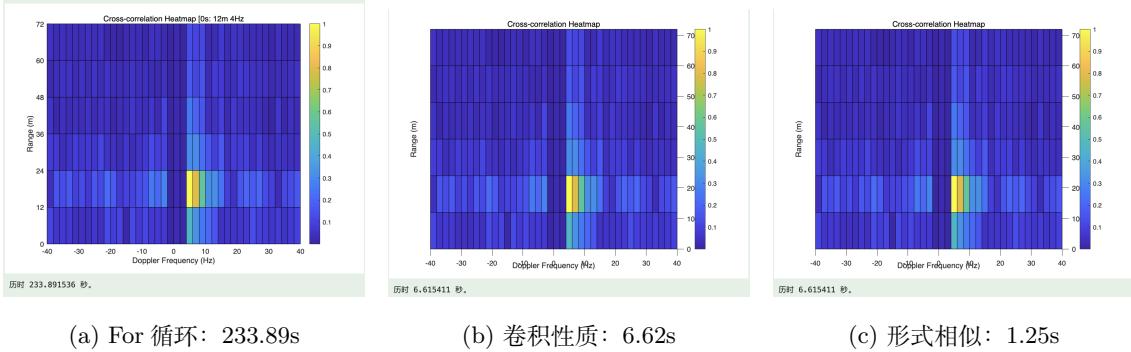


Figure 10. 三种方法计算时间

5. 拓展 2：利用滑窗提高多维度的精度和速度分辨率

减小滑窗步长提高时间精度 如图 11 所示，原本间隔 0.5s 得到一个速度，现在缩小滑窗的步长使得每 0.25s 就得到一个速度，反应了更多更精细的动态信息。

增大滑窗窗长提高速度分辨率 因为我们目标是得到速度的动态变化而不是当前时刻的实时速度大小，所以用越长的时间计算得到的平均速度更加准确。因此如图 11 所示，原本速度是 0.5s 的平均速度，通过增大滑窗得到 1s 的平均速度，提高了速度的分辨率。

增大滑窗窗长提高频率精度 增大窗长除了可以提高速度分辨率，还能增大单次计算的样本量，从而提高频率精度。由公式 $\omega_0 = 2\pi(\Delta f_d)_{\min} = \frac{2\pi}{N\tau_0}$ 可知，提高样本点个数，如我们将 $N \rightarrow 2N$ ，则可以提高频率的精度 $\Delta f_d = 2\text{Hz} \rightarrow \frac{\Delta f_d}{2} = 1\text{Hz}$ 。

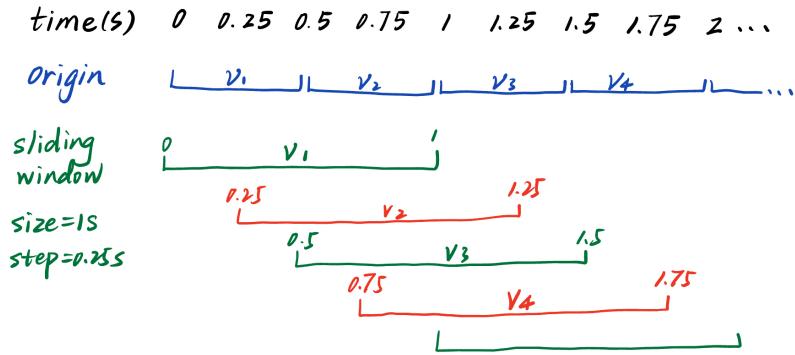


Figure 11. 滑窗原理

通过增大滑窗的窗长、减小滑窗的步长，我们得到了含有更多更精细、精准的速度信息。如图 12 所示，我们采用的窗长为 1s，步长为 0.25s。

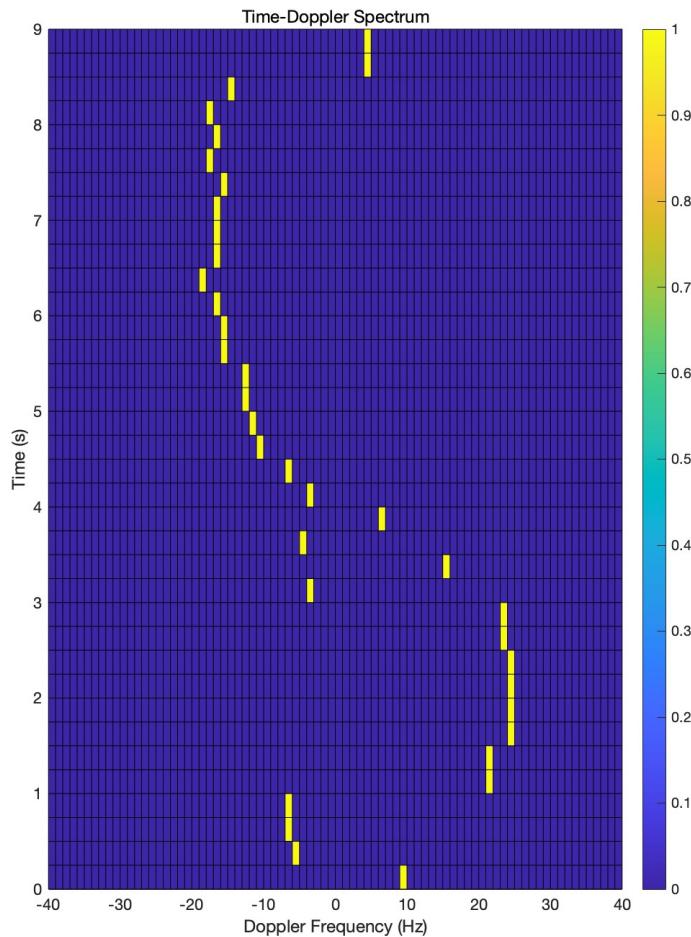


Figure 12. 提高精度和分辨率 Time-Doppler spectrum

如图 13，将不同精度和分辨率全过程的时间速度图像画出，一方面两者大值趋势相同，验证了结果的正确性；另一方面，两者不完全重合，有的地方明显错位，这是因为随着分辨率的提高，速度分辨率改变，更长时间得到的平均速度和短时间得到的不相等，长时间段求出的平均速度包含了部分后面的运动信息导致的。同时也可以明显在精度更高的时候看出 3s 左右的速度由剧烈变化，人运动时速度不会发生巨大突变，这说明可能数据可能有异常。

整体上可以看出，此人先靠近 Detector（表现为速度 >0 ）后远离（表现为速度 <0 ），且由整体斜率看出他先加速、匀速、反向加速、减速、匀速、反向加速。

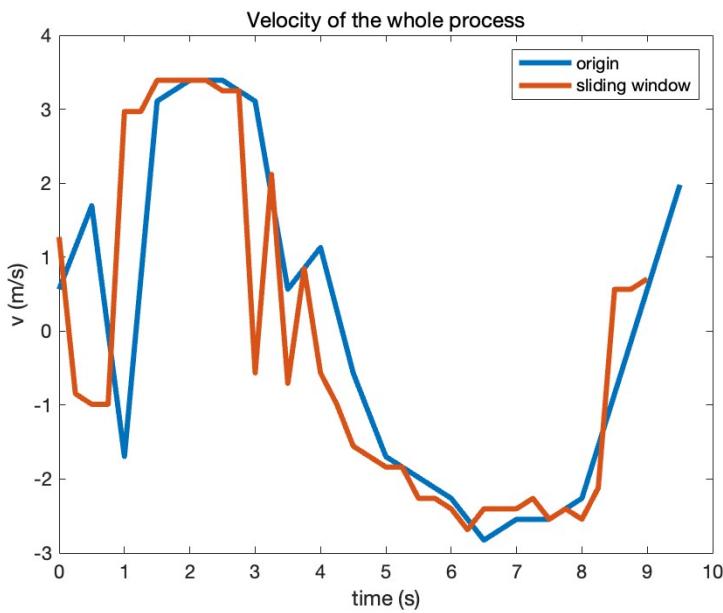


Figure 13. 全过程时间速度图像

Conclusion

1. 收获

- 此次 Project 中我们利用多径效应、处理、分析信号得到了人的运动信息，掌握了利用信号获得目标信息的思想和基本分析技能。
- 同时在不断想办法尝试提高计算速度、程序性能的过程中，我们的逻辑思维、触类旁通的能力也得到了很好的锻炼，对信号和系统的理论知识有了更深入的理解。
- 在处理大量数据的时候，我们掌握了基本批量处理、包装函数的技能，为以后完成更大的项目奠定了良好的基础。

2. 进一步思考

- 仍有疑惑存在：理论上理解，模糊函数最大应该指的是正的最大值（因为从理解上是参考信号的平方），但是实际计算和尝试锐化图像的过程中如果加入 `Cor(Cor<0)=0` 这行代码（作用是将模糊函数求得小于零的值复制为 0）得出的结果会不同。
- 在实际应用中我们还需平衡计算量和精度（步长）、分辨率（窗长）和动态信息的含量。因为步长越小，虽然得到了更精细的变化信息，但是每得到一个速度都需要一次计算，相同窗长下，计算量会大大增大；窗长越长，虽然得到了更高的多普勒频率精度和更高的分辨率，但是相同步长下能得到的窗的个数减少，得到的动态信息量减少。

Score

黎思婕 100 魏雨杉 100 黃姝顏 100

Code

1. Task1

```

1 % 数据加载，初始设置
2 clc,clear;
3 addpath("data");
4 inx_start_time = 1;
5 load(sprintf('data/data_%d.mat',inx_start_time))
6 tau=1/f_s;
7 N=0.5/tau*0.01/0.5; % only 0.01s are used
8 seq_ref=seq_ref(1:end/50);
9 seq_sur=seq_sur(1:end/50);
10 f_ddc=-3e6;
11 bandwidth=9e6;
12
13 %Reference signal图像部分
14 % raw data
15 figure;
16 t=0:0.01:N:0.01-0.01/N;
17 subplot(3,2,1),plot(t,real(seq_ref)),xlabel('Time ...');
18 %s),ylabel('Amplitude'),title('Reference signal (raw)'),ylim([-4e-3,4e-3]);
19 X_ref=fftshift(tau.*fft(seq_ref));
20 w=2*pi/N/tau*(-N/2:N-1-N/2);
21 f=w./(2*pi);
22 subplot(3,2,2),plot(f,20.*log10(abs(X_ref))),xlabel('Frequency (Hz)'),ylabel('Amplitude ...');
23 %dB),title('Spectrum (raw)');
24
25 % DDC data, already know f_ddc
26 w_ddc=2*pi*f_ddc;
27 seq_ref_ddc=seq_ref.*exp(1j*-w_ddc.*t);
28 subplot(3,2,3),plot(t,real(seq_ref_ddc)),xlabel('Time ...');
29 %s),ylabel('Amplitude'),title('Reference signal (after DDC)'),ylim([-4e-3,4e-3]);
30 X_ref_ddc=fftshift(tau.*fft(seq_ref_ddc));
31 subplot(3,2,4), plot(f,20.*log10(abs(X_ref_ddc))), xlabel('Frequency ...');
32 %Hz), ylabel('Amplitude (dB)'), title('Spectrum (after DDC)');
33
34 % LPF
35 [b,a]=butter(20,bandwidth/(f_s/2));

```

```

34 seq_ref_lpf=filter(b,a,seq_ref_ddc);
35 subplot(3,2,5),plot(t,real(seq_ref_lpf)),xlabel('Time ...
(s)'),ylabel('Amplitude'),title('Reference signal (after LPF)'),ylim([-4e-3,4e-3]);
36
37 X_ref_lpf=fftshift(tau.*fft(seq_ref_lpf));
38 subplot(3,2,6), plot(f,20.*log10(abs(X_ref_lpf))), xlabel('Frequency ...
(Hz)'), ylabel('Amplitude (dB)'), title('Spectrum (after LPF)');
39 sgttitle('Reference signal during 0~0.01s')
40 saveas(gcf,'task1_ref.jpg','jpg')
41
42 %Surveillance signal图像部分
43 figure;
44 subplot(3,2,1),plot(t,real(seq_sur)), xlabel('Time ...
(s)'), ylabel('Amplitude'), title('Surveillance signal (raw)'), ylim([-4e-3,4e-3]);
45
46 X_sur=fftshift(tau.*fft(seq_sur));
47 subplot(3,2,2), plot(f,20.*log10(abs(X_sur))), xlabel('Frequency (Hz)'), ylabel('Amplitude ...
(dB)'), title('Spectrum (raw)');
48
49 seq_sur_ddc=seq_sur.*exp(1j*-w_ddc.*t);
50 subplot(3,2,3),plot(t,real(seq_sur_ddc)), xlabel('Time ...
(s)'), ylabel('Amplitude'), title('Surveillance signal (after DDC)'), ylim([-4e-3,4e-3]);
51
52 X_sur_ddc=fftshift(tau.*fft(seq_sur_ddc));
53 subplot(3,2,4), plot(f,20.*log10(abs(X_sur_ddc))), xlabel('Frequency ...
(Hz)'), ylabel('Amplitude (dB)'), title('Spectrum (after DDC)');
54 seq_sur_lpf=filter(b,a,seq_sur_ddc);
55
56 subplot(3,2,5),plot(t,real(seq_sur_lpf)), xlabel('Time ...
(s)'), ylabel('Amplitude'), title('Surveillance signal (after LPF)'), ylim([-4e-3,4e-3]);
57
58 X_sur_lpf=fftshift(tau.*fft(seq_sur_lpf));
59 subplot(3,2,6), plot(f,20.*log10(abs(X_sur_lpf))), xlabel('Frequency ...
(Hz)'), ylabel('Amplitude (dB)'), title('Spectrum (after LPF)');
60
61 sgttitle('Surveillance signal during 0~0.01s')
62 saveas(gcf,'task1_sur.jpg','jpg')

```

2. Task2

1. 数据处理函数：为了方便批量，我们对数据处理过程的代码包装成了函数。

```

1 | function [cur_time,tau,N,seq_ref_lpf,seq_sur_lpf] = preprocessing(data_inx)
2 | % 计算输入信号ddc, lpf之后的输出
3 | % data_indx: the data you want to calculate

```

```

4 addpath("data");
5 load(sprintf('data/data_%d.mat',data_inx))
6 tau=1/f_s;
7 N=0.5/tau;
8 f_ddc=-3e6;
9 bandwidth=9e6;
10 w_ddc=2*pi*f_ddc;
11 t=0:tau:0.5-tau; % 0-N-1
12
13 seq_ref_ddc=seq_ref.*exp(1j*-w_ddc.*t);
14 [b,a]=butter(20,bandwidth/(f_s/2));
15 seq_ref_lpf=filter(b,a,seq_ref_ddc);
16
17 seq_sur_ddc=seq_sur.*exp(1j*-w_ddc.*t);
18 seq_sur_lpf=filter(b,a,seq_sur_ddc);
19 end

```

2. For Loop: For loop 作图部分函数。

```

1 function hotmap_range_f(cur_time,f_d,N,seq_ref_lpf,seq_sur_lpf,tau)
2 %COR_CAL %
3 % range: the range wanted to iterate
4 % f_d: the doppler frequency want to iterate
5 % N: the number of datapoint we want take into calculation
6 n_tau=0:6;
7 range=n_tau*tau*3e8;
8 cor=zeros([length(n_tau),length(f_d)]);
9 for k=n_tau
10     for m=1:length(f_d)
11         for n=1:N
12             if n-k>0
13                 cor(k+1,m)=cor(k+1,m)+ ...
14                     seq_sur_lpf(n)*conj(seq_ref_lpf(n-k))*exp(-1j*2*pi*f_d(m)*(n-1)*tau);
15             end
16         end
17     disp(k);
18 end
19 cor = cor / max(cor(:));
20
21 [col_max, row_idx] = max(cor);
22 [~, col_idx] = max(col_max);
23 row_idx = row_idx(col_idx);
24 figure;
25 surf(f_d,range,abs(cor)),colorbar;
26 view(0,90), xlim([-40,40]), ylim([0,72]), yticks(range), yticklabels(range);

```

```

27 xlabel('Doppler Frequency (Hz)');
28 ylabel('Range (m)');
29 title(['Cross-correlation Heatmap [',num2str(cur_time), 's: ...
29     ',num2str(range(row_idx)), 'm ',num2str(f_d(col_idx)), 'Hz]']);
30 figure1=gcf;
31 saveas(figure1,num2str(cur_time), 'png')
32 end

```

3. 主体程序部分

```

1 clc,clear
2 for m=[1,5,11,15] % 填入想要计算的数据包
3     [cur_time,tau,N,seq_ref_lpf,seq_sur_lpf]=preprocessing(m)
4     f_d=-40:2:40;
5     tic
6     hotmap_range_f(cur_time,f_d,N,seq_ref_lpf,seq_sur_lpf,tau)
7     toc
8 end

```

3. Task3

因为需处理 20 个数据包，我们用了 FFT 来提高计算速度。这里用到的 cor_fft() 函数代码会在后续部分展示。

```

1 clear,clc;
2 time_freq = [];
3 for k = 1:20
4     cor = cor_fft(k);
5     % 取出最大值所在的行序列
6     [M, I] = max(cor(:));
7     [row_idx, ~] = ind2sub(size(cor), I);
8     max_row = cor(row_idx, :);
9     time_freq = cat(1, time_freq, max_row);
10 end
11 save('ts_1.mat','time_freq'); %存储数据方便后续调整作图
12
13 %作图
14 f_d = -40:2:40;
15 time = 0:0.5:10-0.5;
16 [X, Y] = meshgrid(f_d, time);
17 figure;
18 surf(X, Y, abs(time_freq));
19 colorbar;
20 xlabel('Doppler Frequency (Hz)');

```

```

21 ylabel('Time (s)');
22 zlabel('Magnitude');
23 title('Time-Doppler Spectrum');
24 view(0, 90);
25 xlim([-40, 40]);
26 ylim([0, 9.5]);
27 pbaspect([2 3 1]);
28 fig = gcf;
29 fig.Position = [100, 100, 800, 1000];

```

4. 拓展 1：利用 FFT 提高计算速度

为了方便批量操作，我们定义了 Cor_fft() 函数计算不同 τ 和 f_d 得到的模糊函数值。其中代码里有很多需要留意的细节，我们在注释里进行了标注。

1. 利用卷积性质

```

1 function Cor= cor_fft(index)
2 [cur_time,tau,N,seq_ref_lpf,seq_sur_lpf]=preprocessing(index);
3 a_ref=conj(fft(flipud(seq_ref_lpf))); % 行向量，注意一定要先fft变换再求共轭
4 b_sur=fft(seq_sur_lpf);
5 Cor=[];
6 for k=-20:20
7     Cor_fft=a_ref.*circshift(b_sur,-k); % 在频域circ的步长不是fd, 是一个一个的单位
8     Cor_k=ifft(Cor_fft);
9     Cor=cat(2,Cor,Cor_k(1:7).');
10 end
11 Cor = Cor / max(Cor(:));
12 Cor=(Cor).^3; %这里展示的是保留较大值的锐化图像方法
13 end

```

2. 利用与傅立叶变换相似的形式

```

1 function Cor= cor_fft(index)
2 [cur_time,tau,N,seq_ref_lpf,seq_sur_lpf]=preprocessing(index);
3 Cor=[];
4 for k=0:6
5     cor = fftshift(fft(seq_sur_lpf.*circshift(conj(seq_ref_lpf),k)));
6     Cor = cat(1,Cor,cor(N/2-19:N/2+21)); % fftshift翻折少了一点，所以不是完全对称的
7 end
8 Cor = Cor / max(Cor(:));
9 Cor=(Cor).^3;
10 end

```

5. 拓展 2：利用滑窗提高多维度的精度和速度分辨率

- 为了方便，我们将提取最大值所在行的操作与计算模糊函数包装成同一个函数，进一步精简，同时因为频率精度改变，我们也改变了 circshift 的参数。

```

1 function max_row = max_row_extraction(seq_ref_lpf,seq_sur_lpf)
2 a_ref=conj(fft(seq_ref_lpf));
3 b_sur=fft(seq_sur_lpf);
4 Cor=[];
5 for k=-40:40 %频率精度提高
6     Cor_fft=a_ref.*circshift(b_sur,-k);
7     Cor_k=ifft(Cor_fft);
8     Cor=cat(2,Cor,Cor_k(1:7).');
9 end
10 Cor = Cor / max(Cor(:));
11 Cor=(Cor).^1000; % 因为精度高，图像更容易花，所以去了只保留最大值的锐化力度
12
13 % 提取最大值所在行
14 [~,I]=max(Cor(:));
15 [row_idx, ~] = ind2sub(size(Cor), I);
16 max_row = Cor(row_idx, :);
17 end

```

- 为了进行滑窗操作，首先需要将 20 个数据包粘在一起，并一起进行 DDC 和过低通的操作。

```

1 clc,clear;
2 % 粘起来
3 numFiles = 20;
4 dataDir = 'data';
5 addpath(dataDir);
6 load(sprintf('data/data_%d.mat', 1));
7 seqLength = length(seq_sur);
8 data_sur = zeros(seqLength, numFiles);
9 data_ref = zeros(seqLength, numFiles);
10 for k = 1:numFiles
11     load(sprintf('data/data_%d.mat', k));
12     data_sur(:, k) = seq_sur;
13     data_ref(:, k) = seq_ref;
14 end
15
16 % 给定数据处理需要的参数
17 tau = 1 / f_s;
18 N = 0.5 / tau;
19 f_ddc = -3e6;
20 bandwidth = 9e6;

```

```

21 w_ddc = 2 * pi * f_ddc;
22 t = 0:tau:0.5-tau; % 0-N-1
23
24 % DDC和过低通
25 seq_ref_ddc = data_ref .* exp(1j * -w_ddc .* t');
26 [b, a] = butter(20, bandwidth / (f_s / 2));
27 data_ref_lpf = filter(b, a, seq_ref_ddc);
28
29 seq_sur_ddc = data_sur .* exp(1j * -w_ddc .* t');
30 data_sur_lpf = filter(b, a, seq_sur_ddc);
31 save('data_tot.mat', 'data_ref_lpf', 'data_sur_lpf', '-v7.3');

```

3. 滑窗并计算出模糊函数的所有数据，因为计算量太大我们进行了分批计算、储存、处理。

```

1 clc;
2 clear;
3 load('data_tot.mat', 'data_ref_lpf', 'data_sur_lpf');
4
5 % 设置滑窗参数
6 N = 12500000;
7 window_size = 2 * N;
8 stepsize = N / 2;
9 numFiles = 20;
10 dataLength = N * numFiles;
11 overlap = window_size - stepsize;
12 numWindows = floor((dataLength - window_size) / stepsize) + 1;
13
14 % 分批次计算
15 batchSize = 10;
16 time_freq_sw = [];
17 for batchStart = 1:batchSize:numWindows
18     batchEnd = min(batchStart + batchSize - 1, numWindows);
19     batchIndices = batchStart:batchEnd;
20     batchResult = []; %% 临时存储当前批次的结果
21
22     % 处理当前批次的滑窗
23     for i = 1:length(batchIndices)
24         k = batchIndices(i);
25         start_idx = (k - 1) * stepsize + 1;
26         end_idx = start_idx + window_size - 1;
27         sur_window = data_sur_lpf(start_idx:end_idx);
28         ref_window = data_ref_lpf(start_idx:end_idx);
29         max_row = max_row_extraction(ref_window.', sur_window.');
30         batchResult(i, :) = max_row;
31     end
32     time_freq_sw(batchIndices, :) = batchResult; % 储存当前批次结果

```

```

33     batchStart
34
35     end
36     save('ts.mat','time_freq_sw')

```

4. 绘制结果

```

1 f_d = -40:1:40;
2 time = 0:0.25:10-1;
3 [X, Y] = meshgrid(f_d, time);
4 figure;
5 surf(X, Y, abs(time_freq_sw));
6 colorbar;
7 xlabel('Doppler Frequency (Hz)');
8 ylabel('Time (s)');
9 zlabel('Magnitude');
10 title('Time-Doppler Spectrum');
11 view(0, 90);
12 xlim([-40, 40]);
13 ylim([0,9]);
14 pbaspect([2 3 1]);
15 fig = gcf;
16 fig.Position = [100, 100, 800, 1000];

```

6. 作出全过程不同精度速度时间图像

```

1 clc,clear
2 load('./data/data_1.mat').%得到计算需要的常量：载波中心频率
3 load("ts_1.mat"); %task3得到的结果
4 %求出模糊函数最大值所对应的频率
5 [M,I] = max(time_freq,[],2,"linear");
6 [max_row, max_col] = ind2sub(size(time_freq), I);
7 f_d = -40:2:40;
8 max_freq = f_d(max_col);
9 %根据公式计算出多普勒频率对应的速度
10 velocity_1 = max_freq.*3e8/f_c;
11 figure;
12 plot(0:0.5:9.5,velocity_1,LineWidth=3);
13 hold on;
14
15 load("ts.mat"); %利用滑窗求出具有更高精度和分辨率的结果
16 [M,I] = max(time_freq_sw,[],2,"linear");
17 [max_row, max_col] = ind2sub(size(time_freq_sw), I);
18 f_d = -40:1:40;
19 max_freq = f_d(max_col);

```

```
20 velocity_2 = max_freq.*3e8/f_c;
21 plot(0:0.25:9,velocity_2,LineWidth=3),title('Velocity of the whole process'),xlabel('time ...
(s)'),ylabel('v (m/s)');
22 legend('origin','sliding window')
23 hold off;
```