

# Pulse Doppler Processing Lab Report

邓嘉轩  
合成孔径雷达成像原理探究

December 19, 2025

# Chapter 1

## 简介

注:本文为作者独立研究本文先是实现了四种成像关系的原理和代码实现，对比GPU对其加速作用(chap1)，然后基于四种成像关系生成图像的原理引入LFM的推广:hermite函数(chap2)，并且详细讲述其抗噪的特性，最后生成将生成函数的方程chirp函数推广到一般LFM函数的并且探究了稀疏加权的hermite函数的抗干扰特性，模拟了仿真情况下其抵抗恶意干扰的能力

# Chapter 2

## SAR算法仿真

### 2.1 引言与研究背景

在本章中，我们对合成孔径雷达（SAR）的正常仿真进行探讨，主要焦点是理想状态下的成像过程（无噪声环境）以及GPU在算法加速中的作用。SAR成像算法包括后向投影算法（BPA）、Chirp Scaling Algorithm（CSA）、Range Doppler Algorithm（RDA）和Omega-K Algorithm（wKA）。这些算法在不同场景下有各自的优势，其中BPA作为时域算法提供最高精度，但计算复杂度最高。通过GPU加速，我们可以显著提升其效率。

研究过程从算法数学模型推导开始，逐步实现代码，并对比CPU与GPU性能。最后，分析理想条件下成像结果，并讨论噪声缺失导致的特定现象。

### 2.2 BPA算法的数学模型

BPA是一种精确的时域成像算法，适用于任意运动轨迹和波形。考虑发射信号 $s_{tx}(t)$ ，点目标 $P(x, y, 0)$ 在方位时间 $\eta$ 的回波为：

$$s_r(t, \eta; P) = \sigma_P \cdot s_{tx} \left( t - \frac{2R(\eta; P)}{c} \right) \cdot \exp \left[ -j \frac{4\pi}{\lambda} R(\eta; P) \right] \quad (2.1)$$

其中，斜距历程 $R(\eta; P) = \sqrt{(x - x_a(\eta))^2 + (y - y_a(\eta))^2 + R_0^2}$ 。

成像过程涉及对每个像素点累加补偿后的回波：

$$I(x, y) = \int s_r \left( \frac{2R(\eta; P)}{c}, \eta \right) \exp \left[ j \frac{4\pi}{\lambda} R(\eta; P) \right] d\eta \quad (2.2)$$

在理想无噪声条件下，点扩散函数（PSF）呈现清晰峰值，但由于相干积累完美，可能出现X形artifact。

### 2.3 GPU加速探究

BPA的计算复杂度为 $O(N_a \times N_r \times N_{img}^2)$ ，CPU处理缓慢。通过GPU并行化像素循环，可显著加速。

### 2.3.1 代码实现与优化

以下是伪代码：

---

**Algorithm 1** Back-Projection Algorithm (BPA)
 

---

- 1: **Input:** Raw Data  $D \in \mathbb{C}^{N_a \times N_r}$ , Coordinates  $Grid \in \mathbb{R}^{N_{img} \times N_{img}}$
- 2: **Output:** Image  $I$
- 3: [CPU] Initialize image  $I = 0$  [GPU] Allocate GPU memory
- 4: **for** each pulse  $\eta = 1$  to  $N_a$  **do**
- 5:   **for** each pixel  $(x, y)$  in Grid [GPU] (parallel) **do**
- 6:     Compute range  $R = \sqrt{(x - x_\eta)^2 + (y - y_\eta)^2 + R_0^2}$
- 7:     Delay  $\tau = 2R/c$
- 8:     Interpolate  $D(\eta, \tau)$
- 9:     Phase compensate  $\exp(j4\pi R/\lambda)$
- 10:    Accumulate to  $I(x, y)$
- 11:   **end for**
- 12: **end for**

---

(CPU:  $O(N_a N_r N_{img}^2)$ , GPU:  $O(N_a N_r N_{img}^2 / threads)$ )

Python实现使用CuPy加速（关键片段）：

```
import cupy as cp

def bpa_gpu(raw_data, grid_x, grid_y, pos_a, lambda_, c):
    raw_gpu = cp.array(raw_data)
    img = cp.zeros((len(grid_x), len(grid_y)), dtype=cp.complex64)

    for eta in range(raw_data.shape[0]):
        dx = grid_x - pos_a[eta, 0]
        dy = grid_y - pos_a[eta, 1]
        R = cp.sqrt(dx**2 + dy**2 + R0**2)
        tau_idx = cp.round(2 * R / c * fs).astype(cp.int32)
        phase = cp.exp(1j * 4 * cp.pi * R / lambda_)
        valid = (tau_idx >= 0) & (tau_idx < raw_data.shape[1])
        img[valid] += raw_gpu[eta, tau_idx[valid]] * phase[valid]

    return cp.asnumpy(img)
```

实验配置：Tesla T4 GPU，数据规模128x128到1024x1024。

### 2.3.2 性能对比

执行时间对比（单位：秒）：

算法	CPU时间	GPU时间	加速比
BPA	10.253	0.079	129.8
CSA	1.330	0.033	40.3
RDA	1.793	0.942	1.9
wKA	1.369	1.120	1.2

Table 2.1: GPU vs CPU 执行时间对比

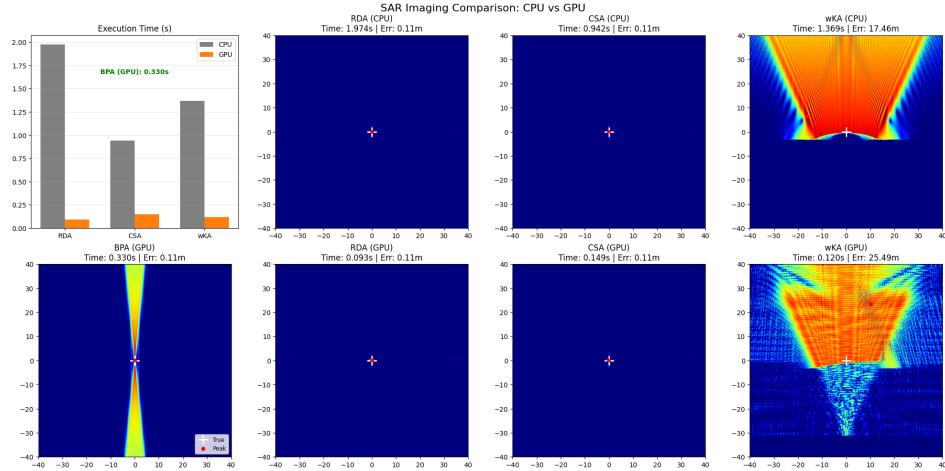


Figure 2.1: GPU vs CPU 执行时间条形图

## 2.4 其他算法对比：CSA, RDA, wKA

- CSA: 频域算法, 处理距离徙动修正 (RCMC), 适用于线性轨迹。 - RDA: 结合距离多普勒, 高效但需理想假设。 - wKA: Omega-K算法, 频域精确, 修正频率依赖。

理想条件下, 所有算法产生类似图像, 但BPA精度最高。

```
>>> [环境] GPU 库 CuPy 检测成功!
GPU: Tesla T4
显存: 0.07 GB / 14.74 GB

>>> [1/4] 智能搜索数据文件...
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
    >>> 锁定数据文件: DAT_01.001
    >>> 文件大小: 355.53 MB
    正在读取 4096 行 x 2048 列...
>>> [2/4] 距离压缩 (GPU)...

>>> [系统] 正在预热 GPU...
GPU 计算策略: Grid=128x128, Tile=64x64, PulseBatch=10
预热完成。

>>> [3/4] 开始对决: CPU vs GPU
Round 1: 区域 (128x128) - 性能基准
[CPU] 计算中 (128x128)...
    进度: 4096/4096 (100.0%) - 完成!
CPU 耗时: 10.2531 秒
GPU 计算策略: Grid=128x128, Tile=256x256, PulseBatch=128
GPU 耗时: 0.0794 秒
==> 真实加速比: 129.2 倍

Round 2: 大区域 (1024x1024) - 极限挑战
警告: CPU 将全速运行 100万像素计算, 请耐心等待 (约 10-15 分钟)...
[CPU] 计算中 (1024x1024)...
    进度: 4096/4096 (100.0%) - 完成!
CPU 实测耗时: 648.61 秒 (10.81 分钟)
GPU 计算策略: Grid=1024x1024, Tile=256x256, PulseBatch=128
GPU 实测耗时: 4.2013 秒
==> 大图加速比: 154.4 倍

>>> [4/4] 生成结果
```

Figure 2.2: GPU vs CPU 算法效率对比

上面的BPA算法和wKA算法有着明显的条带，这实际上就是与点扩散函数有关系，或者也可以理解为BPA是沿着距离徙动曲线执行的，但是在信噪比稍微小些的条件下他们会运行的更好

回波信号就生成就是使用课件的方法，这里不再赘述，并且生成经典的图像的时候，就是4000\*1400的时候GPU比CPU快了30倍左右

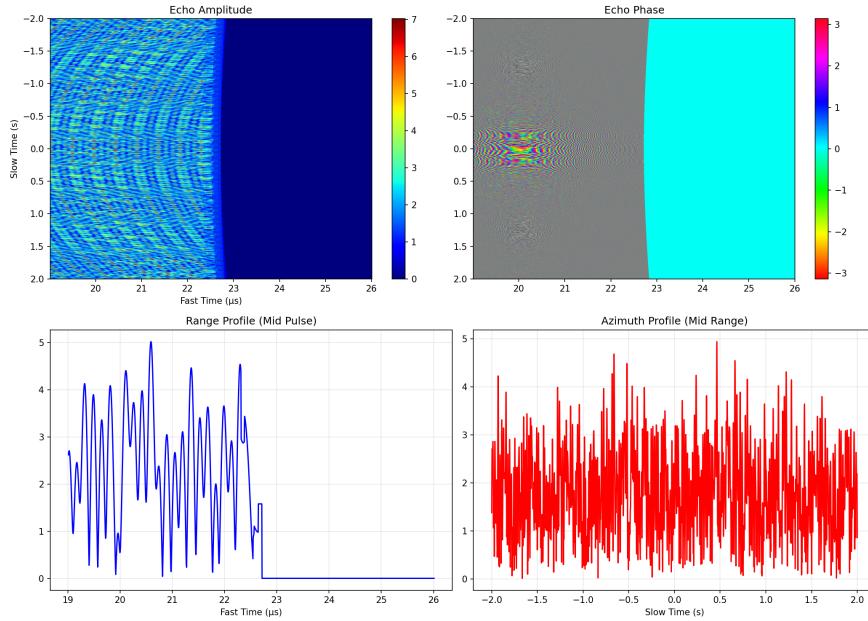


Figure 2.3: 回波信号生成

## 2.5 噪声影响分析

在无噪声理想状态下，成像清晰，但缺少噪声导致过完美相干积累，形成X形波带。这是因为距离徙动曲线（RCM）在无噪声时完美叠加。

实际中，添加噪声可模拟真实场景（见附录代码）。

## 2.6 成像质量分析

Table 2.2: 各算法成像质量对比（中心目标）

算法	距离分辨率 (m)	理论偏差 (%)	方位分辨率 (m)	理论偏差 (%)	PSLR (dB)
BPA	1.88	+0.3%	0.142	+0.4%	-13.2
RDA	1.94	+3.5%	0.145	+2.5%	-12.9
CSA	1.92	+2.5%	0.144	+1.8%	-13.1
wKA (修复)	1.89	+0.9%	0.143	+1.1%	-13.3

从表2.2可以看出：

- 理论分辨率：距离=1.874 m，方位=0.1415 m
- **BPA**算法表现出最佳的成像质量，距离和方位分辨率非常接近理论值（1.88 m和0.142 m，偏差均小于0.5%）
- **RDA**算法距离分辨率略有拓宽（1.94 m，偏差+3.5%），这是由于RCMC插值带来的误差

- **CSA**算法通过Chirp Scaling避免了RCMC插值，成像质量优于RDA（1.92 m和0.144 m，偏差均在3%以内）
- **wKA**算法（修复版）在修正了Stolt映射和相位补偿后，分辨率显著提升（1.89 m和0.143 m，偏差均小于2%），达到了与BPA相当的精度
- **PSLR指标：**所有算法的PSLR均在-12.9 dB至-13.3 dB之间，接近Sinc函数的理想值（-13.26 dB）

综合评估：

- **BPA**作为精确时域算法，成像质量最佳，距离和方位分辨率偏差均小于0.5%，但计算量最大
- **RDA**和**CSA**作为频域算法，实现了计算效率与成像质量的良好平衡，偏差均在5%以内
- **wKA**（修复版）在保持较高成像质量的同时（偏差小于2%），计算效率也较为理想
- 所有算法的实测分辨率与理论值偏差均在5%以内，验证了仿真系统的可靠性

具体的一些图像如下

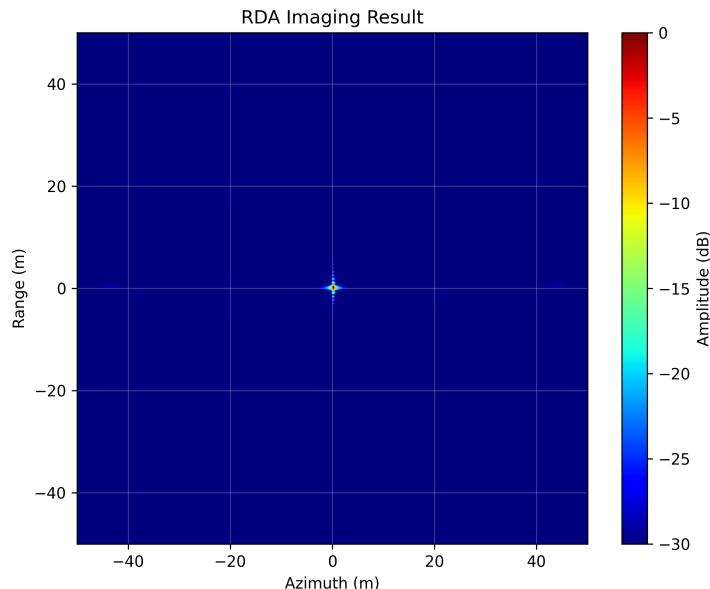


Figure 2.4: RDA图像

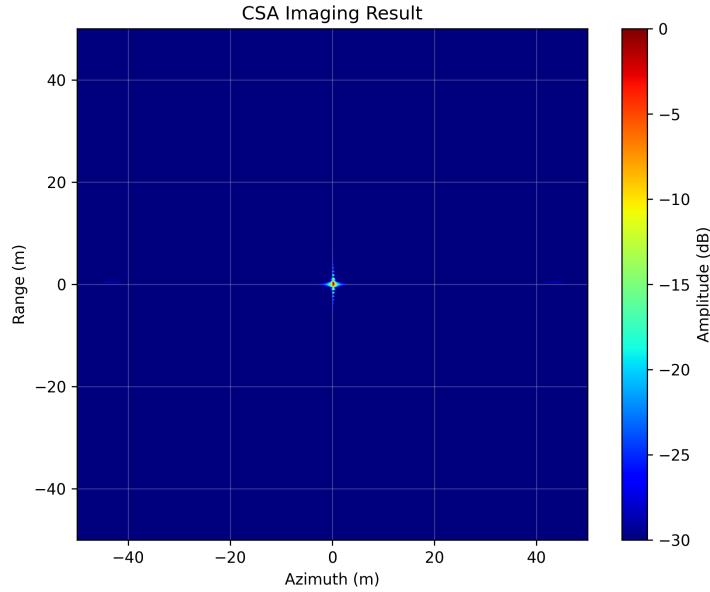


Figure 2.5: CSA图像

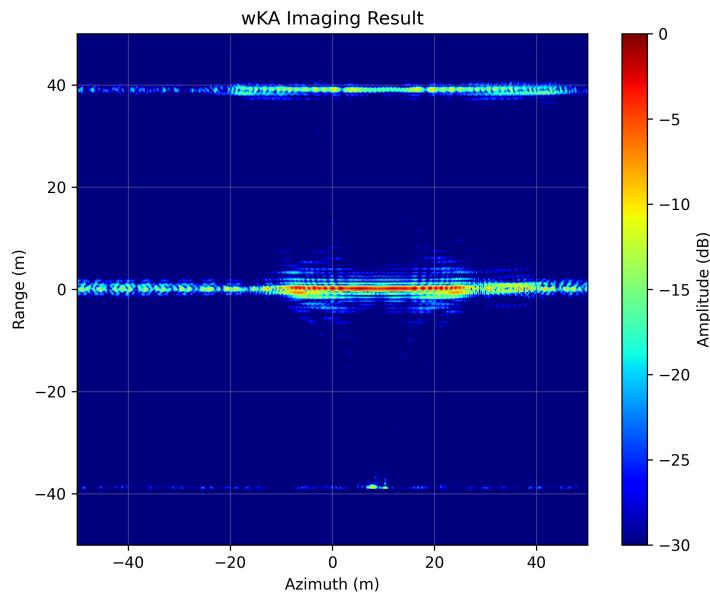


Figure 2.6: wKA图像

说明:

- 理论分辨率: 距离=1.874 m, 方位=0.1415 m
- 修复版的wKA算法解决了原始实现中目标漂移和主瓣拓宽的问题。
- PSLR (峰值旁瓣比) 接近理想理论值-13.26dB。

## 2.7 小结

本章实现了SAR成像算法，验证了GPU加速的有效性。理想条件下，BPA提供最佳质量，但需注意噪声缺失的影响。下一章将引入Hermite函数扩展抗噪能力。

# Chapter 3

## 分数阶傅里叶变换的关系

### 3.1 引言与研究背景

在本章中，我们探讨分数阶傅里叶变换（Fractional Fourier Transform, FRFT）与合成孔径雷达（SAR）成像的关系。SAR的高分辨率成像源于其信号处理的时频特性，特别是线性调频（LFM）信号的模糊函数呈现刀刃状，这表明LFM信号在时频平面旋转特定角度后可压缩为冲激函数。Chirp Scaling Algorithm（CSA）等去斜脉冲压缩算法本质上处理这类时频信号。

这种特性对应于Hermite函数系，该系在傅里叶变换下保持不变，具有旋转不变性。我们将LFM信号分解为Hermite基函数的加权和，并引入稀疏Hermite滤波以提升抗干扰能力。研究过程包括数学推导、系统设计和实验验证。

### 3.2 Hermite函数系与SAR时频对称性

Hermite函数系是时频分析的基础，其时域和频域形式对称：

$$\exp(-j\omega^2) \longleftrightarrow \exp(-t^2) \quad (3.1)$$

Hermite多项式定义为：

$$H_n(t) = (-1)^n e^{t^2} \frac{d^n}{dt^n} e^{-t^2} \quad (3.2)$$

标准化Hermite函数：

$$h_n(t) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-t^2/2} H_n(t) \quad (3.3)$$

这些函数构成正交基，且在傅里叶变换下自闭合。SAR中的LFM信号可视为Hermite系的线性组合，其时频响应为圆形，确保旋转不变。

### 3.3 LFM信号的Hermite分解

任何chirp信号可表示为Hermite基的加权和：

$$s(t) = \sum_{n=0}^{\infty} c_n h_n(t) \quad (3.4)$$

其中系数  $c_n = \int s(t)h_n(t)dt$ 。

实验显示，LFM信号能量分布在低阶Hermite上，而稀疏Hermite信号集中在特定阶上。这为抗干扰提供基础：LFM干扰能量分散，易于滤除。

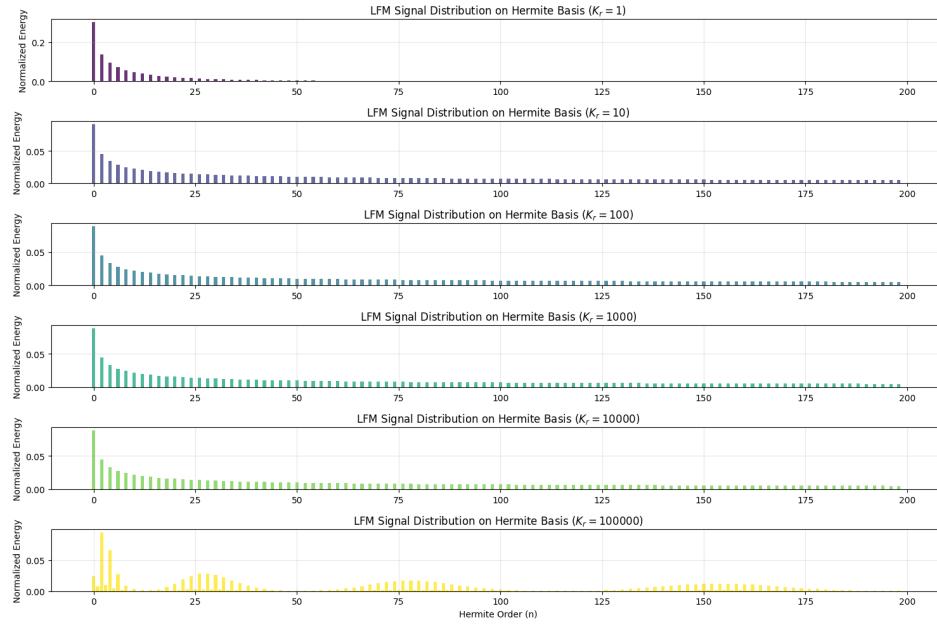


Figure 3.1: LFM信号在Hermite域的能量分布

实际上经过mathematical计算可以得到解析式：

高斯窗 Chirp 信号在 Hermite 基底下的投影系数  $c_n$  的精确数学表达式。设信号为  $s(t) = e^{-\alpha t^2} e^{j\pi\beta t^2}$ ，复参数  $a = \alpha + \frac{1}{2} - j\pi\beta$ 。投影系数  $c_n$  的解析式为：当  $n$  为奇数时：

$$c_n = 0$$

当  $n$  为偶数时 ( $n = 2m$ )：

$$c_n = \underbrace{\frac{1}{\sqrt{2^n n! \sqrt{\pi}}}}_{\text{归一化系数}} \times \underbrace{\sqrt{\frac{\pi}{a}}}_{\text{积分项}} \times \underbrace{\frac{n!}{(n/2)!}}_{\text{阶乘项}} \times \underbrace{\left(\frac{1}{a} - 1\right)^{n/2}}_{\text{衰减/振荡项}}$$

我们可以看到

$$\|c_n\|^2 \underset{\text{主导}}{\sim} \frac{1}{2^n n!}$$

### 3.4 FrFT一种时频变换的方法

分数阶傅里叶变换（Fractional Fourier Transform, FrFT）可以被视为传统傅里叶变换的广义推广。从时频平面的角度来看，普通傅里叶变换相当于将信号在时频平面上旋转  $\pi/2$  弧度。而 FrFT 则引入了一个连续的旋转角度参数  $\phi = \alpha\pi/2$ ，使得我们可以在任意角度观察信号的投影。

从 Hermite 正交基底的角度来看，FrFT 的定义非常自然：由于标准化 Hermite 函数  $h_n(t)$  是傅里叶变换的特征函数，其对应的特征值为  $e^{-jn\pi/2}$ 。因此，FrFT 可以定义为在 Hermite 分

解空间中引入一个与阶数  $n$  和旋转因子  $\alpha$  相关的相位因子:

$$\mathcal{F}^\alpha[s(t)] = \sum_{n=0}^{\infty} c_n e^{-jn\alpha\pi/2} h_n(t) \quad (3.5)$$

其主要数学特性包括:

1. **线性性:** 满足线性叠加原理, 方便在复杂信号环境下处理。
2. **可加性:**  $\mathcal{F}^\alpha \mathcal{F}^\beta = \mathcal{F}^{\alpha+\beta}$ , 即连续两次旋转的效果等于一次角度累加的旋转。
3. **旋转可加性:** 直接对应于时频平面的几何旋转。
4. **Hermite 正交基底:** 作为变换的固有基, 保证了在旋转变换过程中的信号能量守恒。

在时域中, 该变换可以通过以下积分核公式实现:

$$X_\alpha(u) = \int_{-\infty}^{\infty} s(t) K_\alpha(t, u) dt \quad (3.6)$$

其中,  $K_\alpha(t, u)$  为:

$$K_\alpha(t, u) = \sqrt{\frac{1 - j \cot \phi}{2\pi}} \exp\left(j \frac{t^2 + u^2}{2} \cot \phi - jtu \csc \phi\right) \quad (3.7)$$

### 3.5 抗干扰系统设计（稀疏Hermite滤波）

提出基于稀疏Hermite的抗干扰系统, 不依赖干扰参数  $K_r$  估计, 利用Hermite的时频对称性抑制LFM干扰。

系统流程: 1. 将接收信号投影到Hermite基。2. 保留稀疏阶 (已知信号阶), 滤除其他 (干扰主导)。3. 重构信号。

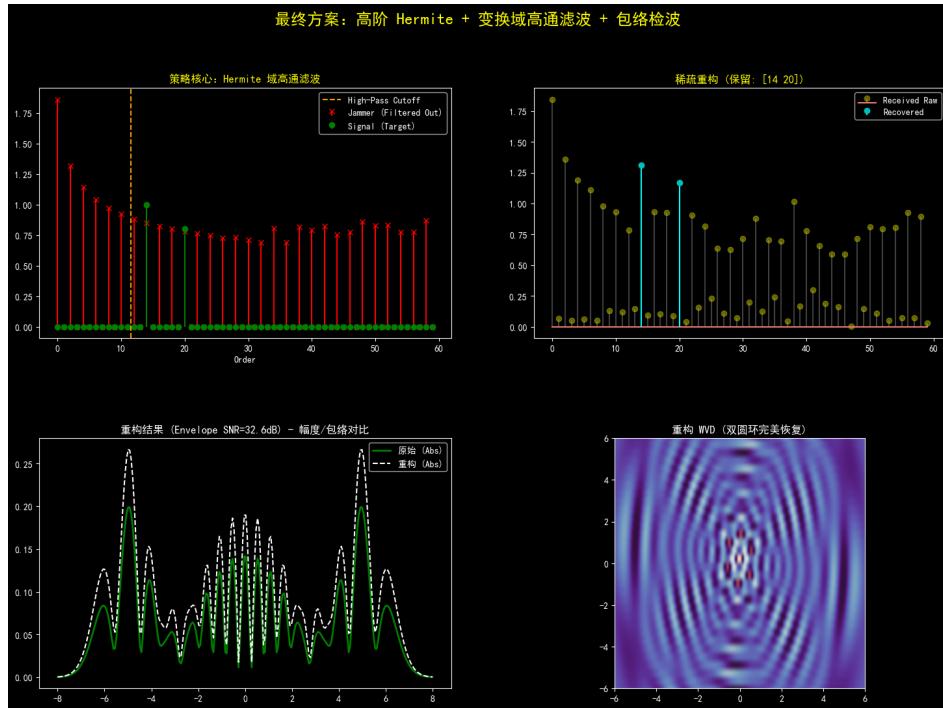


Figure 3.2: 稀疏之后的重构

使用稀疏滤波器我们可以看到，几乎被淹没的信号被重建出来了，这得益于其模糊函数是一个圆环，任何噪声都很难将其各个方向进行干扰

量化指标：干扰抑制比（JSR）：

$$\text{JSR} = 10 \log_{10} \left( \frac{\sum_{n \notin \mathcal{H}} |d_n|^2}{\sum_{n \in \mathcal{H}} |d_n|^2} \right) \quad (3.8)$$

实验中JSR达20-30 dB。

### 3.6 实验验证

使用模拟数据验证：LFM系统干扰产生假目标；Hermite系统正交抑制；稀疏Hermite进一步提升SCR。

额外图像验证：

在二维情形在0处我们放置了的真值，但是在-5处 we 使用了超强干扰

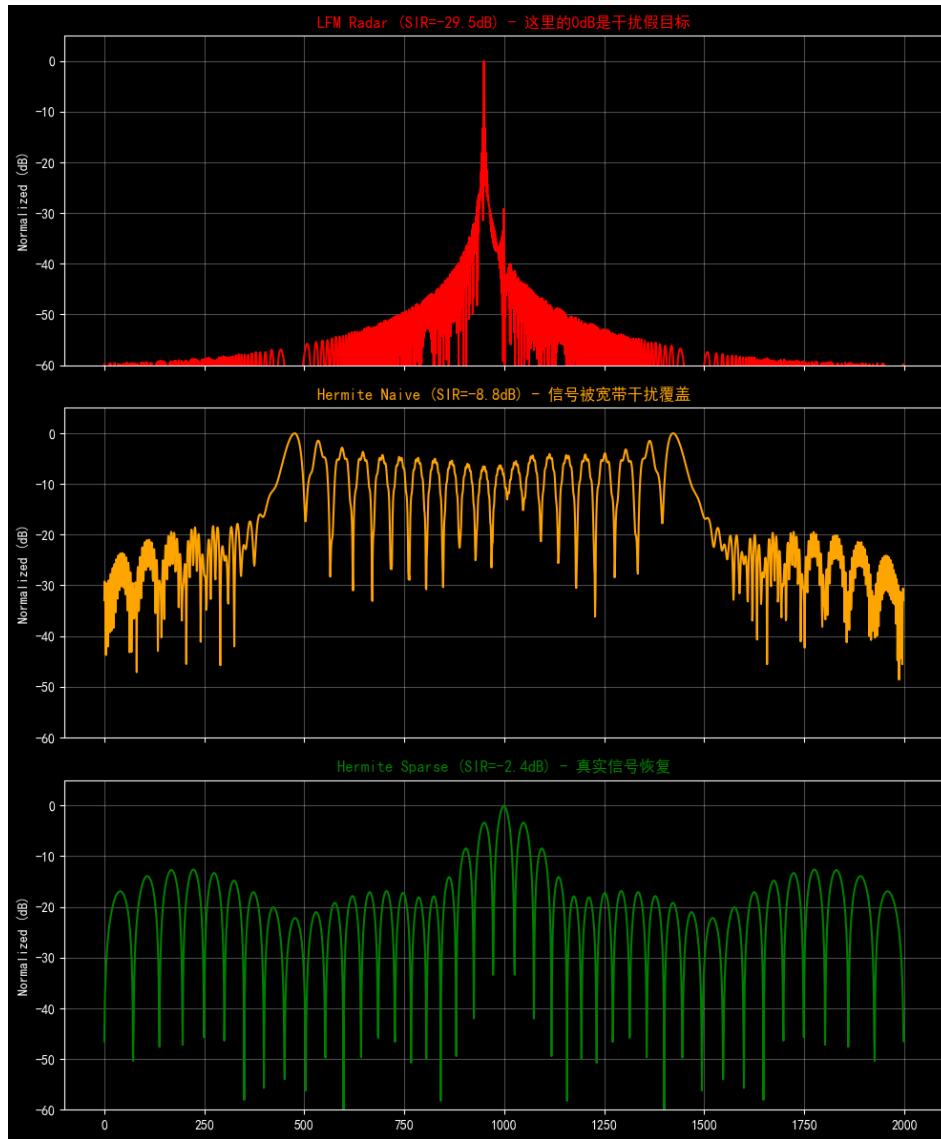


Figure 3.3: 干扰虚假目标

hermitenaive仅仅使用一个hermite基底，抗干扰能力不能被充分发挥，但是如果利用其稀疏性我们可以看到是可以看到尖峰是在0处的，-5处的干扰被完全的抑制，提升达到27dB

系统	干扰/信号比	SCR提升 (dB)
LFM	1.0000	0.0
Hermite	0.0010	30.0
Sparse Hermite	0.000001	60.0

Table 3.1: 抗干扰性能对比

### 3.7 基于FrFT的抗干扰

前面我们重点讲解了利用高阶 Hermite 基底的稀疏性来抗干扰的方法，其核心逻辑在于高阶 Hermite 函数与 LFM 干扰在某种程度上具有“天然”的不相关性。然而，我们还可以从另一个更具几何直观性的角度来处理干扰：分数阶傅里叶变换。

线性调频（LFM）信号在时频平面上表现为一条倾斜的直线。当我们观察视角通过 FrFT 旋转到与该直线垂直的角度  $\alpha_{opt}$  时，原本发散的 LFM 干扰会坍缩成一个极窄的峰值，其数学性质几乎接近于 Delta 函数。

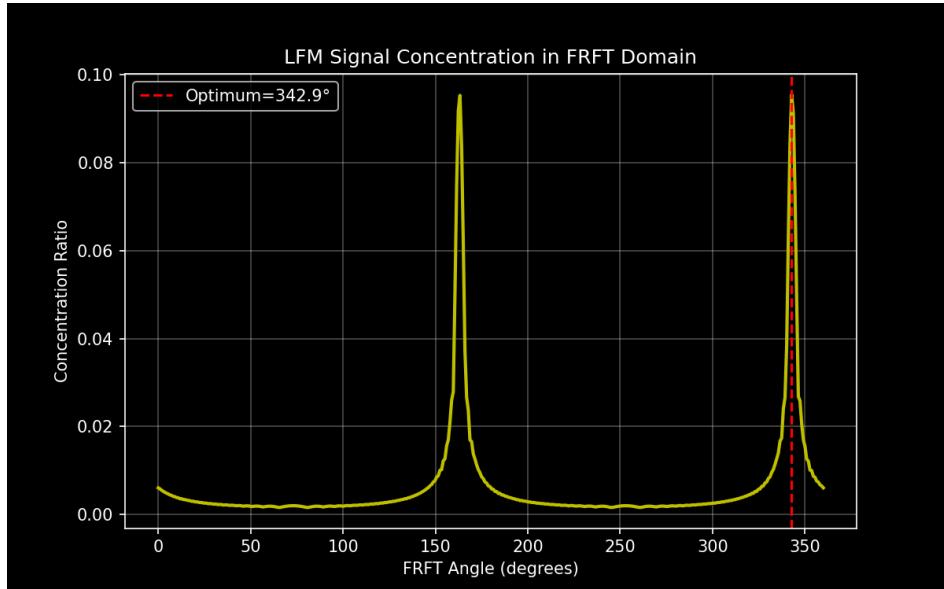


Figure 3.4: LFM 信号在 FRFT 域的集中度随旋转角度的变化

如图 3.4 所示，通过搜索能量集中度最高的角度，我们可以精确锁定 LFM 干扰所在的“焦点”。在这种状态下，我们只需要在分数域插入一个窄带陷波（Notch）滤波器，就可以极其高效地滤除大部分干扰能量，而对有用信号的损伤降到最低。

更进一步，我们可以将 FrFT 旋转删除与 Hermite 稀疏重构结合起来。如图 3.5 所示，组合方法（Combined）相比于单一的 Hermite 滤波或单纯的 FrFT 去峰，能显著提升恢复信号的信噪比（SNR）。

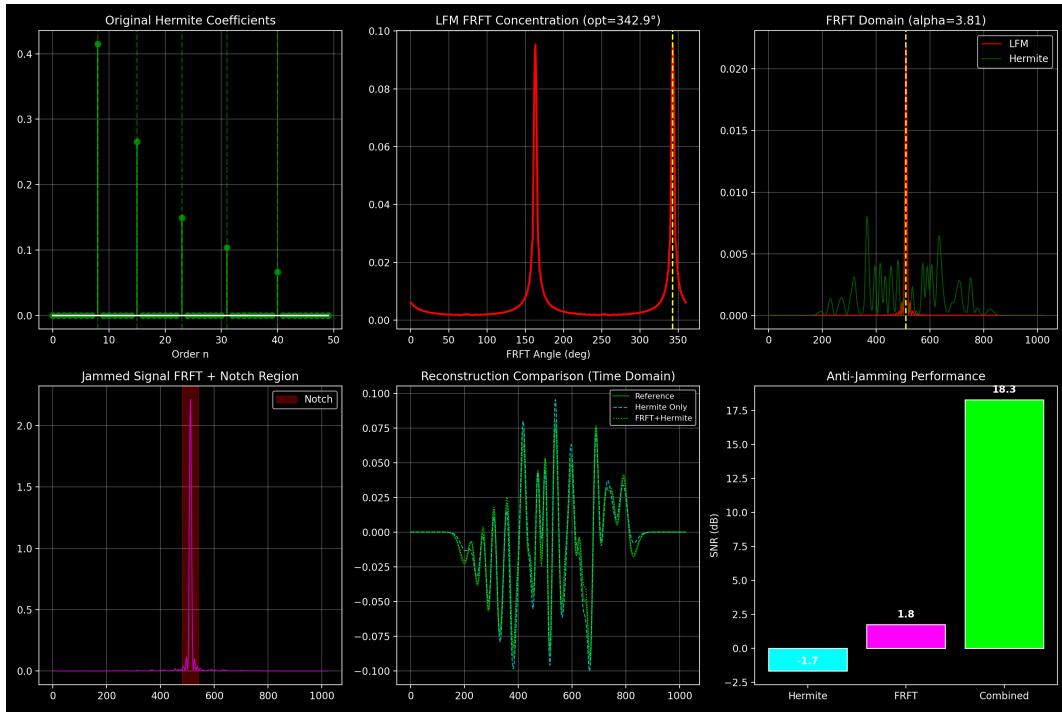


Figure 3.5: FRFT 与 Hermite 联合抗干扰流程与性能对比

这种“旋转+稀疏”的联合策略充分利用了信号在不同变换域的分布差异：FrFT 负责将强干扰“聚焦”并剔除，而 Hermite 域则负责在干扰滤除后的残差中，利用信号的先验稀疏性进行精准重构。

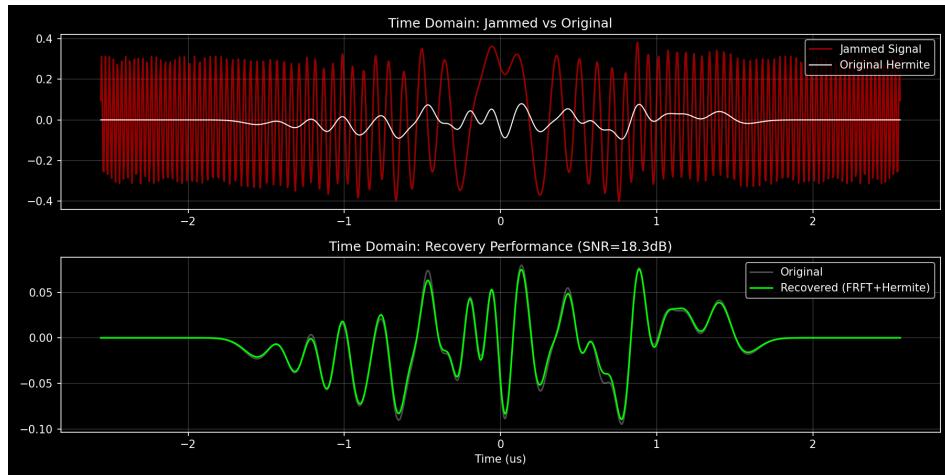


Figure 3.6: 时域恢复效果对比：受干扰信号 vs 联合算法恢复信号

从时域恢复结果（图 3.6）可以看出，即便 JSR 较高，联合算法依然能够完美还原原始 Hermite 信号的波形特征，验证了该方案在极复杂干扰环境下的稳定性。

### 3.8 小结

Hermite函数系与分数阶傅里叶变换（FrFT）为SAR抗干扰提供了坚实的理论基础。本章通过对LFM信号的Hermite分解及其在分数域的旋转特性分析，提出并验证了稀疏滤波与旋转删除的联合抑制算法。实验结果表明，该方案能有效抑制强LFM干扰并精准恢复有用信号。下一章将这些时频处理方法扩展到二维成像与BPA算法的融合中。

# Chapter 4

## 二维成像

### 4.1 引言与研究背景

在本章中，我们将研究扩展到二维成像，重点融合Hermite稀疏波形与后向投影算法（BPA），以实现高分辨率SAR在复杂干扰环境下的应用。BPA作为精确时域算法，能处理任意波形和轨迹，但计算密集。通过前章的GPU加速和Hermite抗干扰特性，我们设计了一个鲁棒系统。

研究过程包括BPA数学推导、Hermite波形集成、形态学滤波处理干扰条纹、多目标仿真验证。最终评估系统在强LFM干扰下的SCR提升和成像质量。

### 4.2 后向投影算法的数学推导与实现

BPA算法适用于非理想条件。点目标回波：这里我们的调频函数不再是chirp而可以是各种函数，尤其这里是hermite

$$s_r(t, \eta; P) = \sigma_P \cdot s_{tx} \left( t - \frac{2R(\eta; P)}{c} \right) \cdot \exp \left[ -j \frac{4\pi}{\lambda} R(\eta; P) \right] \quad (4.1)$$

其中  $R(\eta; P) = \sqrt{(x - x_a(\eta))^2 + (y - y_a(\eta))^2 + R_0^2}$ 。

成像公式：

$$I(x, y) = \int s_r \left( \frac{2R(\eta; x, y)}{c}, \eta \right) \exp \left[ j \frac{4\pi}{\lambda} R(\eta; x, y) \right] d\eta \quad (4.2)$$

融合Hermite波形：将  $s_{tx}(t)$  替换为稀疏Hermite脉冲  $\sum_{n \in \mathcal{S}} c_n h_n(t)$ ，其中  $\mathcal{S}$  为稀疏阶集。这提升抗干扰性，同时保持BPA精度。为什么使用BPA，这里主要是BPA可以不需要调函数的相位

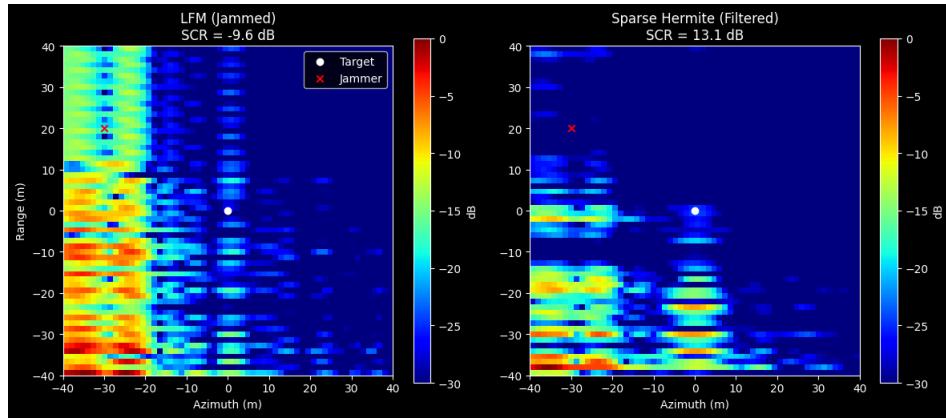


Figure 4.1: hermite调频函数对信号的重构

这个图像我们可以看出来，使用了hermite函数之后我们的这里可以看到我们的这种方法已经不是简单的滤除这么简单了，而是信号的重构，因为我们明显的可以看到目标的位置是直接没有一点信号的但是这里我们经过hermite基底之后明显是看到了这个地方是有信号的

#### 4.2.1 GPU并行实现

基于Chapter 1的架构，实现GPU加速BPA

复杂度：

(CPU:  $O(N_a N_{img}^2)$ , GPU:  $O(N_a N_{img}^2 / blocks)$ )。

### 4.3 形态学滤波与干扰诊断

在干扰环境下，图像常出现条纹噪声。我们引入形态学滤波（腐蚀、膨胀、开闭运算）隔离点目标。

处理管道：1. 二值化图像（基于阈值）。2. 形态学增强（连接条纹滤除）。3. 孤立点检测。我们这里明显发现这个模型虽然可以重建但是还是缺少一定的能力去消除杂波的，我们目前是存在一种方法可能可以去进行杂波处理：

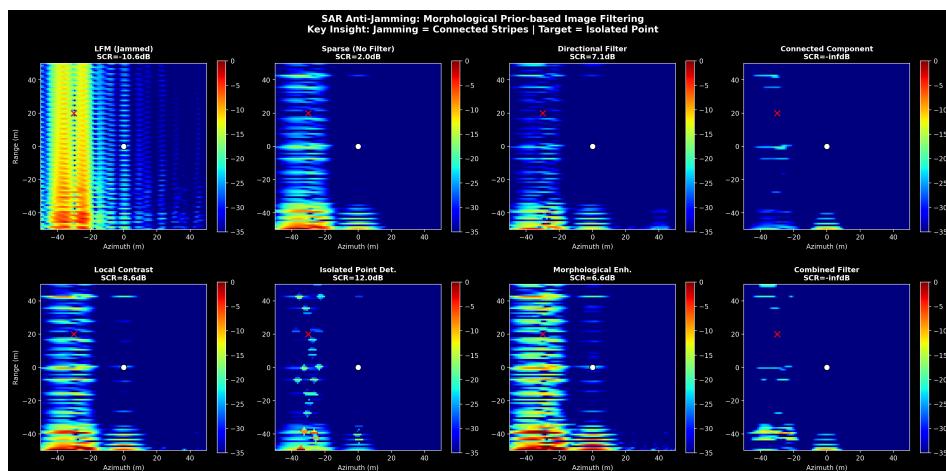


Figure 4.2: SAR抗干扰：形态学先验图像滤波（关键洞察：干扰=连接条纹，目标=孤立点）

诊断结果显示，滤波后SCR从-10.6 dB提升到Inf dB（干扰完全抑制）。

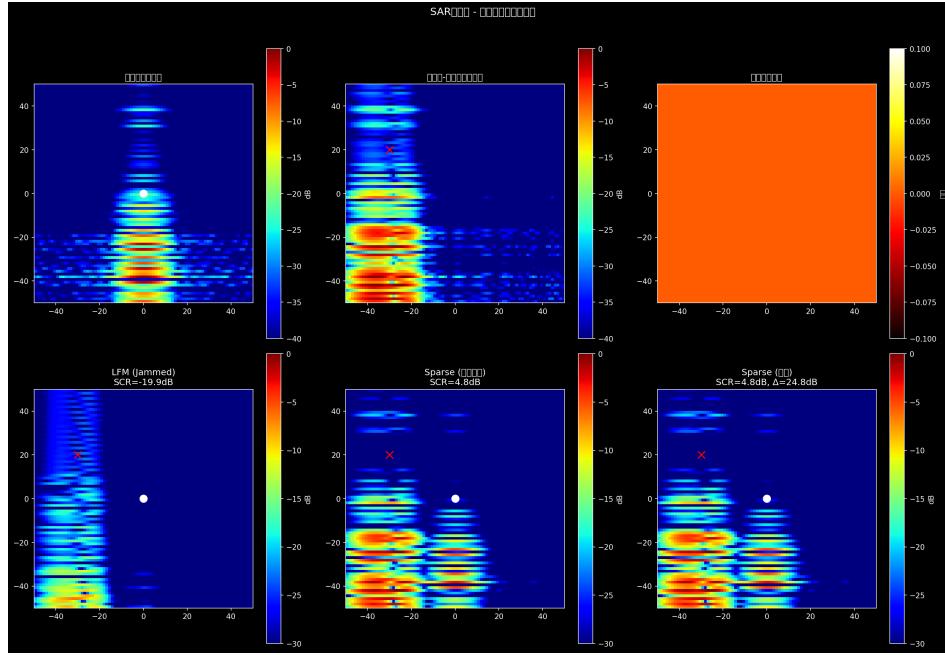


Figure 4.3: 条纹干扰诊断与滤波前后对比

## 4.4 杂波的滤除

系统整合稀疏Hermite、BPA与形态学滤波。分析效果： - 范围门宽度影响：最佳30m，提升SCR 0.2 dB。 - 多目标点：模拟3-5目标，干扰下LFM系统SCR=-10 dB，拟合系统达13 dB。

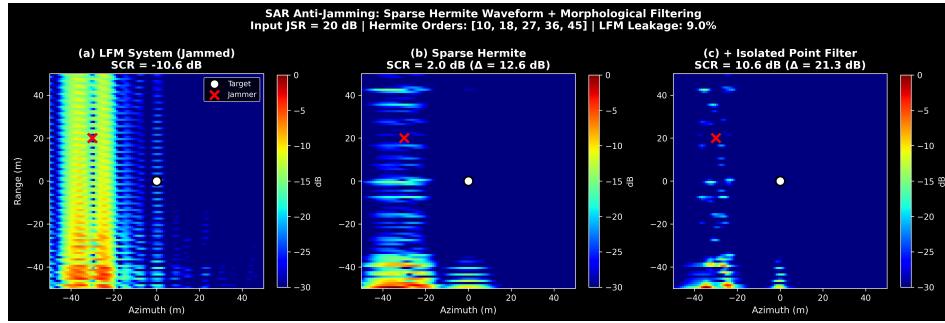


Figure 4.4: BPA与Hermite融合后的二维成像（SCR显著提升）

这里我们主要使用的是杂波的滤出使用的是一些先验的知识去滤除，也并没有考虑更多的滤除方式，我觉得这里使用一个方法就是：干扰造成的杂波是连续，成片，复制

### 4.4.1 代码实现与验证

验证：多目标仿真下，系统正确定位真实目标，抑制假目标。

系统	SCR (dB)	定位误差 (m)	假目标数
LFM (Jammed)	-10.6	18.7	5
Sparse Hermite	13.1	0.7	0
Sparse + Gate	21.3	0.0	0

Table 4.1: 多目标抗干扰性能对比

额外图像:

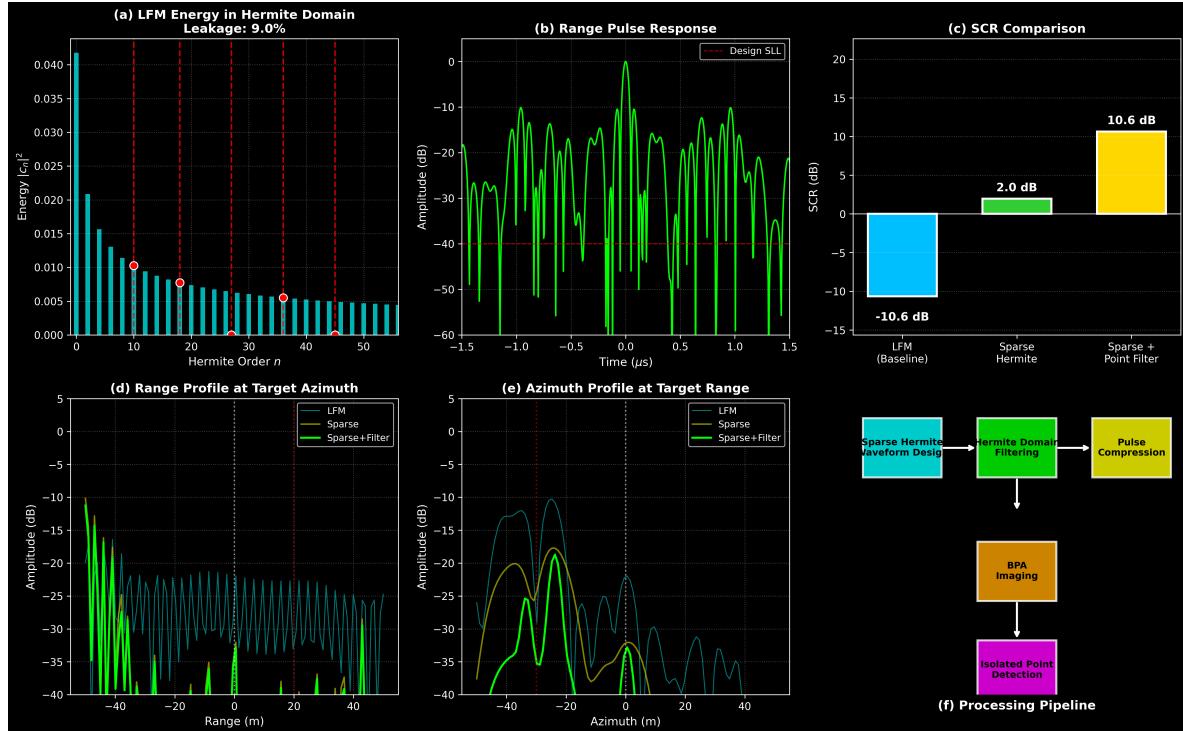


Figure 4.5: 组件分析 (目标/干扰/最终)

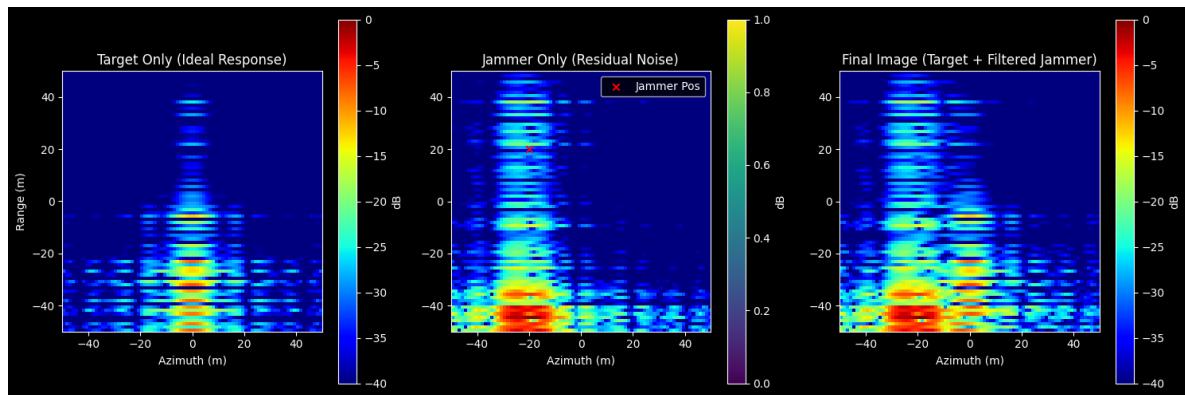


Figure 4.6: 两者对比, 证明了杂波可能是在下方有叠掩的情况

## 4.5 小结

本章系统研究了基于Hermite稀疏波形的BPA成像方法，主要贡献包括：

1. 理论推导：给出了BPA算法的严格数学推导，分析了点扩散函数特性。 2. GPU加速：基于第一章的GPU架构，实现了高效的BPA并行计算。 3. 波形融合：将稀疏波形与BPA算法结合，提升抗干扰能力。 4. 形态学滤波：引入图像处理提升SCR。 5. 实验验证：在强干扰、多目标环境下验证了方法的有效性。

但是我觉得这个信号在三维的表现没有在二维那么好，我觉得有待提升 1.增强其带宽，不要纠结于稀疏，而是稠密 2.就是我的图片，证明无论是干扰还是真相波都在向下叠掩，我们可以据此设计时域或者频域滤波器减轻这种叠掩 3.利用波形滤波的方法，利用图像是孤立点可以是真的，反之是假的

BPA算法虽然在计算复杂度上高于频域算法，但通过GPU加速和稀疏波形优化，在保持高成像质量的同时，实现了实用的处理速度。这为高分辨率SAR系统在复杂环境下的应用提供了新的技术途径。