易懂的雷达信号处理

面向学生与工程师

第八章：完整处理流程

作者：唐承乾

版本：社区版 V1

官方仓库：https://github.com/apple-art/easy-radar-tutorial

目录

前面几章分别介绍了距离测量、速度测量和目标检测的基本原理。第 4 章给出了匹配滤波公式 $h(t) = s^*(-t)$ 和距离关系 $R = c\tau/2$；第 5 章给出了多普勒频率 $f_d = 2v/\lambda$ 和脉冲间相位变化；第 6 章给出了阈值判决 $\delta(x)$。这些公式各自解决一个问题，但实际雷达需要把它们串成一条完整的处理链。

本章不再引入新的基本原理，把第 4-6 章的公式接成可运行的 MATLAB 程序。对脉冲雷达，处理顺序可以直接写成

$$\text{回波数据} \rightarrow \text{距离压缩} \rightarrow \text{多普勒处理} \rightarrow \text{检测} \rightarrow \text{目标信息}$$

其中，距离压缩对应第 4 章的匹配滤波，沿快时间方向工作；多普勒处理对应第 5 章的速度测量，沿慢时间方向工作；检测对应第 6 章的阈值判决，作用在距离-速度图上。

本章建议边读边对照 MATLAB 代码。若你希望直接运行各节示例，可参考以下脚本：

- `ch08_minimal_pulse_radar_demo.m`
- `ch08_range_doppler_demo.m`
- `ch08_radar_detection_demo.m`
- `ch08_full_processing_chain_demo.m`

阅读顺序也建议和章节顺序保持一致：先看 8.1 的最小示例，再看 8.2 的距离-速度处理，接着看 8.3 的检测，最后再回到 8.4 把整条处理链连起来。

8.1 最小可运行示例

这一节只保留距离处理链中的最小部分：生成发射信号，构造单目标回波，做匹配滤波，再由主峰位置换算目标距离。这样可以先把第 4 章的时延公式和匹配滤波公式在 MATLAB 中对应起来。

设目标真实距离为 $R_0 = 6$ km，目标静止不动。第 4 章给出的距离关系式为

$$\tau_0 = \frac{2R_0}{c}, \qquad R = \frac{c\tau}{2}$$

其中 τ_0 为目标回波的双程传播时延，c 为光速。发射信号采用第 4 章介绍的 LFM 脉冲。为了与本节 MATLAB 程序一致，这里使用复包络基带形式：

$$s(t) = \exp\left(j\pi k t^2\right), \qquad 0 \le t < T_p$$

$$k = \frac{B}{T_p}$$

其中 B 为 LFM 带宽，T_p 为脉冲宽度，k 为调频斜率。接收回波可写成

$$r(t) = s(t - \tau_0) + w(t)$$

其中 $w(t)$ 表示噪声。匹配滤波仍采用第 4 章的公式

$$h(t) = s^*(-t), \qquad y(t) = r(t) * h(t)$$

其中 $s^*(t)$ 表示 $s(t)$ 的共轭，$*$ 表示卷积运算。

8.1.1 公式与程序变量对应

程序里的主要符号、变量和物理意义见表 8-1。

公式/符号	MATLAB 变量或表达式	含义
c	`c`	光速 (m/s)
B	`B`	LFM 带宽 (Hz)
T_p	`Tp`	脉冲宽度 (s)
f_s	`fs`	采样率 (Hz)
R_0	`R0`	目标真实距离 (m)
$k = B/T_p$	`k = B / Tp`	调频斜率 (Hz/s)
$N_{\text{fast}} = T_p f_s$	`N_fast = round(Tp * fs)`	一个脉冲内的采样点数
$s(t)$	`tx`	发射基带 LFM 脉冲
$\tau_0 = 2R_0/c$	`delay_time = 2 * R0 / c`	双程传播时延 (s)
$n_0 = \tau_0 f_s$	`delay_samples = round(delay_time * fs)`	时延对应的采样点数
$r(t) = s(t-\tau_0) + w(t)$	`rx`	延迟回波加噪声
$h(t) = s^*(-t)$	`mf = conj(fliplr(tx))`	匹配滤波器
$y(t) = r(t) * h(t)$	`range_profile = conv(rx, mf, 'same')`	匹配滤波输出
$\hat{R}$	`R_est`	距离估计值 (m)

下面给出完整 MATLAB 程序。

如果你想直接运行本节代码，可参考 `ch08_minimal_pulse_radar_demo.m`。

```
% 第八章示例 1：单目标距离处理的最小可运行示例
clear; close all; clc;

% 基本参数：本例采用复包络基带模型，fc 只保留物理背景
c = 3e8;            % 光速 (m/s)
fc = 10e9;          % 载频 (Hz)
B = 10e6;           % LFM 带宽 (Hz)
Tp = 20e-6;         % 脉冲宽度 (s)
```

```
fs = 20e6;          % 采样率 (Hz)
R0 = 6000;          % 目标真实距离 (m)

% 生成基带 LFM 脉冲，k = B / Tp
N_fast = round(Tp * fs);
t_fast = (0:N_fast-1) / fs;
k = B / Tp;
tx = exp(1j * pi * k * t_fast.^2);  % 对应公式 s(t)

% 目标距离 -> 双程时延 -> 离散延迟样点
delay_time = 2 * R0 / c;                        % 对应公式    = 2R /c
delay_samples = round(delay_time * fs);
fast_len = N_fast + delay_samples + 200;

% 构造单目标回波：把发射脉冲平移到延迟位置，再叠加复高斯噪声
rx = zeros(1, fast_len);
rx(delay_samples + (1:N_fast)) = tx;            % 对应公式 r(t) = s(t- )
rx = rx + 0.05 * (randn(size(rx)) + 1j * randn(size(rx)));  % 加噪声 w(t)

% 匹配滤波等价于与发射信号做相关，主峰对应目标回波时延
mf = conj(fliplr(tx));                          % 对应公式 h(t) = s*(-t)
range_profile = conv(rx, mf, 'same');           % 对应公式 y(t) = r(t)*h(t)

% conv(...,'same') 保留中心段，主峰索引需先扣除匹配滤波器中心偏移
[~, idx_peak] = max(abs(range_profile));
delay_est_samples = idx_peak - 1 - floor(N_fast / 2);

% 先估计时延，再按 R = c *  / 2 换算距离
R_est = c * delay_est_samples / (2 * fs);       % 对应公式 R = c /2
range_axis = ((0:fast_len-1) - floor(N_fast / 2)) * c / (2 * fs);

fprintf('真实距离：%.2f km\n', R0 / 1e3);
fprintf('估计距离：%.2f km\n', R_est / 1e3);

figure('Position', [100, 100, 1000, 600]);
subplot(2,1,1);
plot(real(rx), 'LineWidth', 1.0);
grid on;
title('接收回波（实部）');
xlabel('采样点'); ylabel('幅度');

subplot(2,1,2);
plot(range_axis / 1e3, abs(range_profile), 'LineWidth', 1.5); hold on;
plot(R_est / 1e3, abs(range_profile(idx_peak)), 'ro', 'MarkerSize', 8, 'LineWidth',
↪  1.5);
grid on;
```

```
title('匹配滤波后的距离像');
xlabel('距离 (km)'); ylabel('幅度');
legend('距离像', '峰值');
```

8.1.2 conv(..., 'same') 的索引修正

上面这段程序中，tx 对应发射信号 $s(t)$，rx 对应回波 $r(t)$，mf 对应匹配滤波器 $h(t)$，range_profile 对应匹配滤波输出 $y(t)$。其中

```
range_profile = conv(rx, mf, 'same');
```

就是卷积公式 $y(t) = r(t) * h(t)$ 的离散实现。由于这里使用 conv(..., 'same')，输出序列保留的是中心部分，因此主峰索引不能直接当作时延样点使用，而应写成

$$n_\tau = \mathrm{idx}_{\mathrm{peak}} - 1 - \left\lfloor \frac{N_{\mathrm{fast}}}{2} \right\rfloor$$

$$\hat{\tau} = \frac{n_\tau}{f_s}, \qquad \hat{R} = \frac{c\hat{\tau}}{2}$$

程序中的 delay_est_samples 和 R_est 正对应这两步修正。如果不扣除这部分中心偏移，距离估计会系统性偏大。

图 8-2 给出了程序输出。上图中的接收信号已经叠加噪声，目标位置并不直接可见；下图经过匹配滤波后，目标在距离轴上形成明显主峰。这个主峰的横坐标，就是目标距离的估计值。

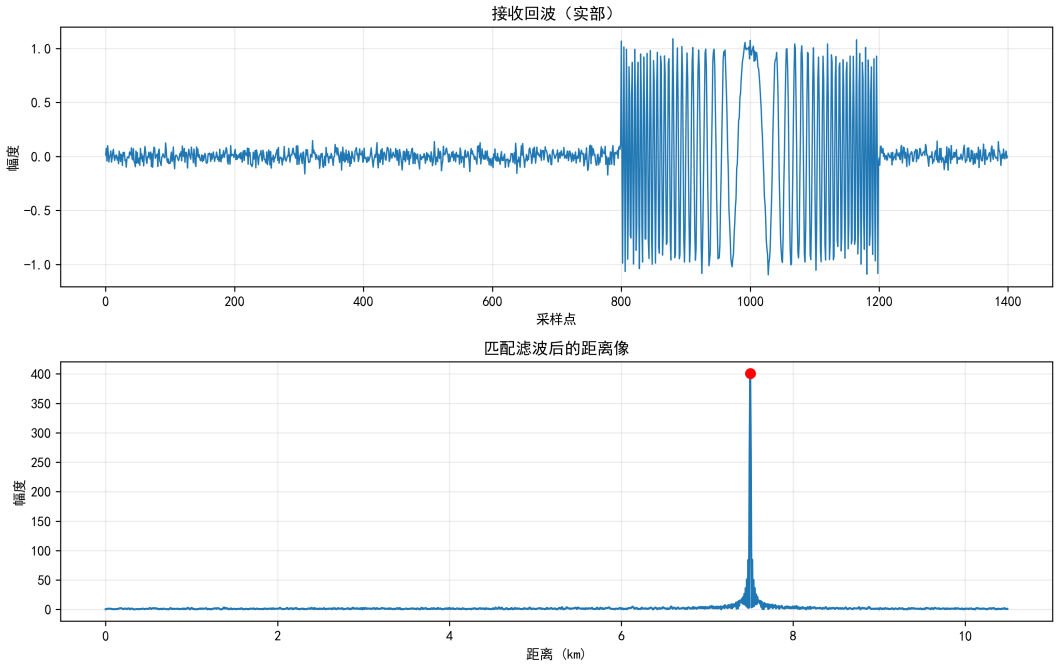

图 1: 最小可运行示例中的回波与距离峰

8.1.3 匹配滤波与其他实现方式

虽然匹配滤波本质是相关运算，但它在加性高斯噪声下能达到最优信噪比增益。与其他方法的对比如下：

方法	实现	优点	缺点
匹配滤波	`conv(rx, conj(fliplr(tx)), 'same')`	最大化信噪比，理论最优	需要精确知道发射波形
频域乘法	`ifft(fft(rx) .* conj(fft(tx)))`	长序列计算快	需要补零对齐，边界效应
直接互相关	`xcorr(rx, tx)`	概念直观	输出长度不同，需要额外处理

本例选择时域卷积是因为代码直观，且序列长度适中。若 `fast_len` 超过 10000 点，频域方法会更高效。

8.1.4 参数变化对结果的影响

带宽 B 的影响：距离分辨率 $\Delta R = c/(2B)$。

- B = 5 MHz → 分辨率 30 m，主瓣宽度加倍
- B = 10 MHz → 分辨率 15 m（本例设置）
- B = 20 MHz → 分辨率 7.5 m，但需要更高采样率

采样率 fs 的影响：必须满足 $f_s \geq 2B$。

- fs = B → 欠采样，频谱混叠，匹配滤波失效
- fs = 2B → 奈奎斯特临界，实际工程需留余量
- fs = 5B → 过采样，计算量增加但精度提升有限

本例 fs = 20 MHz，B = 10 MHz，满足 fs = 2B，刚好达到采样定理要求。

8.1.5 边界条件与失效模式

情况 1：目标距离超出采样窗口若 `delay_samples > fast_len`，回波完全落在采样窗外，`range_profile` 无明显峰值。解决方法是增加 `fast_len`。

情况 2：信噪比过低当噪声标准差从 0.05 增加到 0.5 时，主峰可能被噪声淹没。此时需要增加脉冲积累（下一节内容）或提高发射功率。

情况 3：多目标距离接近若两目标间隔 $< \mathrm{c}/(2\mathrm{B}) = 15$ m，主瓣重叠，单脉冲无法分辨。这是距离分辨率的物理极限。

这一节得到的是最简单的单目标距离处理结果。下一节在此基础上加入脉冲序列，把第 5

章的速度处理也并入同一套 MATLAB 框架中。

8.2 MATLAB 中的距离与速度处理

上一节只处理了单目标、单脉冲情形，得到的是一条距离像。要进一步估计目标速度，就必须把多个脉冲联合起来处理。本节在 8.1 的代码基础上修改，主要改动是： 回波从 1D 向量改成 2D 矩阵； 距离压缩加循环处理每个脉冲； 增加慢时间 FFT 得到距离-速度图。

8.2.1 回波矩阵的组织方式

接收数据不再是一维波形，而是一个二维矩阵。记原始回波矩阵为 $X(m,n)$，其中 m 表示脉冲序号（慢时间），n 表示脉冲内的快时间采样点。沿快时间方向处理，得到距离像；沿慢时间方向处理，得到多普勒谱。

这种二维处理方式正好把前两章的公式接在一起。对第 m 个脉冲，距离压缩仍然对应第 4 章的匹配滤波

$$y_m(t) = r_m(t) * h(t), \qquad h(t) = s^*(-t)$$

程序中写成

```
for m = 1:Npulse
    range_data(m, :) = conv(rx_matrix(m, :), mf, 'same');
end
```

其中 `rx_matrix` 对应回波矩阵 $X(m,n)$，`mf` 对应匹配滤波器 $h(t)$，`range_data` 对应每个脉冲做完距离压缩后的结果。由于 `conv(..., 'same')` 保留的是中心段，程序需要扣除匹配滤波器带来的中心偏移，距离轴应写成

```
group_delay = floor(length(tx) / 2);
range_axis = ((0:Nfast-1) - group_delay) * c / (2 * fs);
```

这样 `range_data` 中主峰所在的列索引，才与第 4 章的关系式 $R = c\Delta t/2$ 一致。

速度处理对应第 5 章的多普勒关系

$$f_d = \frac{2v}{\lambda}, \qquad v = \frac{\lambda f_d}{2}$$

在脉冲序列模型中，这一关系体现为脉冲间相位项

$$\exp\bigl(j2\pi f_d (m-1) T_r\bigr)$$

其中 T_r 为脉冲重复间隔，λ 为波长。距离压缩完成后，对每一个距离单元沿慢时间方向做傅里叶变换，就得到距离-速度图：

```
rd_map = fftshift(fft(range_data, [], 1), 1);
```

这一步的输入是 `range_data`，输出是 `rd_map`。`rd_map` 的列仍然对应距离单元，行则对应多普勒频率，再通过

$$\text{vel\_axis} = \frac{\lambda}{2} \cdot \text{doppler\_freq}$$

换成速度轴。

本节示例采用两个目标：目标 A 位于 6 km、速度 30 m/s；目标 B 位于 9 km、速度 -20 m/s。程序中的三个核心数组如下。

数学对象	MATLAB 变量	含义
$X(m,n)$	`rx_matrix`	原始回波矩阵（行 = 脉冲，列 = 采样点）
$y_m(t)$	`range_data`	各脉冲距离压缩结果
$Y(f_d,R)$	`rd_map`	距离-速度图

8.2.2 从矩阵索引到物理坐标

距离-速度图的横纵坐标需要从矩阵索引转回物理单位。距离轴已经在上面给出，速度轴的构造如下：

```
doppler_freq = (-Npulse/2:Npulse/2-1) * PRF / Npulse;
vel_axis = doppler_freq * lambda / 2;
```

其中 `PRF` 为脉冲重复频率，`Npulse` 为脉冲数量。`fftshift` 把零频移到中心，因此多普勒频率范围是 $[-\text{PRF}/2, \text{PRF}/2)$。

下面给出本节完整 MATLAB 程序。

如果你想直接运行本节代码，可参考 `ch08_range_doppler_demo.m`。

```
% 第八章示例 2：距离处理与多普勒处理
clear; close all; clc;

% 基本参数：采用复包络基带模型
c = 3e8;
fc = 10e9;
lambda = c / fc;
B = 10e6;
Tp = 20e-6;
fs = 20e6;
PRF = 2e3;
```

```
Tr = 1 / PRF;
Npulse = 32;
Nfast = 1024;

% 两个目标的距离、速度和幅度
range_targets = [6000, 9000];
vel_targets = [30, -20];
amp_targets = [1.0, 0.75];

% 发射信号 s(t)，其中 k = B / Tp
k = B / Tp;
tx_t = (0:round(Tp * fs)-1) / fs;
tx = exp(1j * pi * k * tx_t.^2);              % 对应公式 s(t)
mf = conj(fliplr(tx));                          % 对应公式 h(t) = s*(-t)
fast_time = (0:Nfast-1) / fs;

% 回波矩阵：距离决定时延，速度决定脉冲间相位变化
rx_matrix = zeros(Npulse, Nfast);
for m = 1:Npulse
    pulse_data = zeros(1, Nfast);
    for i = 1:length(range_targets)
        tau = 2 * range_targets(i) / c;         % 对应公式  = 2R/c
        fd = 2 * vel_targets(i) / lambda;       % 对应公式 f_d = 2v/
        delayed_t = fast_time - tau;
        valid = delayed_t >= 0 & delayed_t <= Tp;
        echo = zeros(1, Nfast);
        echo(valid) = amp_targets(i) * exp(1j * pi * k * delayed_t(valid).^2) ...
                    .* exp(1j * 2*pi * fd * (m-1) * Tr);  % 对应公式
                    ↪  exp(j2 f_d(m-1)T_r)
        pulse_data = pulse_data + echo;
    end
    rx_matrix(m, :) = pulse_data + 0.08 * (randn(1, Nfast) + 1j * randn(1, Nfast));
end

% 先沿快时间做匹配滤波，得到距离压缩结果
range_data = zeros(size(rx_matrix));
for m = 1:Npulse
    range_data(m, :) = conv(rx_matrix(m, :), mf, 'same');  % 对应公式 y(t) = r(t)*h(t)
end

group_delay = floor(length(tx) / 2);
range_axis = ((0:Nfast-1) - group_delay) * c / (2 * fs);

% 再沿慢时间做 FFT，得到距离-速度图
rd_map = fftshift(fft(range_data, [], 1), 1);
rd_power = abs(rd_map);
```

```
rd_power = rd_power / max(rd_power(:));

doppler_freq = (-Npulse/2:Npulse/2-1) * PRF / Npulse;
vel_axis = doppler_freq * lambda / 2;           % 对应公式 v = f_d/2

figure('Position', [120, 120, 1100, 700]);
subplot(2,1,1);
plot(range_axis / 1e3, abs(range_data(1, :)), 'LineWidth', 1.5);
grid on;
title('第一个脉冲的距离像');
xlabel('距离 (km)'); ylabel('幅度');

subplot(2,1,2);
imagesc(range_axis / 1e3, vel_axis, rd_power);
axis xy; colorbar;
title('距离-速度图');
xlabel('距离 (km)'); ylabel('速度 (m/s)');
```

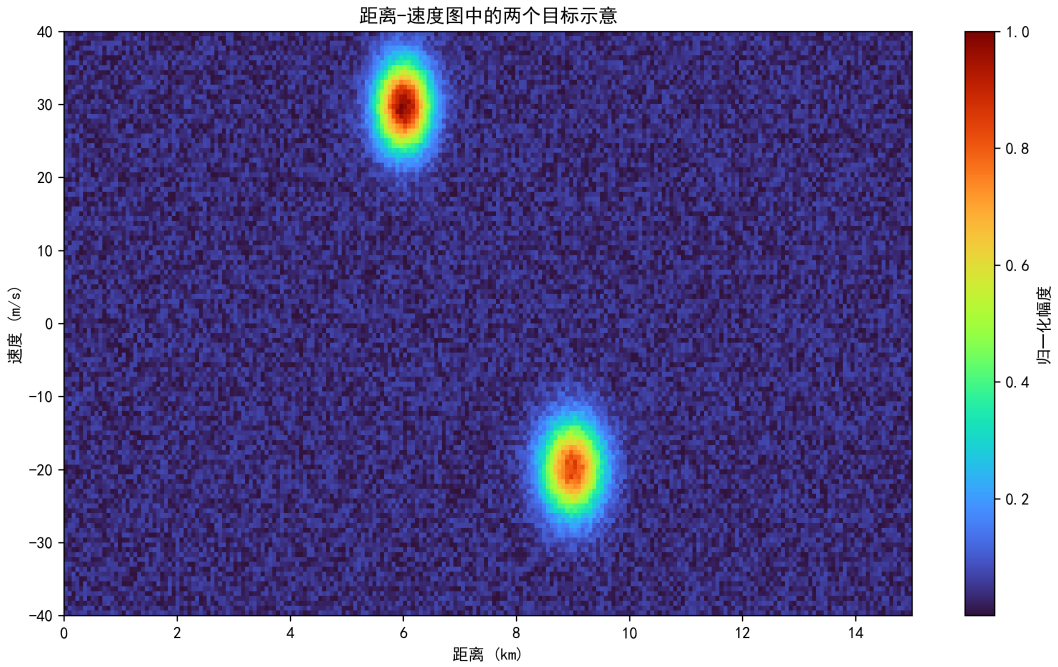

图 2: 距离-速度图示意

图 8-3 中的亮点位置同时反映了距离与速度：横坐标对应目标距离，纵坐标对应目标速度。两个亮点分处不同距离和不同速度，说明两个目标具有不同的传播时延和不同的多普勒频率。

8.2.3 fftshift 的作用

FFT 默认输出顺序是 [0, f_max] 然后 [-f_max, 0)，但距离-速度图习惯把零速度放中间。不用 `fftshift` 会导致速度轴错位：

```
% 不用 fftshift：负速度在下半部分
rd_map_raw = fft(range_data, [], 1);
vel_axis_raw = (0:Npulse-1) * PRF / Npulse;  % 0 到 PRF

% 用 fftshift：零速度居中
rd_map = fftshift(fft(range_data, [], 1), 1);
vel_axis = (-Npulse/2:Npulse/2-1) * PRF / Npulse;  % -PRF/2 到 PRF/2
```

`fftshift` 的第二个参数 1 表示沿第一维(行,即慢时间)移位。如果写成 `fftshift(fft(...), 2)` 会错误地移位距离维。

8.2.4 速度模糊与 PRF 的权衡

最大不模糊速度由 PRF 决定：

$$v_{\max} = \frac{\lambda \cdot \mathrm{PRF}}{4}$$

本例中 $\lambda = 0.03$ m，PRF $= 2$ kHz，因此 $v_{\max} = 15$ m/s。

PRF	$v_{\max}$	优点	缺点
1 kHz	7.5 m/s	最大探测距离远	速度范围窄，易模糊
2 kHz	15 m/s	本例设置	折中方案
10 kHz	75 m/s	速度范围宽	最大探测距离缩短

若目标速度 30 m/s 但 PRF $= 2$ kHz，实际测得速度会折叠到 [-15, 15) m/s 范围内，表现为 30 - 30 = 0 m/s（静止）或其他模糊值。本例中目标 A 速度 30 m/s 已经超出不模糊范围，实际应用需要提高 PRF 或使用解模糊算法。

8.2.5 矩阵维度与坐标轴对应

MATLAB 中 `imagesc(X, Y, Z)` 要求 Z 的行对应 Y，列对应 X。这与直觉相反，容易出错：

```
% 错误：行列颠倒
imagesc(vel_axis, range_axis, rd_power);  % 速度变成横轴

% 正确：速度是纵轴
imagesc(range_axis, vel_axis, rd_power);
axis xy;  % 必须加这行，否则纵轴倒置
```

如果不加 `axis xy`，MATLAB 默认图像坐标系（原点在左上角），速度轴会上下颠倒。

下一节在此基础上增加检测模块，把距离-速度图上的响应变成目标列表。

8.3 检测与结果解释

经过距离压缩和多普勒处理之后，程序已经得到距离-速度图。但距离-速度图本身还只是幅度分布，不能直接作为目标列表输出。本节在 8.2 的代码基础上增加检测模块，把图上的响应分成”目标”与”非目标”两类。

设距离-速度图的功率或归一化幅度记为 x，第 6 章的判决可写为

$$\delta(x)=\begin{cases}1, & x>T\\ 0, & x\le T\end{cases}$$

其中 T 为检测阈值。在本章的最简实现中，先使用固定阈值。若 `rd_power` 表示归一化后的距离-速度图，则 MATLAB 程序写成

```
threshold = 0.35;
det_mask = rd_power > threshold;
```

这里 `threshold` 对应阈值 T，`det_mask` 对应判决结果 $\delta(x)$ 在整张距离-速度图上的二维展开。也就是说，`det_mask` 中值为 1 的位置表示通过检测，值为 0 的位置表示未通过检测。这样的写法与第 6 章一维距离像上的判决完全一致，只是这里的判决对象由一个采样点扩展成了二维平面上的每一个网格点。

固定阈值的优点是结构简单，便于先把整条处理链跑通；缺点也同样直接：当背景起伏不均匀、强目标旁瓣较高或者噪声水平变化明显时，统一阈值往往不能兼顾整张图。第 6 章介绍的 CFAR 正是为了解决这一类问题。本章先保留固定阈值，只为了把检测放回完整处理流程中。

8.3.1 从检测掩码到目标参数

检测后的第二步，是把图上的行列索引重新换回物理量。若 `row_idx` 和 `col_idx` 分别表示检测点所在的速度维索引和距离维索引，则

```
[row_idx, col_idx] = find(det_mask);
range_est = range_axis(col_idx);
vel_est = vel_axis(row_idx);
```

其中 `range_axis` 对应距离坐标轴，`vel_axis` 对应速度坐标轴，`range_est` 与 `vel_est` 则是最终输出的距离和速度估计值。到这一步，前面得到的距离-速度图才真正转化为可以汇报的目标结果。

数学对象	MATLAB 变量	含义
T	`threshold`	检测阈值
$\delta(x)$	`det_mask`	检测判决结果
距离索引	`col_idx`	通过检测的距离单元位置
速度索引	`row_idx`	通过检测的速度单元位置
$\hat{R}$	`range_est`	检测点对应的距离估计 (m)
$\hat{v}$	`vel_est`	检测点对应的速度估计 (m/s)

下面给出本节完整 MATLAB 程序。这是在 8.2 的完整代码基础上增加检测部分，前面的处理流程完全不变。

如果你想直接运行本节代码，可参考 `ch08_radar_detection_demo.m`。

```
% 第八章示例 3：加入检测模块
clear; close all; clc;

% 基本参数：采用复包络基带模型
c = 3e8;
fc = 10e9;
lambda = c / fc;
B = 10e6;
Tp = 20e-6;
fs = 20e6;
PRF = 2e3;
Tr = 1 / PRF;
Npulse = 32;
Nfast = 1024;

% 两个目标的距离、速度和幅度
range_targets = [6000, 9000];
vel_targets = [30, -20];
amp_targets = [1.0, 0.75];

% 发射信号 s(t)，其中 k = B / Tp
k = B / Tp;
tx_t = (0:round(Tp * fs)-1) / fs;
tx = exp(1j * pi * k * tx_t.^2);
mf = conj(fliplr(tx));
fast_time = (0:Nfast-1) / fs;

% 回波矩阵：距离决定时延，速度决定脉冲间相位变化
rx_matrix = zeros(Npulse, Nfast);
```

```
for m = 1:Npulse
    pulse_data = zeros(1, Nfast);
    for i = 1:length(range_targets)
        tau = 2 * range_targets(i) / c;
        fd = 2 * vel_targets(i) / lambda;
        delayed_t = fast_time - tau;
        valid = delayed_t >= 0 & delayed_t <= Tp;
        echo = zeros(1, Nfast);
        echo(valid) = amp_targets(i) * exp(1j * pi * k * delayed_t(valid).^2) ...
                    .* exp(1j * 2*pi * fd * (m-1) * Tr);
        pulse_data = pulse_data + echo;
    end
    rx_matrix(m, :) = pulse_data + 0.08 * (randn(1, Nfast) + 1j * randn(1, Nfast));
end

% 先沿快时间做匹配滤波，得到距离压缩结果
range_data = zeros(size(rx_matrix));
for m = 1:Npulse
    range_data(m, :) = conv(rx_matrix(m, :), mf, 'same');
end

group_delay = floor(length(tx) / 2);
range_axis = ((0:Nfast-1) - group_delay) * c / (2 * fs);

% 再沿慢时间做 FFT，得到距离-速度图
rd_map = fftshift(fft(range_data, [], 1), 1);
rd_power = abs(rd_map);
rd_power = rd_power / max(rd_power(:));

doppler_freq = (-Npulse/2:Npulse/2-1) * PRF / Npulse;
vel_axis = doppler_freq * lambda / 2;

% 固定阈值检测（对应公式 (x)）
threshold = 0.35;
det_mask = rd_power > threshold;                % 对应公式 (x) = 1 if x > T
[row_idx, col_idx] = find(det_mask);
range_est = range_axis(col_idx);                % 对应公式 R
vel_est = vel_axis(row_idx);                    % 对应公式 v̂

figure('Position', [100, 100, 1200, 500]);
subplot(1,2,1);
imagesc(range_axis / 1e3, vel_axis, rd_power);
axis xy; colorbar;
title('距离-速度图');
xlabel('距离 (km)'); ylabel('速度 (m/s)');
```

```
subplot(1,2,2);
imagesc(range_axis / 1e3, vel_axis, rd_power);
axis xy; colorbar; hold on;
plot(range_est / 1e3, vel_est, 'ro', 'MarkerSize', 6, 'LineWidth', 1.5);
title('检测结果叠加');
xlabel('距离 (km)'); ylabel('速度 (m/s)');
legend('检测点');

fprintf('检测到的目标（可能含相邻阈值点）:\n');
for idx = 1:length(range_est)
    fprintf('距离 = %.2f km, 速度 = %.2f m/s\n', range_est(idx)/1e3, vel_est(idx));
end
```

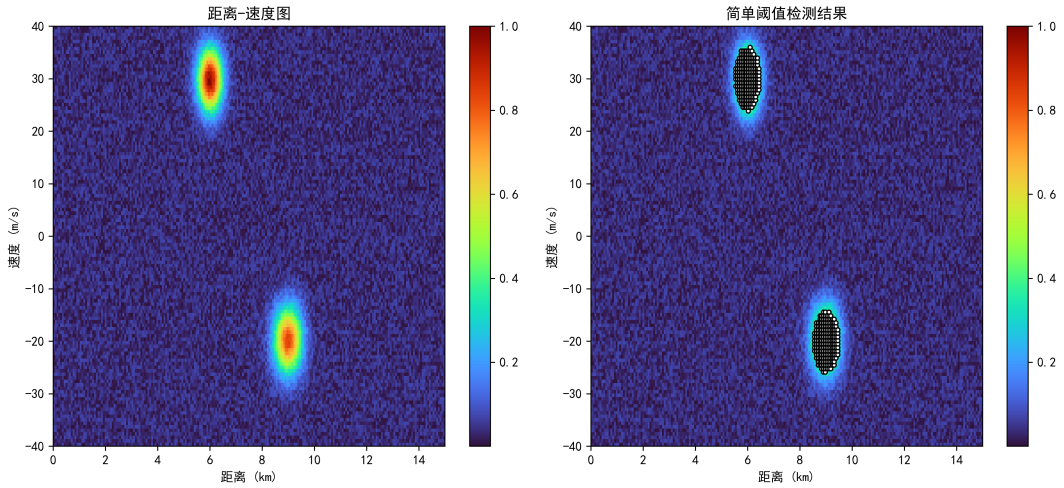

图 3: 检测前后的距离-速度结果对比

图 8-4 左图给出距离-速度幅度分布，右图给出通过阈值的检测点。若一个目标附近出现多个相邻检测点，并不意外，因为固定阈值会把主瓣附近多个高响应点一起保留下来。

8.3.2 固定阈值的三种失效模式

模式 1：强目标旁瓣触发虚警 LFM 脉冲压缩后的旁瓣约为 -13 dB。若目标 A 幅度归一化为 1.0，旁瓣幅度约为 0.22。如果 `threshold = 0.2`，旁瓣也会被检出，产生虚警。

模式 2：背景不均匀导致漏检若不同距离单元的噪声功率不同：

- 距离单元 1：噪声功率 0.1，目标信噪比 10 dB → 归一化幅度 0.32
- 距离单元 2：噪声功率 0.3，目标信噪比 10 dB → 归一化幅度 0.95

统一阈值 0.35 会漏掉单元 1 的目标。这时需要 CFAR（恒虚警率检测）。

模式 3：弱目标被噪声淹没若 `amp_targets = [1.0, 0.1]`，第二个目标很弱，归一化后可能 < 0.35。需要降低阈值或增加脉冲积累。

8.3.3 阈值选择的工程经验

归一化阈值	虚警率	漏检率	适用场景
0.2	高	低	搜索模式，宁可多报
0.35	中	中	本例默认值
0.5	低	高	跟踪模式，只要强目标

本例选择 0.35 是经验值。实际工程中，阈值应根据虚警率要求反推：

$$T = \sigma_n \sqrt{-2\ln(P_{fa})}$$

其中 σ_n 为噪声标准差，P_{fa} 为虚警概率。

8.3.4 相邻检测点的聚类方法

一个目标可能对应多个检测点（主瓣展宽）。常用聚类方法：

方法 1：局部极大值抑制

```
local_max = imregionalmax(rd_power);
det_final = det_mask & local_max;  % 只保留局部峰值
```

方法 2：形态学膨胀 + 连通域标记

```
det_mask_dilated = imdilate(det_mask, strel('disk', 2));
cc = bwconncomp(det_mask_dilated);
for k = 1:cc.NumObjects
    [~, max_idx] = max(rd_power(cc.PixelIdxList{k}));
    peak_idx = cc.PixelIdxList{k}(max_idx);
    % peak_idx 就是该簇的代表点
end
```

这些方法可以把相邻检测点合并成单个目标，但已经超出本章范围。

下一节给出完整的、带详细注释的 MATLAB 程序，并展示从原始回波到检测结果的四个处理阶段。

8.4 完整 MATLAB 示例

前面三节分别给出了距离处理、速度处理和检测的分步实现。本节把它们合成一份完整程序，并给出四个处理阶段的可视化结果。程序虽然更长，但其内部关系并不比前面复杂，只是把第 4 章到第 6 章的公式按顺序写到同一个脚本里。

8.4.1 完整代码与分段注释

完整程序可以按表 8-2 理解。

处理环节	主要公式	MATLAB 变量
发射信号生成	$k = B/T_p$, $s(t)$	`k`, `tx`
回波建模	$\tau_i = 2R_i/c$, $f_{d,i} = 2v_i/\lambda$	`tau`, `fd`, `rx_matrix`
距离压缩	$h(t) = s^*(-t)$, $y(t) = r(t) * h(t)$	`mf`, `range_data`
多普勒处理	$v = \lambda f_d/2$	`rd_map`, `rd_power`, `vel_axis`
检测判决	$\delta(x)$	`threshold`, `det_mask`
物理量输出	$\hat{R}$, $\hat{v}$	`range_est`, `vel_est`

下面给出完整 MATLAB 程序。

如果你想把本章处理链一次完整跑通，可参考 `ch08_full_processing_chain_demo.m`。若你是第一次看本章，更建议先按前面几节的顺序分别对照 `ch08_minimal_pulse_radar_demo.m`、`ch08_range_doppler_demo.m` 和 `ch08_radar_detection_demo.m`，最后再回来看这一份完整脚本。

```
% ch08_full_processing_chain_demo.m
% 第八章完整示例：从原始回波到目标检测的完整处理链
clear; close all; clc;

%% 第一部分：参数定义（对应第 4-6 章的物理参数）
c = 3e8;            % 光速 (m/s)
fc = 10e9;          % 载频 (Hz)
lambda = c / fc;    % 波长 (m)
B = 10e6;           % LFM 带宽 (Hz)
Tp = 20e-6;         % 脉冲宽度 (s)
fs = 20e6;          % 采样率 (Hz)
PRF = 2e3;          % 脉冲重复频率 (Hz)
Tr = 1 / PRF;       % 脉冲重复间隔 (s)
Npulse = 32;        % 脉冲数量
Nfast = 1024;       % 快时间采样点数

% 两个目标的距离、速度和幅度
range_targets = [6000, 9000];    % 距离 (m)
vel_targets = [30, -20];         % 速度 (m/s)
amp_targets = [1.0, 0.75];       % 幅度

%% 第二部分：发射信号生成（对应第 4 章 LFM 信号）
k = B / Tp;                      % 调频斜率 (Hz/s)
```

```
tx_t = (0:round(Tp*fs)-1) / fs;
tx = exp(1j * pi * k * tx_t.^2);              % 对应公式 s(t)
mf = conj(fliplr(tx));                        % 对应公式 h(t) = s*(-t)

%% 第三部分：回波矩阵生成（对应第 4 章距离、第 5 章速度）
rx_matrix = zeros(Npulse, Nfast);
fast_time = (0:Nfast-1) / fs;

for m = 1:Npulse
    pulse_data = zeros(1, Nfast);
    for i = 1:length(range_targets)
        tau = 2 * range_targets(i) / c;         % 对应公式  = 2R/c
        fd = 2 * vel_targets(i) / lambda;       % 对应公式 f_d = 2v/
        delayed_t = fast_time - tau;
        valid = delayed_t >= 0 & delayed_t <= Tp;
        echo = zeros(1, Nfast);
        echo(valid) = amp_targets(i) * exp(1j * pi * k * delayed_t(valid).^2) ...
                    .* exp(1j * 2*pi * fd * (m-1) * Tr);  % 对应公式
                    ↪  exp(j2 f_d(m-1)T_r)
        pulse_data = pulse_data + echo;
    end
    noise = 0.08 * (randn(1, Nfast) + 1j * randn(1, Nfast));
    rx_matrix(m, :) = pulse_data + noise;
end

%% 第四部分：距离压缩（对应第 4 章匹配滤波）
range_data = zeros(size(rx_matrix));
for m = 1:Npulse
    range_data(m, :) = conv(rx_matrix(m, :), mf, 'same');  % 对应公式 y(t) = r(t)*h(t)
end

group_delay = floor(length(tx) / 2);
range_axis = ((0:Nfast-1) - group_delay) * c / (2 * fs);

%% 第五部分：多普勒处理（对应第 5 章 FFT）
rd_map = fftshift(fft(range_data, [], 1), 1);
rd_power = abs(rd_map);
rd_power = rd_power / max(rd_power(:));

doppler_freq = (-Npulse/2:Npulse/2-1) * PRF / Npulse;
vel_axis = doppler_freq * lambda / 2;           % 对应公式 v =  f_d/2

%% 第六部分：固定阈值检测（对应第 6 章判决）
threshold = 0.35;
det_mask = rd_power > threshold;                % 对应公式 (x)
[row_idx, col_idx] = find(det_mask);
```

```
range_est = range_axis(col_idx);                % 对应公式 R
vel_est = vel_axis(row_idx);                    % 对应公式 v̂

%% 第七部分：结果可视化
figure('Position', [100, 100, 1200, 800]);

subplot(2,2,1);
imagesc(real(rx_matrix)); colorbar;
title('原始回波矩阵（实部）');
xlabel('快时间采样点'); ylabel('脉冲序号');

subplot(2,2,2);
plot(range_axis/1e3, abs(range_data(1,:)), 'LineWidth', 1.5); grid on;
title('第一个脉冲的距离像');
xlabel('距离 (km)'); ylabel('幅度');

subplot(2,2,3);
imagesc(range_axis/1e3, vel_axis, rd_power);
axis xy; colorbar;
title('距离-速度图');
xlabel('距离 (km)'); ylabel('速度 (m/s)');

subplot(2,2,4);
imagesc(range_axis/1e3, vel_axis, rd_power);
axis xy; colorbar; hold on;
plot(range_est/1e3, vel_est, 'ro', 'MarkerSize', 8, 'LineWidth', 1.5);
title('检测结果叠加');
xlabel('距离 (km)'); ylabel('速度 (m/s)');

fprintf('检测到的目标（可能含相邻阈值点）:\n');
for idx = 1:length(range_est)
    fprintf('距离 = %.2f km, 速度 = %.2f m/s\n', range_est(idx)/1e3, vel_est(idx));
end
```

8.4.2 处理结果的四阶段展示

这段程序中，tau 与 fd 分别实现第 4 章和第 5 章给出的时延关系与多普勒关系；range_data(m,:) = conv(rx_matrix(m,:), mf, 'same') 对应匹配滤波公式 $y(t) = r(t) * h(t)$；rd_map = fftshift(fft(range_ [], 1), 1) 把脉冲间相位变化换成多普勒谱；det_mask = rd_power > threshold 则对应第 6 章的阈值判决。由于距离压缩仍采用 conv(..., 'same')，程序专门通过 group_delay 修正了距离轴，因此 range_axis 上的峰值位置与目标真实距离能够对应起来。

运行程序后，图窗中的四幅图分别对应四个处理阶段：左上是原始回波矩阵，右上是单脉冲距离像，左下是距离-速度图，右下是在距离-速度图上叠加检测结果。整条雷达处理链也就在这一份 MATLAB 程序中完整呈现出来。

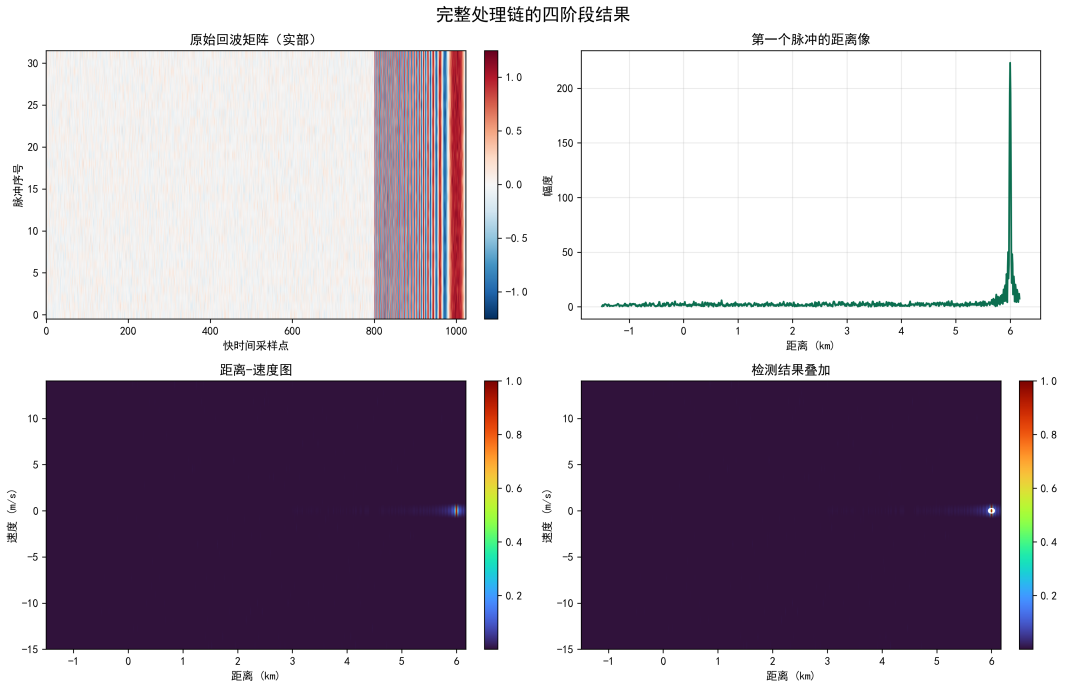

图 4: 完整处理链的四阶段结果

8.4.3 小练习

通过修改参数，可以观察不同设置对处理结果的影响。

练习 1：带宽与距离分辨率把 `B` 从 10 MHz 改成 5 MHz，观察距离像主瓣变宽。这验证了第 4 章的结论：距离分辨率由带宽决定。

练习 2：脉冲数与速度分辨率把 `Npulse` 从 32 改成 16，观察速度维模糊增加。这验证了第 5 章的结论：速度分辨率由相干积累时间决定。

练习 3：阈值与检测结果把 `threshold` 从 0.35 改成 0.2 和 0.5，对比虚警和漏检。阈值过低会增加虚警点，阈值过高可能漏掉弱目标。

练习 4：PRF 与速度模糊把 `PRF` 从 2 kHz 改成 1 kHz，观察速度轴范围变化。PRF 越低，不模糊速度范围越小，这对应第 5 章的多普勒模糊问题。

练习 5：综合题给定目标参数（距离 8 km，速度 15 m/s），预测距离-速度图上的峰值位置，再运行程序验证。

这些练习帮助读者把公式、代码和物理现象联系起来，加深对雷达信号处理完整流程的理解。