



中国研究生创新实践系列大赛
中国光谷·“华为杯”第十九届中国研究生
数学建模竞赛

学 校

武汉大学

参赛队号

22104860055

1.

易旭憧

队员姓名

2.

周宇轩

3.

潘娟霞

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生

数学建模竞赛

题 目 **移动场景超分辨定位问题**

摘 要：

移动场景的超分辨率定位具有极大的应用价值，然而，现有的面向频连续波雷达 FMCW 的算法，例如快速傅里叶变换测距算法、MUSIC 算法、压缩感知算法等，皆不同时具备低复杂度以及高分辨率这两个特征。因此，本文就如何实现移动场景低复杂度的超分辨率定位进行了数学建模，模型有效地解决了移动场景下低复杂度地实现精确、高效定位的问题。

针对问题一，根据提供的无噪声仿真数据，考虑到天线阵列尺寸较小而多个目标物体之间的距离很近，无法使用直接测距交会的算法，因此在 2D-MUSIC 算法的基础上，编写了专门针对 FMCW 阵列天线雷达信号的 FMCW-MUSIC 算法，建立了超分辨定位模型，确定了目标物体数 $K=2$ ，在测距和方位角搜索步长分别为 0.01m ， 0.05° 的参数设置下，求解出两个目标物体的极坐标分别为 $(-0.25^\circ, 7.00\text{ m})$ 和 $(0.5^\circ, 7.00\text{ m})$ ，定位精度约为 1cm （径向与切向）。此外，为了提升算法效率，防止因提高分辨率带来的复杂度增高，我们还对 FMCW-MUSIC 算法进行了效率优化：主要通过频谱峰值测距信息辅助以及多层级分辨率搜索。最终求解问题一中数据只需 0.22 秒，比优化前的算法提升了 200 多倍。

针对问题二，根据提供的高斯噪声仿真数据，在问题一中 FMCW-MUSIC 算法的基础上，针对环境噪声场景对空间-时间域扫描窗口大小尺寸 l_1, l_2 进行了参数调优和改进；最终选取窗口尺寸为 $(20, 10)$ 时，可清晰地定位到两个目标物体，其极坐标分别为 $(-0.3^\circ, 8.20\text{ m})$ 以及 $(0.45^\circ, 8.20\text{ m})$ ，并且保持了较高的运算效率。

针对问题三，考虑到所需定位的物体为动态，我们充分利用物体运动的连续性对搜索范围进行预测，辅以梯度下降法，在问题二中算法基础上加以改进，实现了在线低复杂度的连续跟踪 FMCW-MUSIC 算法，求解出了两个物体的运动轨迹。此外，我们评估了在使用/未使用运动连续性条件下的四种跟踪定位方案性能，最终运动连续性约束+梯度下降法的方案求解时间最短，为 0.027 秒；样点计算数从不考虑连续性情况下的 20090 次下降至 39 次，复杂度明显降低。

针对问题四，天线阵列由于老化、压力等原因，其自身的定位会引入系统性误差，

但是考虑到单天线 FMCW 的测距性能不会受到其影响，利用这一点可以对算法进行约束，尽可能减小结构性噪声带来的影响。基于此，可以求得两个目标物体的极坐标分别为 $(0.15^\circ, 6.01\text{m})$ 以及 $(-0.15^\circ, 6.09\text{m})$ ，精度约为 3 厘米。

关键词：超分辨定位，2D-MUSIC 算法，FMCW-MUSIC 算法，移动场景

目录

1. 问题重述与分析	4
1.1 问题背景	4
1.2 问题分析	4
2. 模型假设与符号说明	6
2.1 模型假设	6
2.2 符号说明	6
3. 问题一模型建立与求解	7
3.1 模型建立	7
3.11 FMCW 雷达信号基本形式	7
3.12 FMCW 快速傅里叶变换测距的基本原理	7
3.13 FMCW-MUSIC 算法基本原理	8
3.2 问题求解	12
3.21 数据预处理	12
3.22 使用 FMCW-MUSIC 算法进行目标定位	13
3.23 目标物体数 K 值的确定	13
3.24 搜索步长的选取	15
3.25 目标物体定位结果	17
3.26 FMCW-MUSIC 算法搜索效率的优化	18
4. 问题二模型建立与求解	20
4.1 模型建立	20
4.2 模型求解	20
4.21 数据预处理	20
4.22 噪声环境下 FMCW-MUSIC 窗口参数 l_1, l_2 的调优	21
4.23 噪声环境下的目标物体定位	24
5. 问题三模型建立与求解	26
5.1 模型建立	26
5.2 模型求解	27
5.21 数据基本情况分析	27
5.22 不考虑物体运动连续性情况下的 FMCW-MUSIC 定位	28
5.23 考虑物体运动连续性情况下的 FMCW-MUSIC 定位	29
5.24 目标物体连续跟踪定位结果	30
5.25 性能分析评估	31
6. 问题四模型建立与求解	32
6.1 模型建立	32
6.2 模型求解	32
6.21 数据预处理	32
6.22 在结构误差下进行 FMCW-MUSIC 目标定位	33
6.23 单天线测距信息约束的 FMCW-MUSIC 目标定位	34
6.24 结构误差下目标物体定位结果	34
参考文献	36
附录	37

1. 问题重述与分析

1.1 问题背景

所谓移动场景超分辨定位，即在探测范围内存在多个物体信号叠加的情况且被探测的物体处于移动的情况下，仍能利用鲁棒的低复杂度在线算法将其精准定位到。移动场景的超分辨定位在日常生活中有极大的应用价值。例如，自动驾驶场景中精准定位周围行人车辆的运动速度和位置；家庭生活中利用扫地机器人等工具在房间内快速找到丢失的小物品；无人机飞行过程中准确探测到周围可能出现的障碍物等等。

现有移动场景定位产品通常采用调频连续波雷达 FMCW^[1]发射接收雷达信号波来定位。其信号处理的算法包括基于傅里叶变换的现有产品基线算法、MUSIC 算法以及压缩感知算法等。现有产品基线算法是通过加 Hamming 窗，然后做快速傅里叶变换来测距、测角。该算法复杂度虽低，但分辨率也很低，无法满足移动场景超分辨的要求。MUSIC（多重信号分类）算法^[2]是 1979 年由 Schimdt 提出的，它利用数据的相关矩阵具有空间结构特性这一特点，将数据空间分解为信号和噪声子空间，并利用子空间的正交性质，对谱峰进行搜索。它是真正意义上的超分辨算法，只是它需要较多的采样值来达到超分辨，且容易受到噪声的干扰而导致分辨率下降。压缩感知算法首先是由 Claerbout 和 Muir 应用在地质信号处理上^[3]，后来由 Donoho、Candes 及华裔科学家 Tao 发展起来。它是一种利用信号的稀疏性，对信号进行远小于奈奎斯特采样率的采样，并精确重构信号的算法^[4]，极大的减小了采样成本，不过该算法的缺点在于高精度与低复杂度不可兼得。

总之，以上三种算法都无法同时保证超分辨率和低复杂度，因此，亟需对当前算法进行改进来实现移动场景超分辨定位。在这个背景下，本题针对特定设备以及应用场景进行了信号数据的仿真，给出了以下四个待解决的定位问题。

1.2 问题分析

1) 问题一：针对无噪声仿真数据建立定位模型。

问题一中提供的是理想情况下的无噪声仿真数据，因此数据噪声的问题可以暂时忽略，需要关注的主要是实现更高的测距、测角分辨率，从而实现（多个）物体的定位。针对于本套系统，可以了解到，阵列天线的孔径为 $L = 0.0815$ 米，86 个等效天线之间的间距仅为 $d = 0.085 / (86 - 1) = 0.00096$ 米，。因此，对某一目标物体进行测距时，不同的等效天线所测得的距离值会非常相近（考虑目标物体位于正前方 5 米处，则不同天线的测距值真值之差不超过 0.001 米）。所以，单纯使用测距值进行交会测量来测得目标物体的位置是不现实的。此外，本系统面向超分辨率定位，需要考虑多个目标物体非常邻近的情况。因此，我们将编写超分辨率算法——改进后的专门针对 FMCW 的 MUSIC 算法（FMCW-MUSIC 算法）来解决这个问题。

2) 问题二：针对高斯噪声仿真数据建立超分辨定位模型。

与问题一相比，问题二同样是对一个 chirp 的数据进行处理，不过，问题二的数据增加了环境噪声。信号信噪比的下降将会直接对测距、测角的精度以及分辨率带来影响。所以在使用相应的算法进行求解时，需要考虑到噪声的影响，防止极端的参数设置引起错误的求解。我们考虑对问题一中所编写的 FMCW-MUSIC 算法进行特定参数的优化和改进，以实现噪声条件下物体的探测和超分辨率定位。

3) 问题三：针对运动物体数据建立在线超分辨低复杂度定位模型，并验证算法性能。

相较于问题一和二静态物体，问题三针对的是动态物体。即需要考虑连续的多个 chirps 的数据，并且在准确定位动态目标的基础上，实现更高的算法效率，以满足低复杂

度的要求。此外，对目标物体的连续跟踪还需要解决前后时刻同一目标物体的关联问题。考虑到物体在空间中运动的连续性，可以充分利用这一条件，实现目标连续跟踪算法并对算法进行效率上的提升。

4) 问题四：设计提升定位算法鲁棒性的改进算法。

使用 MUSIC 等天线阵列算法进行测距、定向时，通常需要已知天线阵列的几何情况。但是在实际使用过程中，由于老化、压力等原因，天线阵列的空间几何特性可能会发生变化，引入系统性的误差，对其性能带来影响。但是，单天线本身的测距性能是不会受到天线阵列几何变化的影响的，我们可以利用这一点来约束算法，减少其误差并提高算法效率。

2. 模型假设与符号说明

2.1 模型假设

根据题意，我们作出如下假设：

- 1) 在每个场景下，均假设有 K 个需要确定的物体在雷达的探测范围内（以原点为中心半径 10 米以内、开口向上张开圆心角为 100° 的扇形区域）。
- 2) 等效虚拟天线阵列中的每一个单元依次进行发射和接收，由于周期极短，可以等效为同时发射和接收。
- 3) 基于依次发射和接收的设计，等效虚拟天线阵列中的每一个单元仅仅会接收到自己发射的信号的回波，而不会受到其他天线发射的信号的影响。
- 4) 应用场景中的运动物体移动相对较慢，在短时间内运动轨迹较为规律。

2.2 符号说明

表 1 相关符号说明

符号	含义
A_{TX}	信号发射功率
f_0	载频
γ	调频斜率
T	chirp 周期
j	虚数单位
R_0	距离
A_{RX}	信号接收功率
τ	接收时延
c	光速
δf	频谱的分辨率
δR	测距的分辨率
R_k	天线与目标之间的距离
θ_k	信号的发射角
N	采样数
K	目标物体数
Na	天线数
l_1	空间域窗口尺寸
l_2	时间域窗口尺寸
$step_R$	测距搜索步长
$step_\theta$	方位角搜索步长

3. 问题一模型建立与求解

3.1 模型建立

我们基于 2D-MUSIC 算法原理，设计了一套专门针对本 FMCW 雷达系统的 MUSIC 算法，以确保能够高分辨率地识别出无噪声条件下（多）物体位置。同时，单天线 FMCW 雷达通过快速傅里叶变换实现的测距功能也可以用于辅助该算法，提高算法效率。

3.11 FMCW 雷达信号基本形式

在此首先给出 FMCW 雷达信号的基本形式。FMCW 雷达的基本原理框图如图 1 所示。

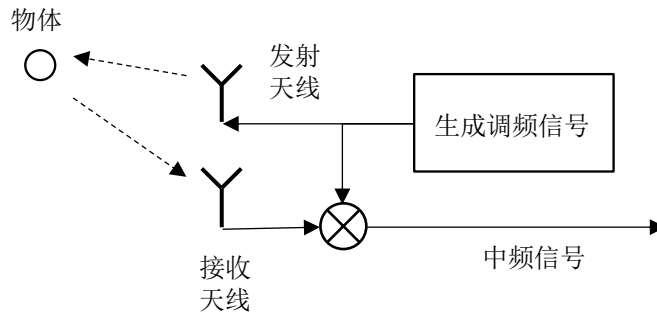


图 1 FMCW 雷达框图

假设在一个 chirp 周期 T 内，雷达发射信号的频率及发射波信号分别为：

$$f(t) = f_0 + \gamma t, 0 \leq t < T \quad (1)$$

$$s_{TX}(t) = A_{TX} \exp(j2\pi(f_0 t + \frac{\gamma}{2} t^2)), 0 \leq t < T \quad (2)$$

则雷达经过距离为 R_0 的物体反射后的接收信号为：

$$s_{RX}(t) = A_{RX} \exp(j2\pi(f_0(t - \tau) + \frac{\gamma}{2}(t - \tau)^2)), \tau \leq t < T \quad (3)$$

发射信号和接受信号在 $[\tau, T]$ 上有重叠，将这两个信号输入混频器，即可得到中频（IF）信号：

$$s_{IF}(t) = s_{TX}(t)s_{RX}^*(t) = A_{TX}A_{RX} \exp(j2\pi(\gamma\tau t + f_0\tau)), \tau < t \leq T \quad (4)$$

其中忽略了 τ^2 ($\tau \ll 1$) 项。

3.12 FMCW 快速傅里叶变换测距的基本原理

考虑对于单个天线，测量单一目标物体的情况。认为发射和接收天线相同的情况下，天线接收并进行相关运算之后的中频信号为：

$$s_{IF}(t) = A e^{j2\pi(\gamma\tau t + f_0\tau)} + w(t), 0 < t \leq T \quad (5)$$

可以看到，中频信号主要的频率分量在 $f_{IF} = \gamma\tau$ 处，如图 2 所示。而 $f_0\tau$ 主要影响信号的相位。

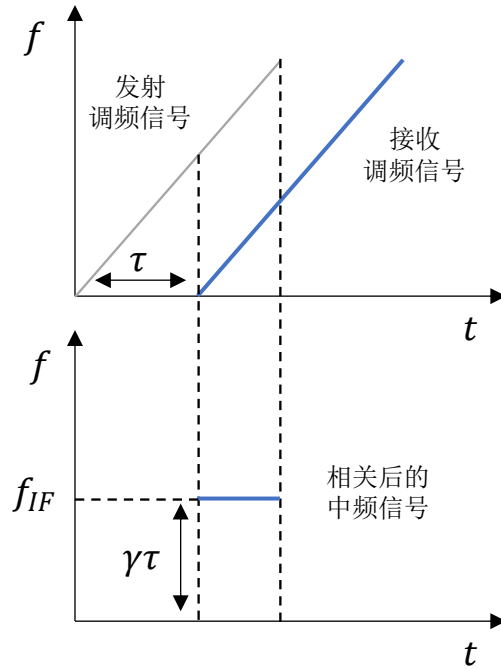


图2 FMCW 的 IF 信号频率与信号时延之间的关系

通过对采样后的中频信号进行离散傅里叶变换，可以在频谱上寻找功率的峰值 \hat{f}_{IF} ，该峰值可以直接用于计算信号传播的时延和距离。

$$\hat{R}_0 = c \cdot \frac{\hat{\tau}}{2} = c \cdot \frac{\hat{f}}{2\gamma} \quad (6)$$

如此获得的测距值只使用了中频信号的主频信息，没有考虑相位等信息，非常快速和直接；然而，其测距精度直接受限于频谱的分辨率，也即中频信号的采样率。

针对于本问题，一个 chirp 的信号时长为 $T = 3.2 \times 10^{-5}$ 秒，频谱的分辨率为 $\delta f = \frac{1}{T} = 3.125 \times 10^4$ 赫兹，调频斜率为 $\gamma = 78.986 \times 10^{12}$ 赫兹/秒，则测距的分辨率为

$$\delta R = c \cdot \frac{\delta f}{2\gamma} = 0.0593 \text{ (m)}$$

即利用朴素的频谱峰值进行测距，其测距分辨率仅为 6 厘米左右。

当考虑多个目标物体的时候，当不同物体之间与天线的距离差超过 6 厘米时，利用频谱峰值进行测距可以分辨出不同物体；而当不同物体相距较近或与天线的距离差较小时，利用频谱峰值进行测距无法将不同物体分辨出来。测距分辨率较低同样也限制了多天线利用测距交会来实现位置确定的可能性，尤其是在本问题中，不同等效天线之间的间距小于 10 厘米。因此，想要在本问题的情况下实现超分辨率及多物体的测距、测角，不能只利用频谱峰值这一信息，而是要充分利用天线阵列的信号特性，实现高精度的测量。

3.13 FMCW-MUSIC 算法基本原理

MUSIC 算法是一种基于空间谱估计的算法，特点是测向分辨率较高，在天线阵列信号处理领域占据着举足轻重的地位。基本的，MUSIC 算法通过一系列天线构成天线阵，

然后通过天线阵对空间信号进行接收。当完成阵列接收数据的获取后，构造协方差矩阵，并对其进行特征值分解，将数据空间分解为信号、噪声子空间；二者是相互正交的，利用这些正交特性可以构造空间谱峰。这一方法常用来估计波达角（Direction of Arrival, DOA）^[5]。在此基础上，如果空间谱的搜索空间是二维的，那么就可以统称为 2D-MUSIC 算法^[6]。典型地，在 DOA 估计中，2D-MUSIC 算法可用于同时估计入射波的方位角和俯仰角^[7]。

2D-MUSIC 算法可以用于不同的雷达信号调制方式。文献^[8]和文献^[9]针对于 FMCW 的雷达系统，构建了距离-方位角域的 2D-MUSIC 算法，可以用于同时测距和测角。通常情况下，2D-MUSIC 算法往往用于 MIMO 的天线阵列系统。而本问题所述的系统与一般的 MIMO 天线阵列系统存在一定的区别。在 MIMO 系统中，阵列中的每一个天线发射相互正交的信号，并且接收端同时接收所有天线发射的信号。对于 M 个发射天线， N 个接收天线的系统来说，总共有 $M \times N$ 个信号收发元素。而在本问题中，由于依次发射、接收的设计，每一个收发信号的天线接收的都是自己发射信号的回波，总共只有 N_a 个信号收发元素（ N_a 为天线数），如图 3 所示。

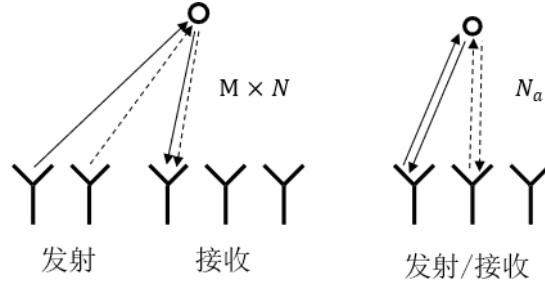


图 3 MIMO 阵列天线(左)与本题目天线系统(右)的区别

考虑到本系统的上述机制，阵列天线接收并处理得到信号可以表示为：

$$\mathbf{x}_{R,IF}(t) = \sum_{k=1}^K \mathbf{A} \cdot \text{diag}(\mathbf{a}(\theta_k))^2 \mathbf{s}_{IF}(t) \quad (7)$$

$$\mathbf{a}(\theta_k) = [e^{-j2\pi \frac{d}{\lambda} \sin(\theta_k)} \quad e^{-j2\pi \frac{2d}{\lambda} \sin(\theta_k)} \quad \dots \quad e^{-j2\pi \frac{N_a d}{\lambda} \sin(\theta_k)}]^T \quad (8)$$

$\mathbf{a}(\theta_k)$ 主要反映了由于天线间距以及与天线目标物体之间方位角所引起的距离/相位差，如图 4 所示。相比于一般的 MIMO 系统，上述的系统相对来说实际上更加简单。

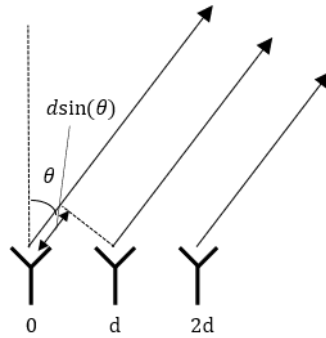


图 4 阵列天线时延示意

对于虚拟天线阵列中的第 p 个天线来说，对于目标物体 k ，其发射接收的时延可以表示为

$$\tau_k = \frac{2}{c} [R_k + p \cdot d \cdot \sin(\theta_k)] \quad (9)$$

可以看到该时延同时与天线与目标之间的距离 R_k 还有信号的发射角 θ_k 有关（在此不考虑物体运动引起的多普勒效应的影响）。

对于上一节所述的阵列天线收到的信号，我们考虑其在一个 chirp 间的所有 N 个采样，可以获得 $N_a \times N$ 的数据矩阵。数据矩阵的纵向表示空间上（天线阵列）的采样，横向表示时间上的采样。对于这个空间-时间域数据矩阵，使用一个 $l_1 \times l_2$ 的窗口对其进行扫描，如图 5 所示。

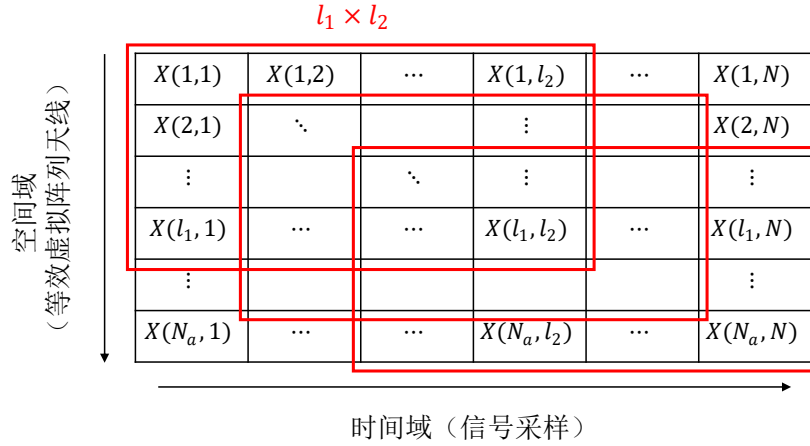


图 5 空间-时间域窗口扫描

这样我们可以获得 $p_1 \times p_2$ 个子矩阵，其中 $p_1 = N_a - l_1 + 1$ 属于信号的空间域， $p_2 = N - l_2 + 1$ 属于信号的时间域。每一个子矩阵反映了局部的空间与时间内信号的信息。对于每一个子矩阵，将其重新排列为一个 $l_1 l_2 \times 1$ 的向量，并将这些向量堆叠为新的数据矩阵 $\tilde{\mathbf{X}}$ 。新的数据矩阵 $\tilde{\mathbf{X}}$ 的大小为 $l_1 l_2 \times p_1 p_2$ 。按照下式计算平滑之后的协方差矩阵：

$$\mathbf{C} = \frac{1}{2p_1 p_2} [\tilde{\mathbf{X}} \tilde{\mathbf{X}}^H + \mathbf{J}(\tilde{\mathbf{X}} \tilde{\mathbf{X}}^H)^* \mathbf{J}] \quad (10)$$

$$\mathbf{J} = \text{flip}(\mathbf{I}) \quad (11)$$

其中， \mathbf{J} 为 $l_1 l_2 \times l_1 l_2$ 的转递矩阵，用于对矩阵进行翻转：

对协方差矩阵 \mathbf{C} 进行奇异值分解（Singular Value Decomposition, SVD），可以得到该协方差矩阵的特征值和特征向量。理论上， \mathbf{C} 的前 K 维（等于目标物体数量）特征值和特征向量，对应于信号子空间，而后 $l_1 l_2 - K$ 维对应于噪声子空间。噪声子空间的特征向量 $\tilde{\mathbf{W}}$ 与天线发射的信号有相互正交的关系，可以利用该特性构造空间谱中的峰值。最终，2D-MUSIC 的空间谱可以如下进行构造：

$$S(\tau_i) = \frac{1}{\mathbf{a}^H(\tau_i) \tilde{\mathbf{W}} \tilde{\mathbf{W}}^H \mathbf{a}(\tau_i)} \quad (12)$$

其中， $\mathbf{a}(\tau_i)$ 为导向矢量，其形式为

$$\mathbf{a}(\tau_i) = [e^{-j2\pi[f_0 \tau_i(0)]}, e^{-j2\pi[f_0 \tau_i(0) + \gamma \tau_i(0) \frac{1}{f_s}]}, \dots, e^{-j2\pi[f_0 \tau_i(0) + \gamma \tau_i(0) \frac{(l_2-1)}{f_s}]}]$$

$$e^{-j2\pi[f_0\tau_i(1)]}, e^{-j2\pi\left[f_0\tau_i(1)+\gamma\tau_i(1)\frac{1}{f_s}\right]}, \dots, e^{-j2\pi\left[f_0\tau_i(1)+\gamma\tau_i(1)\frac{(l_2-1)}{f_s}\right]},$$

$$\dots,$$

$$e^{-j2\pi[f_0\tau_i(l_1-1)]}, e^{-j2\pi\left[f_0\tau_i(l_1-1)+\gamma\tau_i(1)\frac{(l_2-1)}{f_s}\right]}, \dots, e^{-j2\pi\left[f_0\tau_i(l_1-1)+\gamma\tau_i(1)\frac{(l_2-1)}{f_s}\right]}\top$$

时延 τ_i 对应于具体的 R 和 θ 根据公式（9）进行计算和取值。即，对于每一个 (R, θ) 的数值对，可以计算出一个对应的 $S(\tau_i)$ 。如此一来便可以遍历 (R, θ) ，绘制出二维的距离-方位角的响应图。在响应图上，可以使用常用的搜索局部最大值的方法识别出目标物体。

在此给出具体的数据处理流程图如图 6 所示。

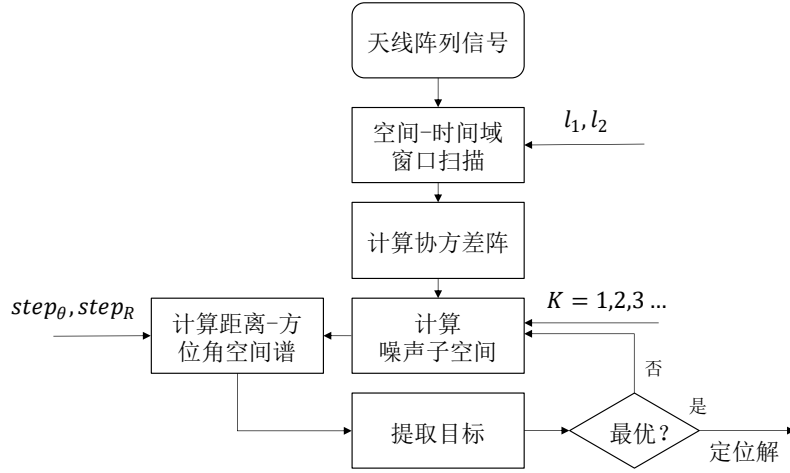


图 6 FMCW-MUSIC 数据处理流程图

为了发挥出 FMCW-MUSIC 的最优性能，需要正确地选取如下的参数：

1) 目标物体数 K 。

如果目标物体数 K 不正确，会引起信号协方差阵在进行分解后信号子空间和噪声子空间的错误选取，从而影响目标物体探测的正确性。

2) 空间-时间域窗口在空间（天线阵列）维度上的长度 l_1 以及时间维度上的长度 l_2 。

对于针对于 FMCW 阵列天线雷达的 2D-MUSIC 算法来说，空间-时间域窗口的选取至关重要，对于探测的精确性以及处理效率有很大的影响。

3) 测距的搜索步长 $step_R$ 和方位角的搜索步长 $step_\theta$ 。

在距离-方位角的二维空间进行搜索时，如果搜索的步长太大，可能会漏检响应大的值；如果搜索的步长太小，会直接影响算法的实时性能。

其中，窗口大小 l_1, l_2 以及搜索步长 $step_R, step_\theta$ 需要根据具体的系统以及应用需求进行参数调优，选定后即可保持不变；而目标物体数 K 和具体场景相关，需要进行在线的估计。现有文献有 AIC, MOL, AICc 等方法^[9]可以实现 K 的估计；而在此，我们选取一种简单遍历选取最优结果的方法进行 K 值的确定，其算法简单描述如下：

- ① K 从 1 开始进行遍历；
- ② 使用 K 当前的取值，进行 FMCW-MUSIC 处理提取目标物体；
- ③ 若当前提取的目标物体数小于 K ，则 $K - 1$ 为最优的 K 值；若当前提取目标数等于 K ，则 $K = K + 1$ ，回到第二步。

经过上述算法，可以得到最大的使得 K 与提取目标数保持一致的取值，即为最优的 K 。

3.2 问题求解

3.2.1 数据预处理

基于 3.12 节 FMCW 雷达利用频谱峰值进行测距的基本原理，考虑先对等效虚拟天线阵列中各天线的中频信号采样数据进行离散傅里叶变换并观察其频谱。首先，对第一个天线（#1）信号进行处理，获得频谱如图 7 所示。

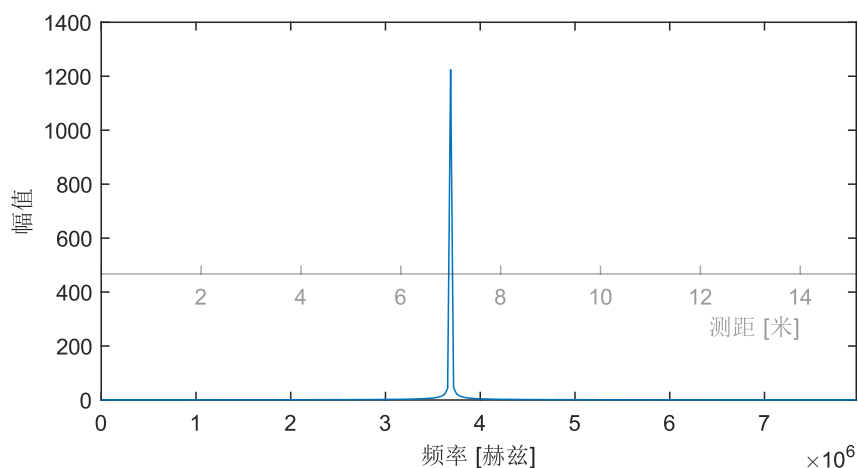


图 7 天线#1 中频信号的频谱（及对应的测距信息）

从图中可以看到，天线#1 的信号频谱上在 $f = 3.687500 \times 10^6$ 赫兹处存在一个非常明显的峰值。结合 3.12 节中的说明，可以判断，在与天线#1 距离 $\hat{R} = c \cdot \frac{f}{2\gamma} \approx 7.00$ 米（分辨率为 0.0593 米）处，存在目标物体。

同时对所有 86 个等效天线的中频信号进行快速傅里叶变换，可以获得如图 8 所示的频谱图像。从图像中可以看到，86 个等效天线信号频谱的峰值是一致的，这一方面是因为频谱的分辨率有限，另一方面也是因为阵列天线的排布非常紧密，其与目标物体之间的测距值几乎不存在差异。

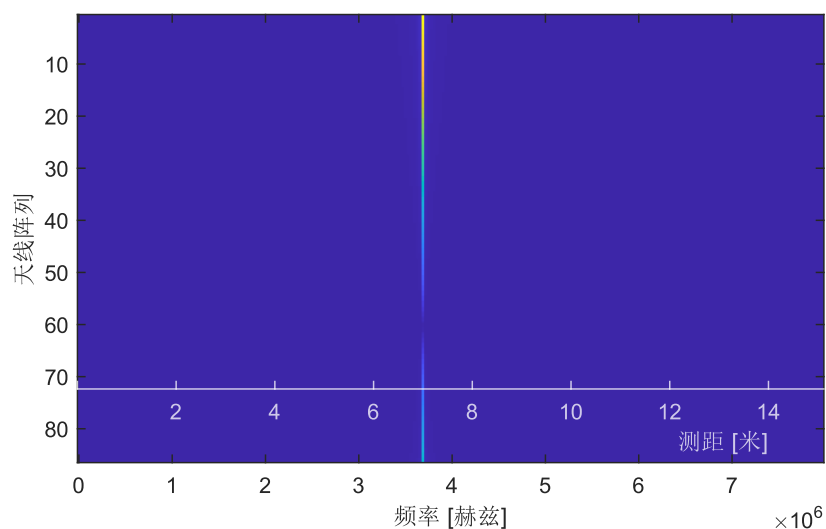


图 8 阵列天线中频信号的频谱（及对应的测距信息）

基于上述结果，可以明确的是，在距离阵列天线大约 7.00 米处存在目标物体，但目标物体的确切测距值、方位角以及物体数目，都无法得到。

3.22 使用 FMCW-MUSIC 算法进行目标定位

基于此，考虑进一步使用 3.13 节所述的 FMCW-MUSIC 算法对信号数据进行处理。

在此，我们首先对空间-时间域窗口的尺寸 $[l_1, l_2]$ 以及搜索步长 $[step_R, step_\theta]$ 选取如下较为保守的参数取值

$$\begin{cases} l_1 = 10 \\ l_2 = 10 \\ step_R = 0.1 \\ step_\theta = 0.25^\circ \end{cases}$$

并在下面给出 FMCW-MUSIC 算法确定目标数 K 的具体过程。

3.23 目标物体数 K 值的确定

首先选取 $K = 1$ ，绘制 FMCW-MUSIC 的距离-角度二维空间响应图如图 9 所示。

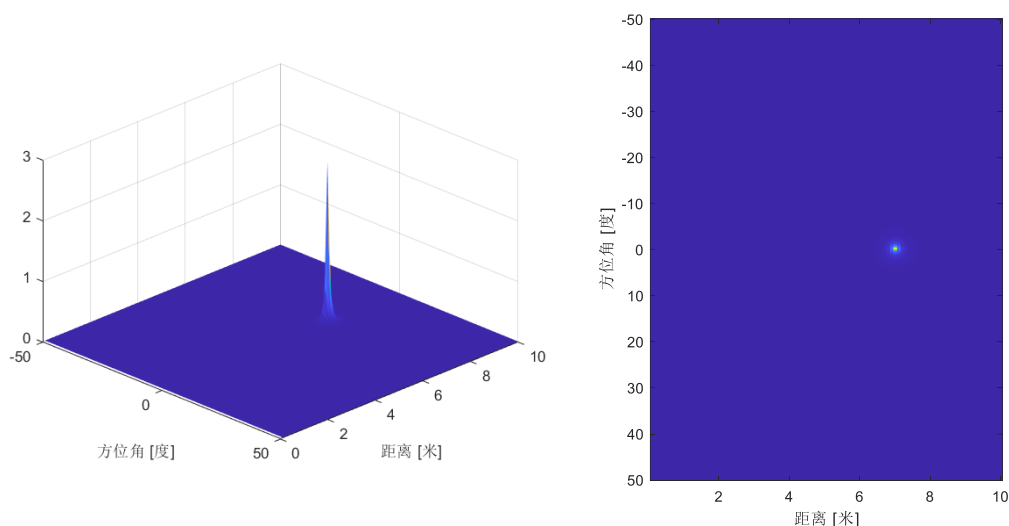


图 9 选取目标物体数 $K = 1$ 时的距离-方位角空间谱

从图中可以看到，二维空间谱上出现了一个显著的峰值，其对应的距离恰恰在 7 米附近，与中频信号频谱峰值对应的测距值是接近的。考虑到实际上可能存在更多的目标物体。因此，继续对更大的 K 值进行尝试。

选取 $K = 2$ ，绘制 FMCW-MUSIC 的距离-角度二维空间响应图如图 10，图 11 所示。

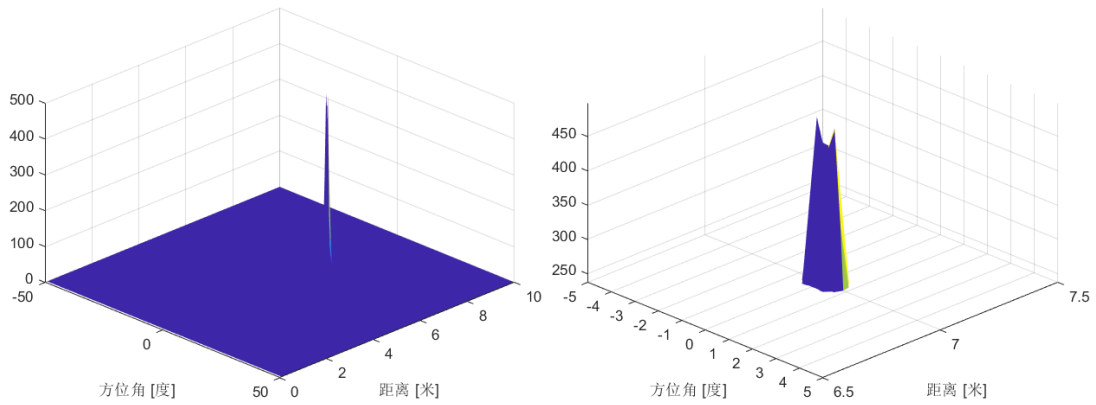


图 10 选取目标物体数 $K = 2$ 时的距离-方位角空间谱（曲面）

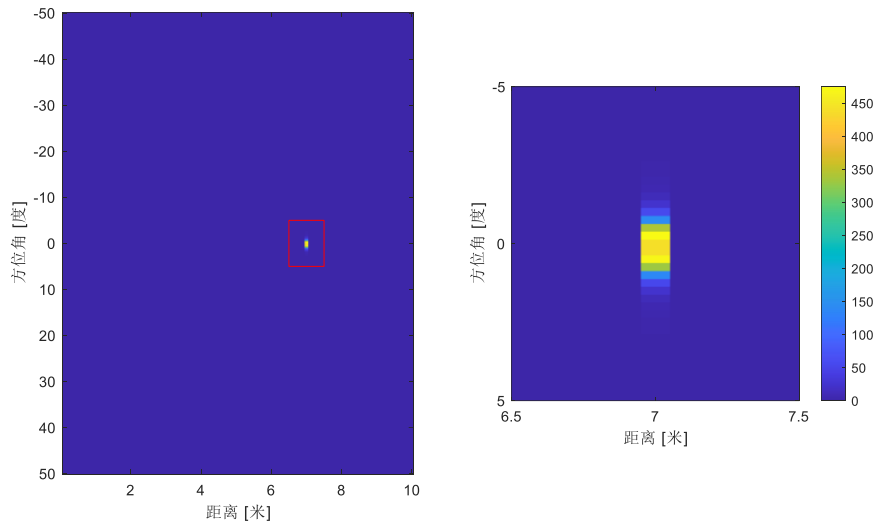


图 11 选取目标物体数 $K = 2$ 时的距离-方位角空间谱

可以看到，当选取 $K = 2$ ，二维空间谱上在距离约等于 7 米，方位角约等于 0 度的位置出现了两个非常邻近的波峰。所选取的 0.25° 的角度搜索步长勉强可以分辨出两个波峰的存在。考虑到如果只存在一个目标物体，不应该呈现这样的两个波峰；因此，基本可以推翻 $K = 1$ 得到的只有一个目标物体的结论。

进一步的，选取 $K = 3$ 、 $K = 4$ ，绘制 FMCW-MUSIC 的距离-角度二维空间响应图如图 12，图 13 所示。

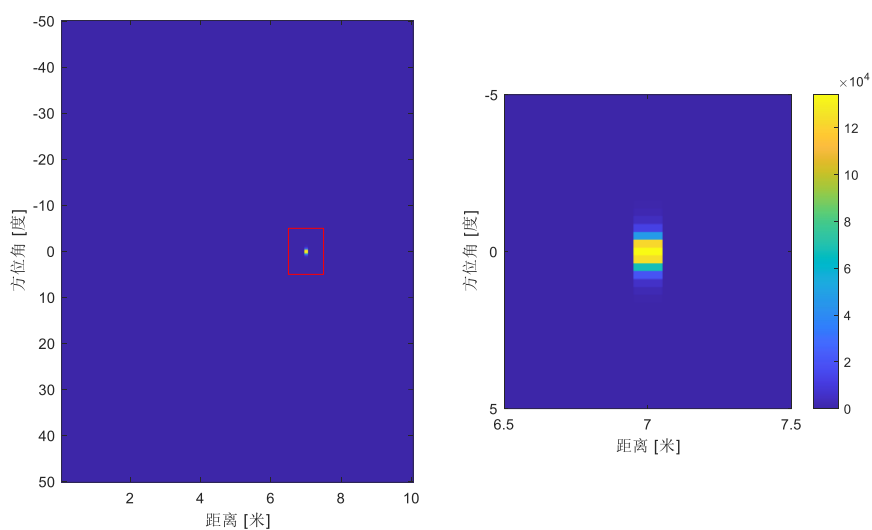


图 12 选取目标物体数 $K = 3$ 时的距离-方位角空间谱

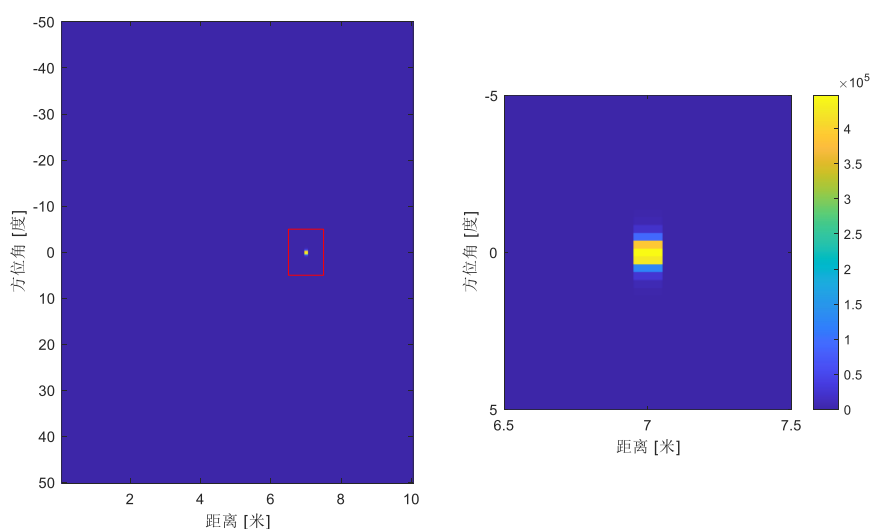


图 13 选取目标物体数 $K = 4$ 时的距离-方位角空间谱

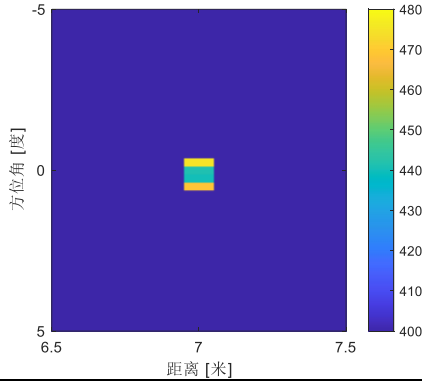
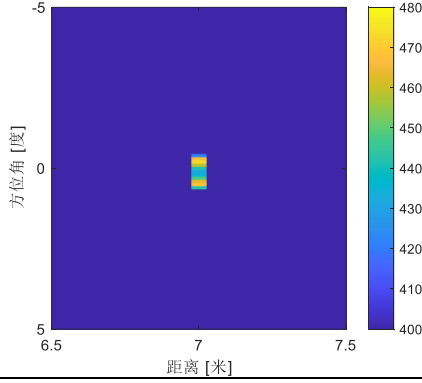
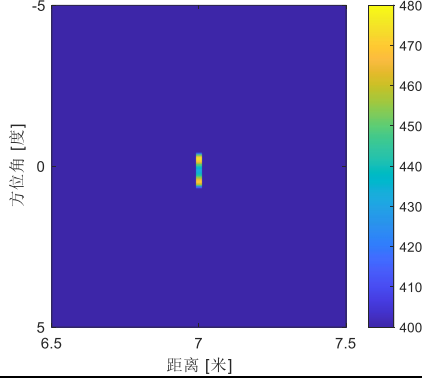
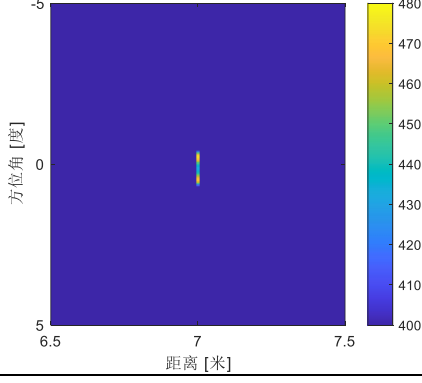
可以发现，在 $K = 3, 4$ 的情况下，无法从二维空间谱上探测到更多的波峰了。这一定程度上说明，在现有的分辨率条件下，无法从当前的信号数据中分辨出两个以上的目标物体。因此，在现有条件下，选取最优的 $K = 2$ 即目标物体为两个的情况作为 K 的估值。

3.24 搜索步长的选取

尽管默认的参数已经可以探测出物体并对其进行定位，但是在此基础上，我们希望尽可能提高测距和测角的分辨率，从而获得两个邻近的目标物体的精确位置。为此，可以考虑选择更加精细的步长进行距离-方位角二维空间上的搜索。

我们选择了多种搜索步长方案进行 FMCW-MUSIC 的处理，结果在表 2 中给出。为了更清晰的显示出波峰，空间谱的幅值显示范围相比图 11 进行了调整。

表 2 不同搜索步长设置对应的距离方位角二维空间谱

搜索步长设置	距离-方位角二维空间谱
$\begin{cases} step_R = 0.1\ m \\ step_\theta = 0.25^\circ \end{cases}$	
$\begin{cases} step_R = 0.05\ m \\ step_\theta = 0.1^\circ \end{cases}$	
$\begin{cases} step_R = 0.02\ m \\ step_\theta = 0.05^\circ \end{cases}$	
$\begin{cases} step_R = 0.01\ m \\ step_\theta = 0.05^\circ \end{cases}$	

从图表中可以观察到，在仿真信号没有噪声的情况下，如果搜索的步长足够小，FMCW-MUSIC 的测距分辨率可以达到 1 厘米以下，而测角分辨率可以达到 0.1° 以下，可以认为实现了目标物体的超分辨定位。为了实现最好的定位精度，我们采用 $(0.01\text{m}, 0.05^\circ)$ 搜索步长的方案，对目标物体进行定位。

3.25 目标物体定位结果

通过在距离-方位角二维空间谱上寻找峰值点的位置，可以确定目标物体的坐标。在本问题中，考虑到二维空间谱的平滑性，我们只需要使用一个简单的局部极值搜索方法来探测代表目标物体位置的样点，算法描述如下：

- ① 统计二维平面上的最大值 S_{max} ；
- ② 遍历二维平面，对于每一个样点，若该样点的值大于 $S_{max}/2$ 且该值比所有邻近点上的值都大，则将其作为目标物体的候选点。

在二维空间谱上使用上述算法探测极值点，获取目标物体的坐标并在图中标注，如图 14，图 15 所示。

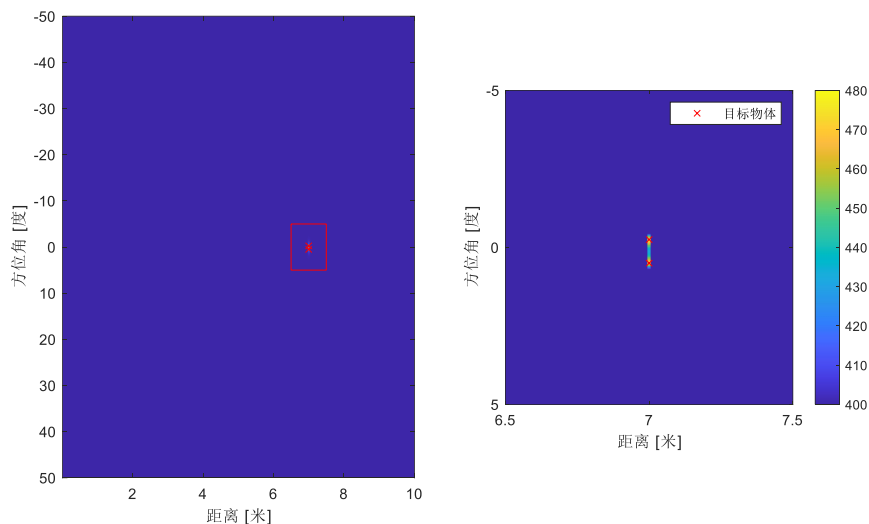


图 14 问题 1 目标物体定位结果（距离-方位角平面）

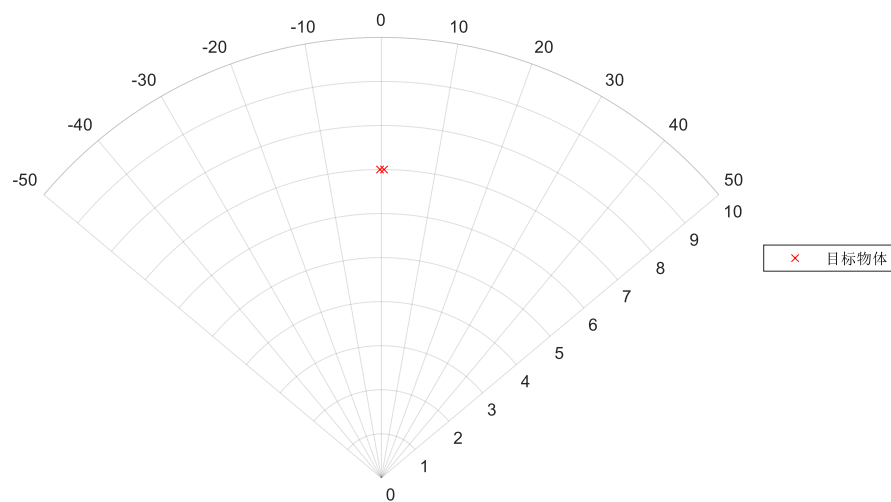


图 15 问题 1 目标物体定位结果（极坐标系）

可以算出，两个目标物体的极坐标分别为 $(-0.25^\circ, 7.00\text{ m})$ 和 $(0.5^\circ, 7.00\text{ m})$ ，其精度在径向为 1 厘米左右，在切向为 1 厘米 $(0.1^\circ/180^\circ \times \pi \times 7.00)$ 左右。

3.26 FMCW-MUSIC 算法搜索效率的优化

需要注意的是，在不断减小步长时，虽然探测的分辨率得以增加，但同时二维空间谱搜索的样点数也从一开始的 40100 个样点提升至 2001000 个样点，提升了约 500 倍；在 intel i5-10210U + matlab 2021b 的计算平台上（单线程），计算一个 chirp 的用时达到了 50 多秒，这一点是难以接受的，直接影响到算法的实用性。对此，考虑到以下两点：

- 1) 目标物体的距离可以由快速傅里叶变换得到的 FMCW 中频信号频谱峰值快速获得，精度在数厘米，不需要对整个距离空间进行搜索；
- 2) 二维空间中目标物体较为稀疏，大部分样点都与目标物体无关，基本没有响应值。因此，可以采取以下的策略来提高二维搜索的效率：

- 1) 首先对需要处理的信号进行快速傅里叶变换，获取粗略的目标物体测距值后，将距离搜索的范围限定在近似测距值的附近。
- 2) 使用一种多层次分辨率的搜索方法，首先使用较大的步长在空间中进行搜索，搜索到较大的响应值之后再在附近进行较小步长的搜索；类似的策略可保持多个层级，从而显著的降低搜索的复杂度。

改进得到的 FMCW-MUSIC 算法的数据处理流程图如图 16 所示。

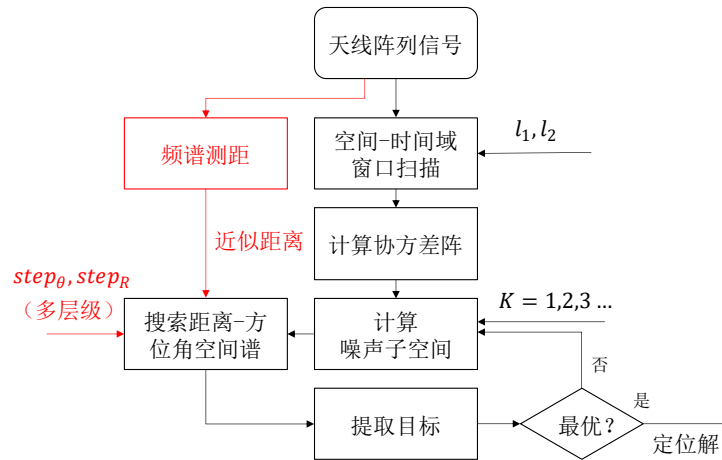


图 16 改进后的 FMCW-MUSIC 数据处理流程图

在进行多层次分辨率搜索时，为了防止漏检，同样需要恰当的选择各层级的步长。我们在此以三层级搜索，距离步长为 $[0.1, 0.02, 0.01]$ （米），方位角步长为 $[1, 0.2, 0.05]$ （°）为例，给出进行多层次搜索的示例，如图 17 所示。在距离-方位角的二维空间谱上进行搜索时，首先以较大的步长进行搜索（搜索层级 1），计算的样点比较稀疏；当搜索到较大的响应值时（超过阈值 1），在该样点的附近进入搜索层级 2，进行更细致的搜索；当搜索的响应值超过阈值 2 时，在进入搜索层级 3，进行最精细的搜索。通过使用这样的多层次分辨率的搜索，可以避免在整个空间谱范围内进行细致的搜索，大大减少多余的计算量。

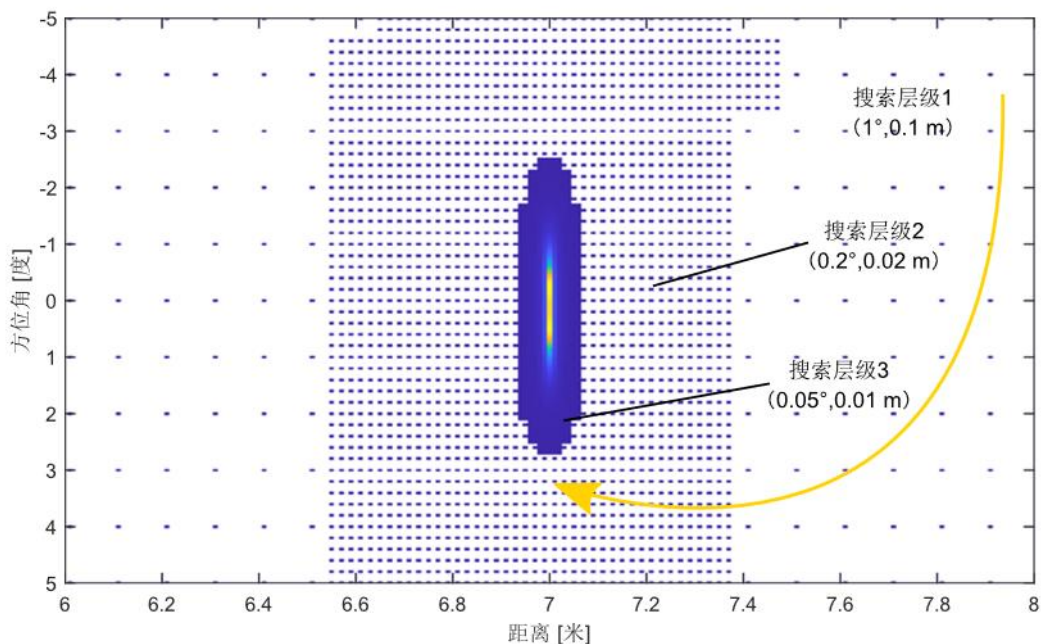


图 17 FMCW-MUSIC 多层级分辨率搜索示意图

为了评估频谱峰值距离辅助和多层级分辨率对于算法性能的提升，我们选择了不同的步长方案，对于该组数据进行重复测试，以评估不同策略下效率方面的提升，结果如表 3 所示。

表 3 不同方案参数说明

序号	使用 FFT 距 离	距离步长 (三级) [米]	方位角步长 (三级) [°]	采样点计 算次数	搜索耗时
1	不使用	$[-, -, 0.01]$	$[-, -, 0.05]$	2001000	54.7 秒
2	使用	$[-, -, 0.01]$	$[-, -, 0.05]$	40020	0.93 秒
3	不使用	$[0.1, 0.02, 0.01]$	$[1, 0.2, 0.05]$	26411	0.77 秒
3	不使用	$[0.2, 0.02, 0.01]$	$[1, 0.2, 0.05]$	18818	0.59 秒
5	不使用	$[0.1, 0.02, 0.01]$	$[1, 0.5, 0.05]$	23210	0.66 秒
6	使用	$[0.1, 0.02, 0.01]$	$[1, 0.2, 0.05]$	5552	0.22 秒

进行上述的评估时，确保了每次搜索的最终结果都是完全一致的。可以发现，无论是使用频谱峰值获取的距离进行限定，还是使用多层级分辨率的搜索方式，都会带来非常明显的效率提升。方案 6 相比方案 1 效率提升了 248.6 倍。

有一点需要注意：从表中可以看到，搜索耗时与采样点计算次数并不完全成线性的关系，这一点可能是由于 matlab 运行逻辑代码的效率较低，造成了多余的耗时。如果使用 C/C++ 等静态语言进行编译，程序的效率还会有非常大的提升。

4. 问题二模型建立与求解

4.1 模型建立

在问题一中，由于数据是无噪声的仿真数据，FMCW-MUSIC 窗口尺寸参数 l_1 ， l_2 的选取对于结果的影响很小。实际上，在 FMCW 阵列天线雷达的 2D-MUSIC 算法中，在有噪声的条件下，用于扫描空间-时间域数据矩阵的窗口尺寸 $l_1 \times l_2$ 会明显影响算法的性能。因此，对于问题二来说，为了保证算法在环境噪声影响下的分辨率和信噪比，需要考虑增大窗口尺寸；同时，增大窗口尺寸会带来更大维度的矩阵乘法运算，明显地增加算法的耗时。尽管可以使用 3.26 小节中的加速策略大大增加了二维搜索的效率，还是需要尽可能地确保窗口尺寸不要过大以至于影响算法的实用性。

在本问题中，考虑针对于环境噪声场景，对 FMCW-MUSIC 窗口尺寸参数 l_1 ， l_2 进行调优，使其在该场景下提供较高分辨率定位效果的同时保持较高的算法效率。

4.2 模型求解

4.2.1 数据预处理

使用快速傅里叶变换对天线信号数据进行预处理，天线#1 的频谱如图 18 所示。

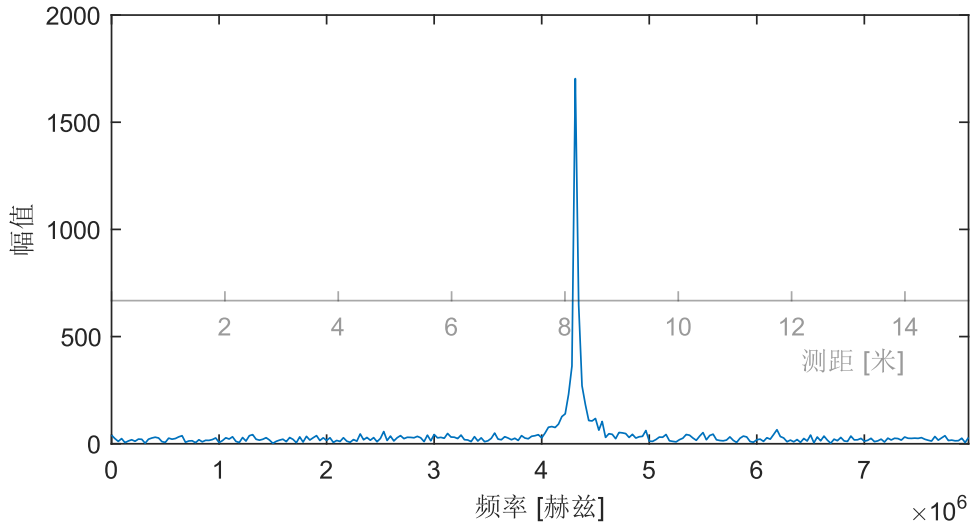


图 18 噪声环境下天线#1 中频信号的频谱（及对应的测距信息）

从频谱中可以看到环境噪声的存在，但是 FMCW 的中频信号峰值还是很锐利，可以用于获取距离的近似值。

接下来绘制天线阵列信号的频谱，如图 19 所示。天线阵列信号的频谱上存在一个明显的峰值，且各天线是一致的；基本情况和问题一情况类似，不过能看到中频以外频率上存在一定水平的噪声。

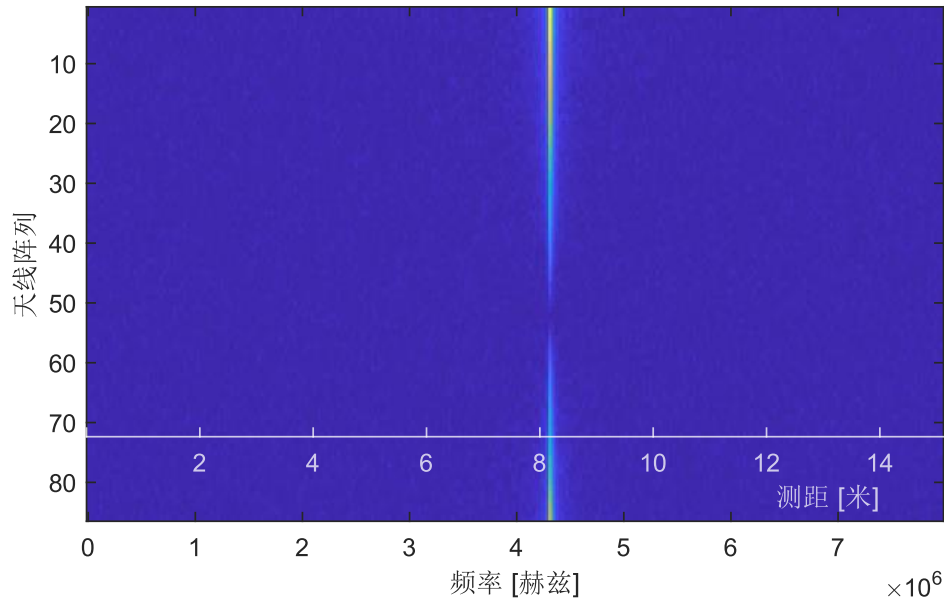


图 19 噪声环境下阵列天线中频信号的频谱（及对应的测距信息）

4.22 噪声环境下 FMCW-MUSIC 窗口参数 l_1, l_2 的调优

接下来，使用 FMCW-MUSIC 算法对阵列天线信号数据进行处理，求解目标物体位置。

首先使用 $l_1 = 10$ ， $l_2 = 10$ 的空间时间域窗口长度以及小步长进行搜索。在对目标物体数 K 进行确定的过程中，当 $K = 2$ 时发现如图 20 的情况。

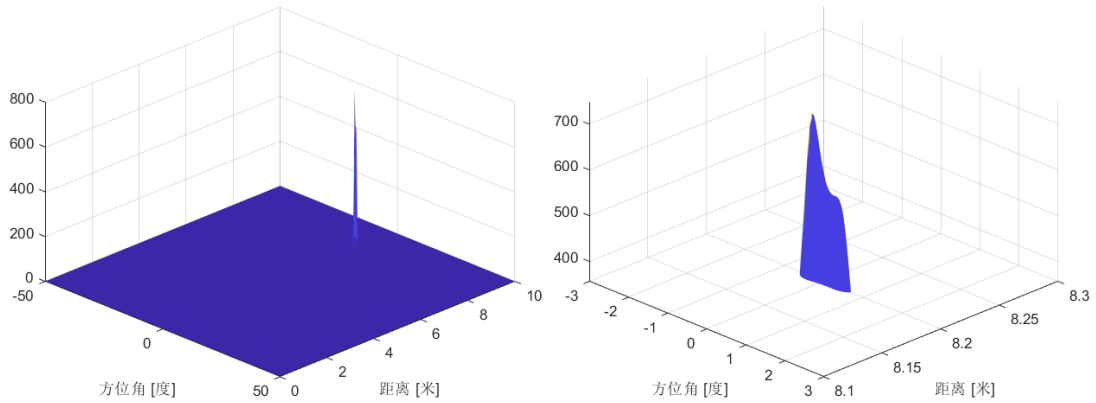


图 20 选取窗口大小为（10，10）时的距离-方位角空间谱（曲面）

从图中可以看到，距离-方位角空间谱上出现了两个波峰，但是比较扭曲，其中一个波峰的信噪比很低。可以推测，在低信噪比的情况下，信号在经由 FMCW-MUSIC 处理之后出现了信息一定程度上的失真。所以，我们考虑增大空间时间域窗口长度来提高算法在噪声环境下的表现。

使用 $l_1 = 20$ ， $l_2 = 20$ 的空间-时间域窗口尺寸进行 FMCW-MUSIC 的处理，结果如图 21 所示。

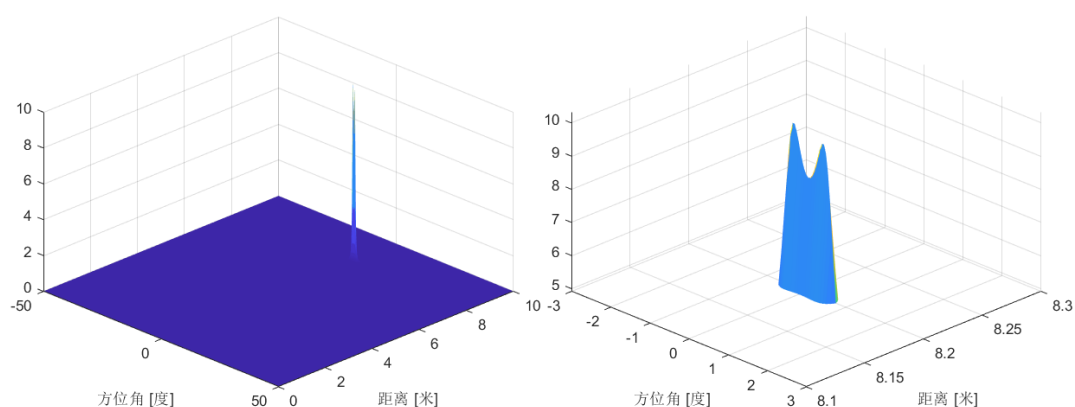


图 21 选取窗口大小为 (20, 20) 时的距离-方位角空间谱 (曲面)

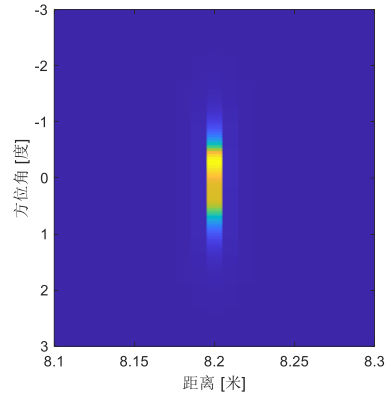
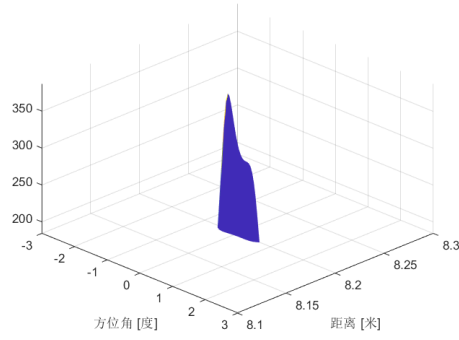
从图中看到, 在使用了 $l_1 = 20$, $l_2 = 20$ 的窗口之后, 空间谱上的波峰明显变得更清晰了, 可以直接确定多目标物体的位置。不难看出, 窗口大小的选择对于 FMCW-MUSIC 在环境噪声下的分辨效果有直接的影响。

为了进一步选取当前硬件系统和环境噪声下最优的窗口参数, 我们选取了不同的窗口方案, 分别观察波峰的情况, 结果如表 4 所示。

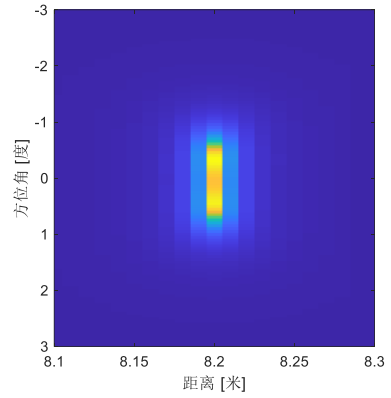
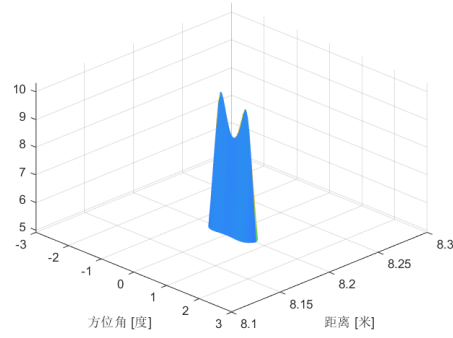
表 4 不同窗口方案结果

搜索步 长设置 (l_1, l_2)	距离-方位角二维空间谱 (曲面)	距离-方位角二维空间谱 (平面栅格)
(10,10)		
(20,10)		

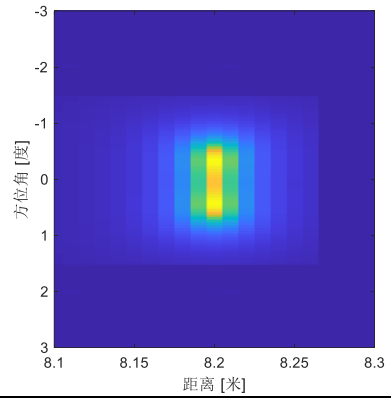
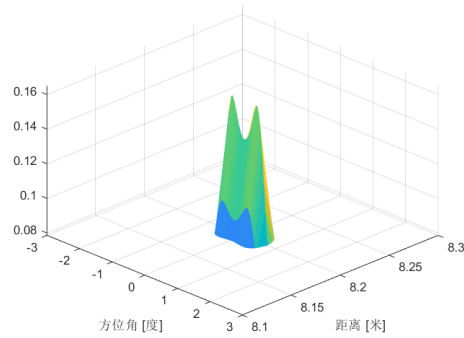
(10,20)



(20,20)



(40,40)



从表中可以看到， l_1 参数即空间域上窗口长度的选取，对于测角的分辨率有着比较关键的影响。当选取 $l_1 = 10$ 时，无法明确地分辨出两个波峰；而当选取 $l_1 = 20, 40$ 时，均可以较好的分辨出两个波峰。而时间域上窗口大小参数 l_2 的选取，在本题的情况中，对于测距和测角都没有明显的改善。

除了分辨能力的评估之外。同样对不同窗口尺寸下的算法效率进行评估。四种搜索方案的耗时在表 5 中给出。（评估效率时使用了傅里叶变换频谱峰值的测距值进行了距离范围的限定，为了控制计算样点数目相同没有采用多层级分辨率方法进行加速搜索。）

表 5 不同搜索方案的耗时

搜索步长设置 (l_1, l_2)	搜索耗时 (秒)
(10,10)	0.91
(20,10)	1.88
(10,20)	1.90
(20,20)	8.61
(40,40)	170.15

从表中可以看到，增大窗口大小会明显提高搜索算法的计算量。实际应用时，需要在精度和效率上进行取舍。在本问题的应用场景中，选取 $l_1 = 20, l_2 = 10$ 即可在计算量相对较小的情况下满足精度的条件。

在当前参数方案的基础上，进行 K 值的估计和目标物体的定位，可以确定目标物体的数量为 2，在此省略具体的过程。

4.23 噪声环境下的目标物体定位

在选定了本场景下窗口大小 l_1, l_2 的取值之后，使用 FMCW-MUSIC 进行目标物体的探测和定位。定位结果如图 22 和图 23 所示。

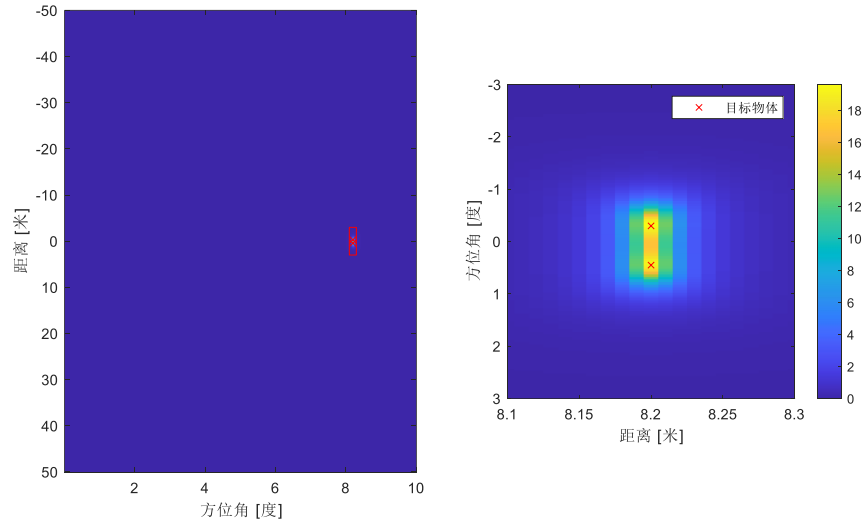


图 22 问题 2 目标物体定位结果（距离-方位角二维平面）

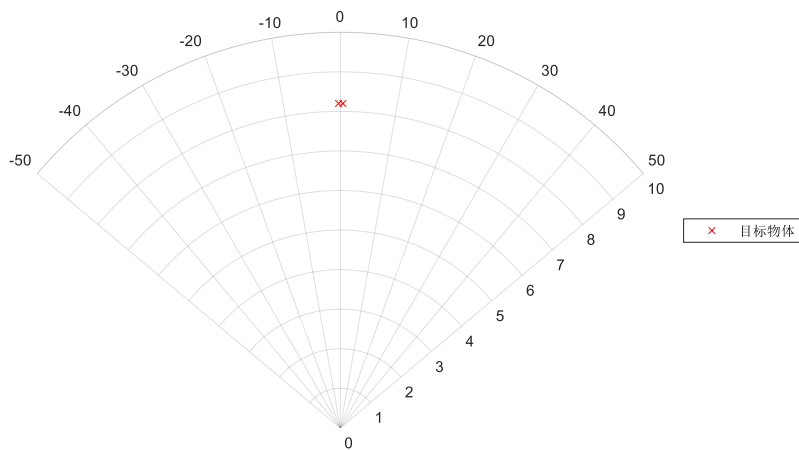


图 23 问题 2 目标物体定位结果（极坐标系）

定位结果中，目标物体的极坐标分别为 $(-0.3^\circ, 8.20 \text{ m})$ 以及 $(0.45^\circ, 8.20 \text{ m})$ ，其精度在径向为 1 厘米左右，在切向为 1 厘米 $(0.1^\circ/180^\circ \times \pi \times 8.20)$ 左右。在选取了合适的参数之后，环境噪声对于 FMCW-MUSIC 算法进行目标定位的精度影响较小。

5. 问题三模型建立与求解

5.1 模型建立

经过前两个问题对于算法的优化，当前的 FMCW-MUSIC 算法已经可以在有环境噪声的情况下进行较为高效且精确的瞬时定位了。而在需要在一段时间内对目标物体进行连续定位的场景下，考虑到物体在空间上的变化不会太大，可以充分利用物体运动连续性的这一条件，对算法进行进一步的提升。

具体来说，对于前一个 chirp 探测出来的目标物体，根据一定的运动模型（如匀速模型）可以对其在当前 chirp 时的空间位置进行一个预测；基于这个预测的位置，一方面可以减少二维搜索的范围，另一方面可以利用空间谱的平滑特性使用梯度下降法^[10]进一步进行加速搜索的过程。通过充分利用物体运动的连续性，不仅可以快速确定新一时刻的目标位置，还可以同时完成前后目标之间的关联，从而拥有更好的性能和效率。

在此对考虑物体运动连续性情况下 FMCW-MUSIC 目标跟踪算法做出具体的说明。利用一帧数据中的多个 chirp 对目标进行连续的定位，具体过程如下：

1) 第 1 个 chirp

处理第 1 个 chirp 的数据时，由于不存在物体位置先验信息，处理与之前的 FMCW-MUSIC 相同，同样需要确定出物体数目 K ，以及对物体的具体位置进行定位（测距、测角）。设第一个 chirp 目标物体 k 的测距和测角值为 $(\hat{R}_{k,1}, \hat{\theta}_{k,1})$ 。

2) 第 1 个 chirp 到第 2 个 chirp

在处理第 2 个 chirp 时，考虑到目标物体在短时间内位移较小，可以用第一个 chirp 时探测出目标物体的位置来预测当前时刻目标物体的位置。不妨假设在该系统的应用场景下，两个 chirp 之间目标物体的运动不会超过 0.1 米，则目标物体 k 在当前时刻的搜索范围为

$$\begin{aligned}\hat{R}_{k,2} &\in [\hat{R}_{k,1} - 0.1, \hat{R}_{k,1} + 0.1) \\ \hat{\theta}_{k,2} &\in \left[\hat{\theta}_{k,1} - \frac{0.1}{\hat{R}_{k,1}} \times \frac{180}{\pi}, \hat{\theta}_{k,1} + \frac{0.1}{\hat{R}_{k,1}} \times \frac{180}{\pi} \right)\end{aligned}$$

可以看到搜索范围被大大的限定了。不仅如此，FMCW-MUSIC 距离-方位角空间谱在局部的平滑性，使得在位置初值良好的情况下，通过梯度下降法快速找到最优解成为了可能。其搜索范围发生了如下的变化。

整个测距测角平面 \rightarrow 局部的测距测角平面 \rightarrow 到达极值点的最短路径

搜索的复杂度从全局二维平面变为局部二维平面最后变为了一维的路径，复杂度得以大大降低。

3) 第 n 个 chirp 到第 $n+1$ 个 chirp

第 n 个 chirp 到第 $n+1$ 个 chirp 的搜索过程与第 1 个 chirp 到第 2 个 chirp 的过程类似，但是可以进一步使用近似的“匀速”模型，基于物体短时间内运动的连续性获取更好的位置初值，即

$$\hat{R}_{k,n+1} = \hat{R}_{k,n} + (\hat{R}_{k,n} - \hat{R}_{k,n-1})$$

$$\hat{\theta}_{k,n+1} = \hat{\theta}_{k,n} + (\hat{\theta}_{k,n} - \hat{\theta}_{k,n-1})$$

可以期望，在物体运动较为规律的情况下，使用匀速运动模型提高位置初值的精度，可以进一步大大减少搜索需要计算的样点范围，尤其是对于梯度下降法而言。

5.2 模型求解

5.2.1 数据基本情况分析

首先对信号进行快速傅里叶变换，观察一帧数据中不同 chirps 的中频信号频谱情况（选取第 1、16、32 个 chirp），如图 24 所示。可以看到，在第一个 chirp，频谱上显示出一个峰值；而在第 16 个和第 32 个 chirp 上，频谱上显示出 2 个峰值。显然，这可以反映出，探测范围中存在多个物体，且发生了位置的动态变化。

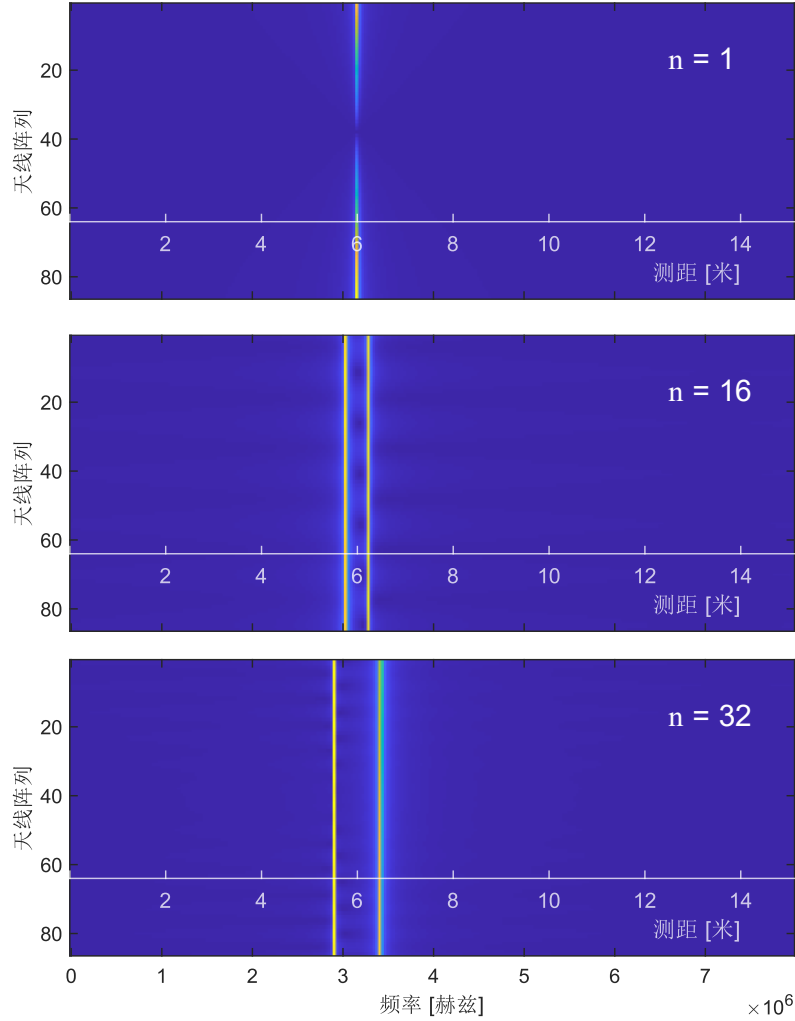


图 24 一帧信号数据中不同 chirps 的频谱情况（选取第 1、16、32 个 chirp）

5.22 不考虑物体运动连续性情况下的 FMCW-MUSIC 定位

使用问题二中的系统参数，首先对第一个 chirp 的数据进行处理，利用简单遍历的方法检测目标物体 K 的取值，可以得到目标物体 $K = 2$ 。之后，对一帧的 32 个数据进行依次处理。第 1、16、32 个 chirp 的 FMCW-MUSIC 空间谱以及物体定位结果如图 25 所示。

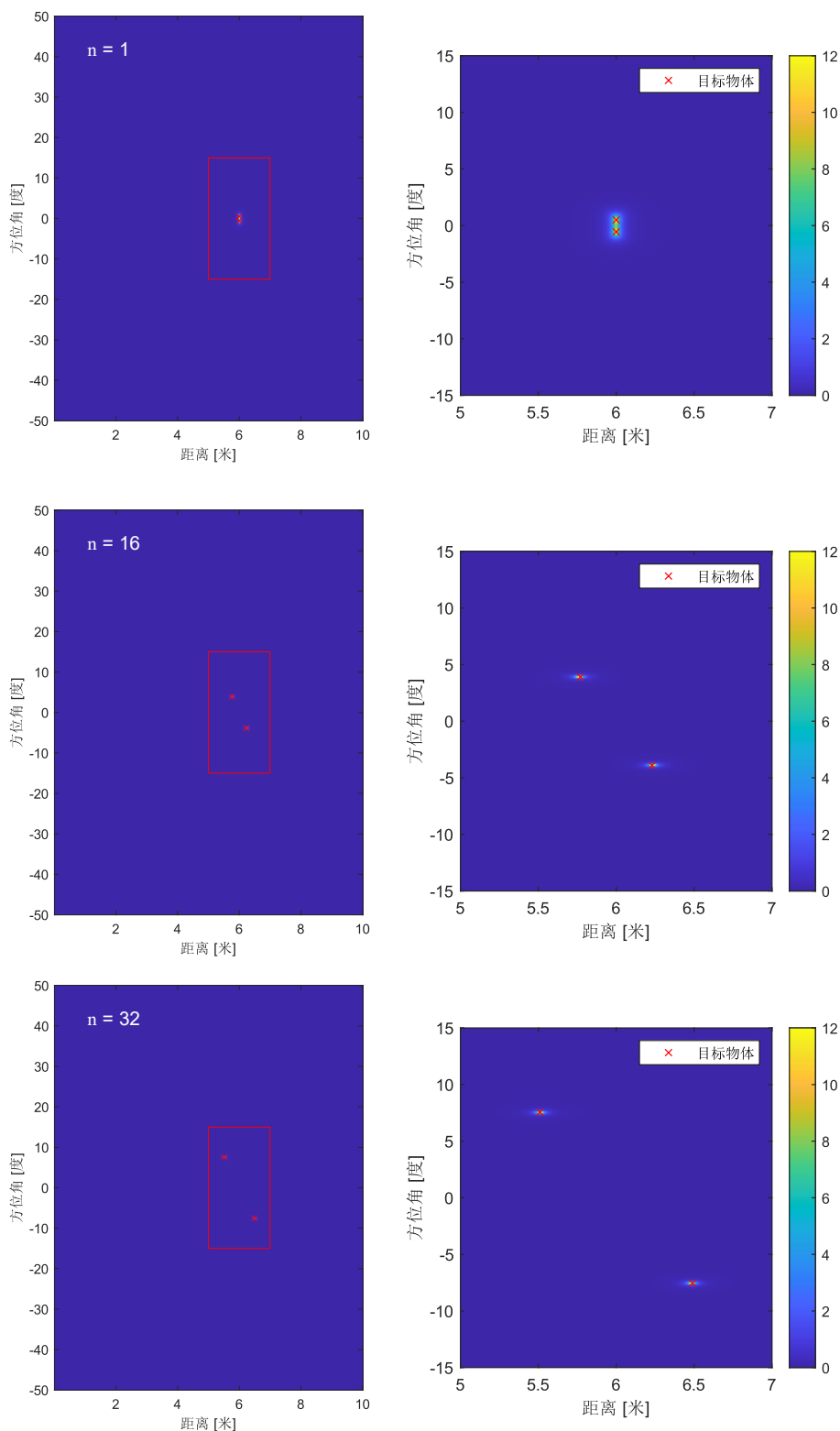


图 25 一帧信号数据中不同 chirp 的目标物体定位情况（选取第 1、16、32 个 chirp）

综合 32 个 chirps 的定位结果，可以画出如图 26 所示的定位散点图。图中散点的颜色表示时间先后顺序（由浅色到深色）。

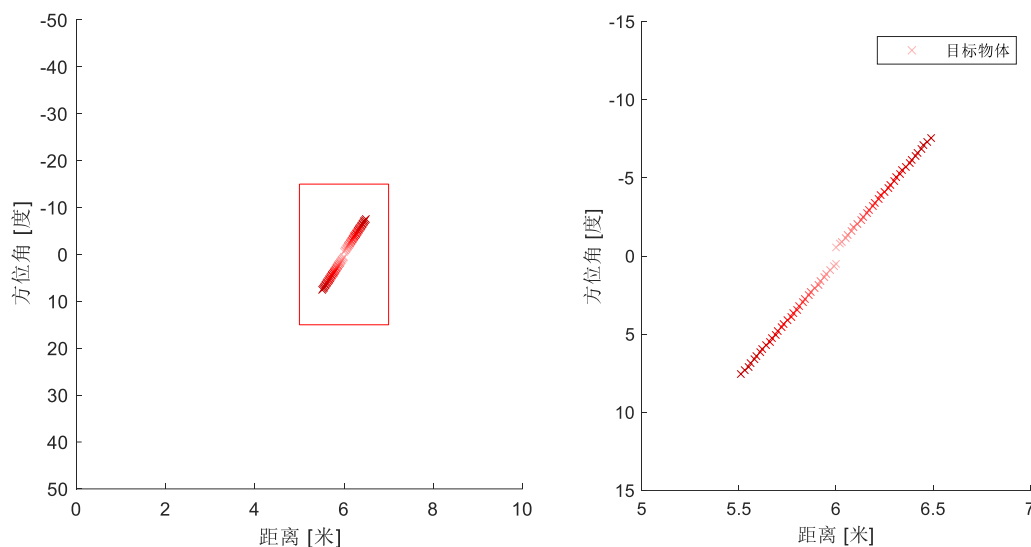


图 26 一帧信号数据的目标物体定位情况（每个 chirp 独立处理）

从目标物体的散点图中，可以比较明显的判断出两个物体起初处于邻近位置，之后逐渐分离，一个向阵列天线靠近，一个向阵列天线远离。不过需要注意的是，使用以上各个 chirps 独立处理的方法，虽然可以探测出每个时刻所有物体的位置，但是其本身并没有将多个时刻关于同一目标物体的多个位置关联起来以确定目标物体的运动轨迹；此外，以上的方法没有充分利用物体运动连续性的信息来辅助 FMCW- MUSIC 算法的搜索。

5.23 考虑物体运动连续性情况下的 FMCW-MUSIC 定位

我们使用 5.1 节中所述的连续跟踪 FMCW- MUSIC 算法，对一帧数据进行处理。在此对搜索过程具体说明如下：

1) 第 1 个 chirp

不存在先验位置信息，对整个二维平面搜索，可以得到目标物体数 $K = 2$ 以及目标物体的位置。

2) 第 1 个 chirp 到第 2 个 chirp

以本题数据中左侧（方位角为负）的目标为例，给出了第一个 chirp 到第二个 chirp 的在有运动连续性辅助情况下的跟踪过程，如图 27 所示。

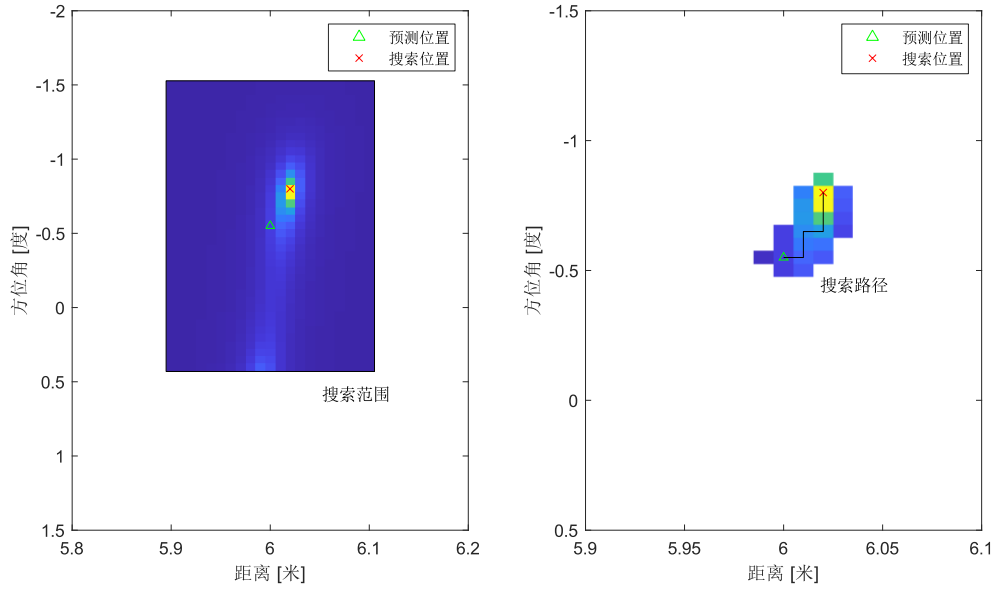


图 27 运动连续性约束下目标物体跟踪过程（左：局部二维搜索，右：梯度下降法搜索）

图中，绿色的三角为目标预测位置，在此即上一时刻的探测到的目标位置。而红色的叉表示本时刻目标物体所在的位置。

左图给出了局部二维搜索的情况。可以看到，FMCW-MUSIC 的搜索范围被限定在了 0.2 米左右的方框中；相比于在整个探测范围内去搜索峰值，运动连续性的约束大大减少了需要计算的采样点数量。

右图给出了梯度下降搜索的情况。在此，我们使用最为朴素的梯度下降法，即探测每一个当前样点附近的四个样点，选取响应值最大的作为前进方向。可以看到，由于 FMCW-MUSIC 的空间谱较为平滑，而位置连续性提供了较为可靠的先验位置；在此条件下，梯度下降法可以非常快地收敛到目标位置。其复杂度相比于局部二维搜索又大大地降低了。

3) 第 n 个 chirp 到第 $n+1$ 个 chirp

与 2) 类似，不再展开说明。

5.24 目标物体连续跟踪定位结果

在此，直接使用上述的基于梯度下降法的连续目标跟踪定位方法，对问题三的数据进行处理。得到的物体跟踪结果如图 28 所示。

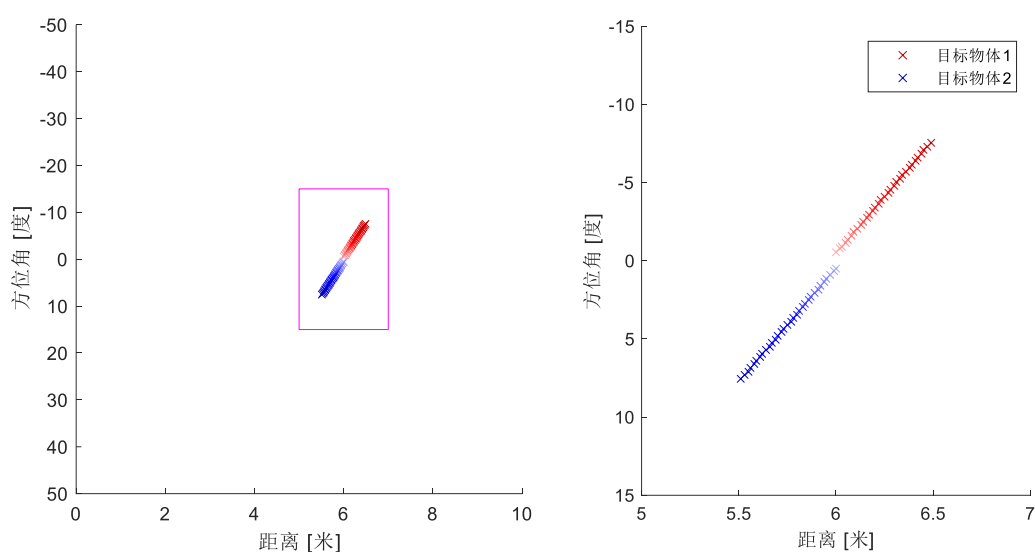


图 28 一帧信号数据的目标物体定位情况（使用运动连续性进行跟踪）

可以看到，目标物体定位的效果和前文的图 26 是完全一致的，但是过程的耗时降低了非常多，这一点将会在下一节性能分析评估中加以论述。此外，由于是在前后时刻之间进行跟踪，因此可以自然地获取不同时刻物体之间的关联——反映在图上，我们可以直接获得两个目标物体各自的运动轨迹，而非指向不明的散点。

5.25 性能分析评估

为了评估所实现的目标跟踪定位算法的性能，在此采用不同的 FMCW-MUSIC 处理策略对本问题的信号数据进行处理。统计不同方案的算法耗时以及 FMCW-MUSIC 二维搜索时的采样点计算次数，如表 6 所示（表中的数据不考虑第 1 个 chirp 时的初始探测）。

表 6 不同方案算法的耗时及采样点计算次数

序号	方案	平均采样点计算次数 (每个 chirp)	平均搜索耗时 (每个 chirp)
1	不使用任何策略	2001000	108.2 秒
2	使用多层级分辨率加速搜索	20090	1.14 秒
3	运动连续性+局部搜索	1694	0.13 秒
4	运动连续性+梯度下降	39	0.027 秒

可以发现，在使用运动连续性和前后连续跟踪这一条件之后，FMCW-MUSIC 搜索所需要计算的采样点数和耗时都可以极大的降低。尤其是采用了梯度下降法之后，仅仅需要计算几十个采样点就可以到达最优值，可以期望满足在线、低复杂度的算法要求。

6. 问题四模型建立与求解

6.1 模型建立

本题的关键在于，在天线阵列信号数据存在结构性噪声的情况下，尽可能地保证系统探测并精确定位目标物体的能力。考虑到本问题的应用场景下，无法使用已知的信号源来对阵列天线自身进行较准，而天线的几何结构发生的变化是完全未知的；因此，需要考虑充分利用相对可靠的信息对天线阵列的测量进行约束。

天线阵列的定位功能依赖其本身的几何设计（具体来说，如天线间距，是否呈直线或平面等）。但是值得注意的是，在本问题中，由于不是通常的 MIMO 系统，单天线本身的测量性能不会受到天线阵列几何变化的影响。也就是说，使用基于 FMCW 的单天线进行测距，是一个较为可靠的信息；无论阵列天线的几何特性发生什么变化，单天线的中频信号频谱始终可以提供一个精度约为 0.0593 米的测距信息。如 3.26 节所述，这个测距信息可以给 FMCW-MUSIC 算法带来有效的帮助，减小搜索空间，提高算法的效率。而在阵列天线本身性能受到影响的情况下，这一测距信息可以期望起到更加重要的作用，对 FMCW-MUSIC 算法进行约束，尽可能压制结构性噪声带来的影响。

6.2 模型求解

6.2.1 数据预处理

首先对阵列天线的数据进行快速傅里叶变换，计算出频谱如图 29 所示。

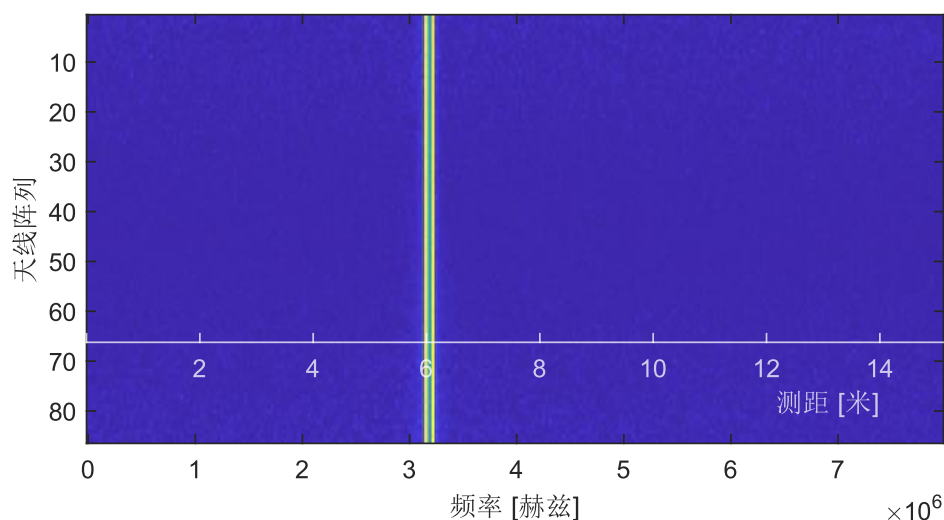


图 29 问题 4 中阵列天线中频信号的频谱（及对应的测距信息）

从图中可以看到，频谱上存在两个明显的峰值，分别对应于测距值为 5.99 米和 6.11 米处。

以上的过程中，中频信号频谱的计算没有使用到阵列天线的特性，而是对各个天线分别进行处理，因此其得到的测距信息不会受到阵列天线几何结构误差带来的影响。其测距分辨率虽然较低，但是可以被看做是相对准确可靠的信息，可以在之后用于辅助 FMCW-MUSIC 的定位。

6.22 在结构误差下进行 FMCW-MUSIC 目标定位

接下来考虑在存在未知结构性误差的情况下使用 FMCW-MUSIC 算法进行目标物体的确定。为了对这种特殊场景（即存在天线自身定位误差）进行分析，我们对 K 值的估计过程进行了具体的查看。搜索步长和空间-时间域窗口尺寸的设置沿用问题 2 和问题 3 中的设置。

首先选取 $K = 1$ ，得到 FMCW-MUSIC 的空间谱如图 30 所示。

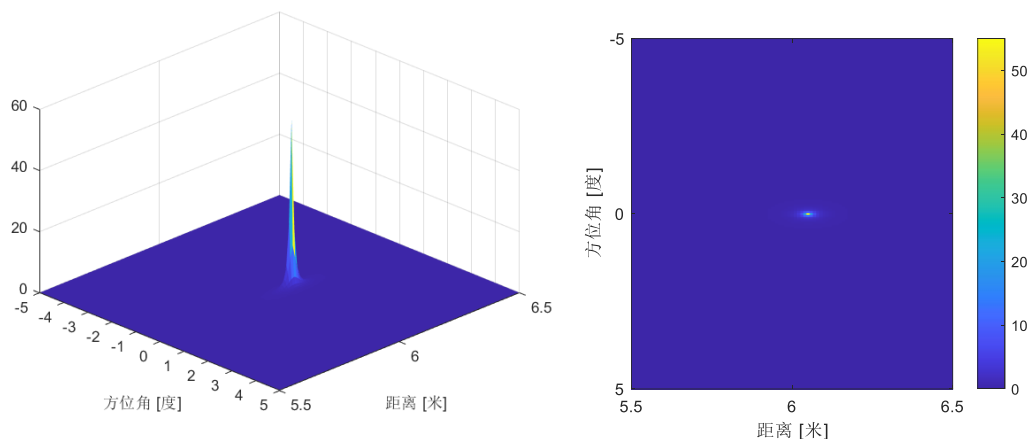


图 30 问题 4 中选取目标物体数 $K = 1$ 时的距离-方位角空间谱

可以看到，空间谱上出现了一个非常强的峰值，其具体的坐标为 $(0^\circ, 6.05 \text{ m})$ 。显然，这和 3.1 节中频谱峰值测距的数值以及两个峰值的情况并不符合，因此排除这种解的情况。

在此之后，选取 $K = 2$ ，得到 FMCW-MUSIC 的距离-方位角二维空间谱如图 31 所示。

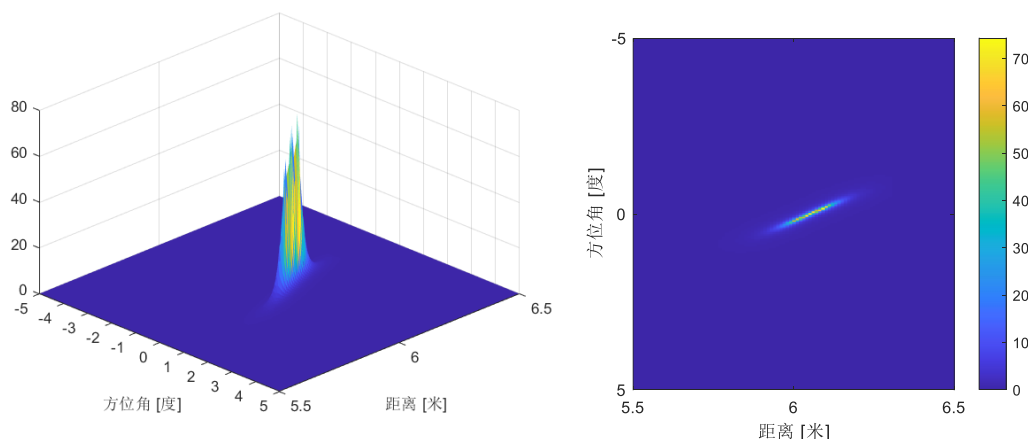


图 31 问题 4 中选取目标物体数 $K = 2$ 时的距离-方位角空间谱

从上图中可以发现一个有趣的现象：距离-方位角的二维空间谱上，出现了邻近的多个突刺点，不再具有局部平滑的特点。可以推测，这并不是因为环境噪声所引起的，更大的可能是由于天线自身的结构发生了一些几何变化所带来的结构性噪声，使得 FMCW-MUSIC 算法在高分辨率的情况下失效了。

6.23 单天线测距信息约束的 FMCW-MUSIC 目标定位

如 6.1 节所述，可以考虑使用单天线 FMCW 测距的结果，对 FMCW-MUSIC 进行一定的约束，从而使其在目前情况下的定位表现最大化。在此利用 FMCW 的测距信息（5.99 m 和 6.11 m，见 6.21 节），对 FMCW-MUSIC 的搜索施加一个比较严格的约束（测距值的 ± 3 cm），限定其搜索范围，重新进行搜索，得到距离-方位角的二维空间谱如图 32 所示。

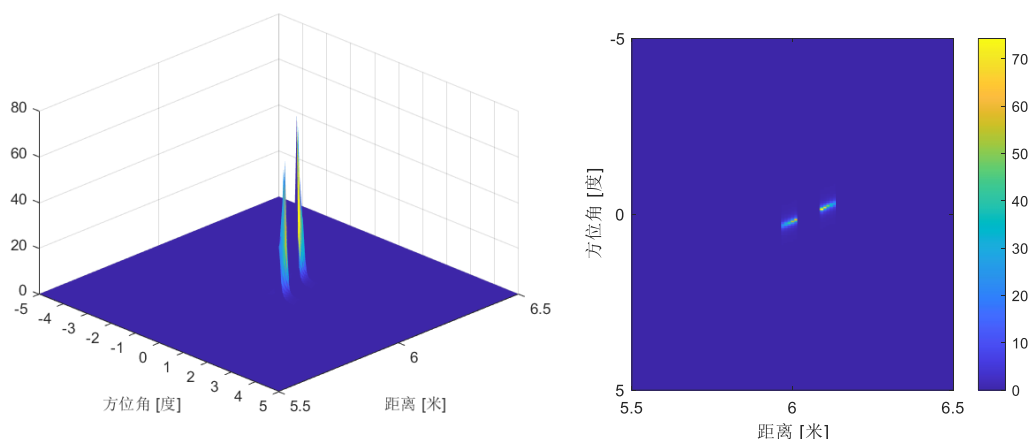


图 32 使用 FMCW 测距信息约束搜索范围后的距离-方位角空间谱

从结果中可以看到，尽管空间谱上依然存在邻近的突刺，但是大致上可以判别出两个波峰的情况，基本上可以判定为 2 个目标物体产生的信号。考虑到当前的天线阵列存在未知的结构性的噪声，很难实现理想情况下的测距、测角精度。而目前辨认出的波峰，基本上也可以达到一个测距厘米级、测角 0.5° 以下的精度，处于可以接受的范围。

类似的，选取 $K = 3, 4$ ，进行波峰的寻找，可以得到在现有分辨率下无法识别更多物体的结论，在此不加赘述。因此，可以确定最优的目标物体数目估值 $K = 2$ 。

6.24 结构误差下目标物体定位结果

区别于问题 1 中通过寻找局部极值点的算法，目前结构性噪声使得空间谱不再平滑；因此，我们在此提高局部极值点的要求，要求其为局部 5×5 的二维空间内的最大值点。按照如此的策略在 FMCW-MUSIC 空间谱上进行目标物体的搜索，可以得到定位结果，如图 33、图 34 所示。

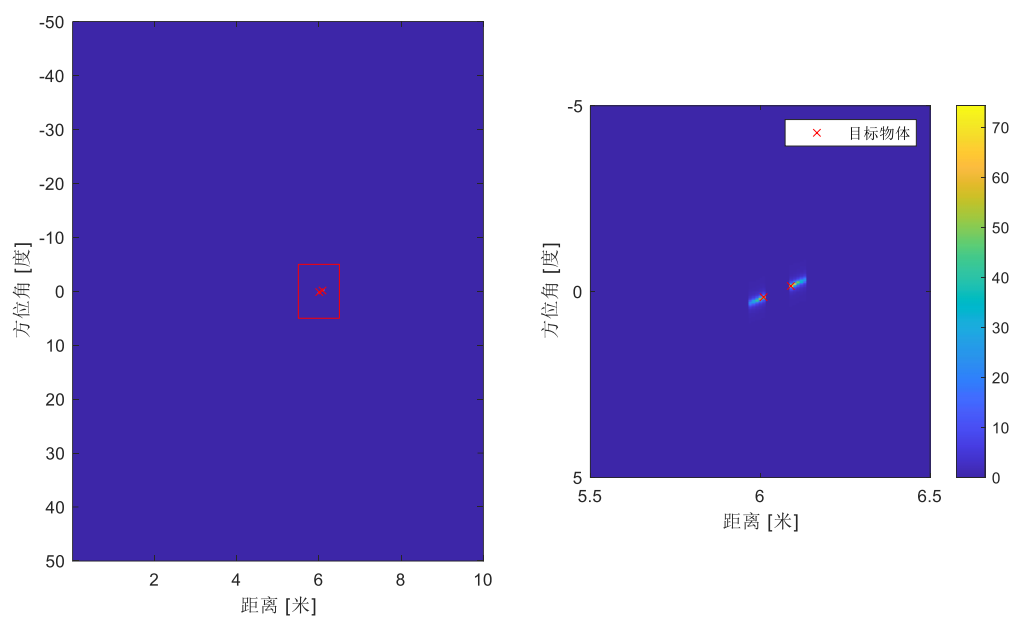


图 33 问题 4 目标物体定位结果（距离-方位角平面）

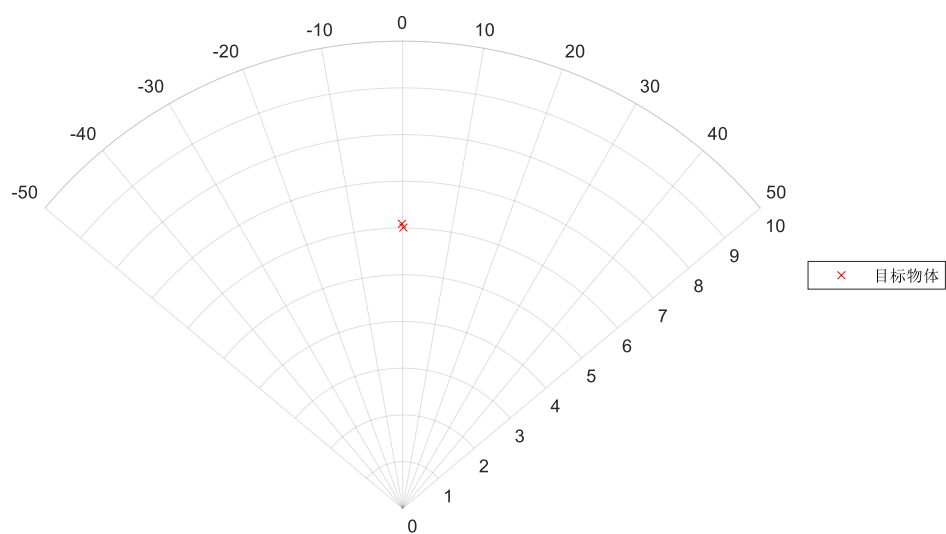


图 34 问题 4 目标物体定位结果（极坐标系）

探测得到的两个物体的坐标分别为 $(0.15^\circ, 6.01\text{m})$ 以及 $(-0.15^\circ, 6.09\text{m})$ 。定位精度考虑退化为理想情况下的三倍，即径向精度和切向精度均在 0.03 米左右。

参考文献

- [1] 姬娜娜. 微小型毫米波测距系统设计[D]. 中北大学, 2022.
DOI:10.27470/d.cnki.ghbgc.2022.000126.
- [2] R. O. Schmidt. Multiple emitter location and signal parameter estimation[J]. *Antennas Propagation, IEEE*, 34(3): 276-280, 1986.
- [3] J. F. Claerbout, F. Muir. Robust modeling with erratic data[J]. *Geophysics*, 38(5): 826-844, 1973.
- [4] M. A. Herman, T. Strohmer. High-resolution radar via compressed sensing. *IEEE transactions on signal processing*, 57(6): 2275-2284, 2009.
- [5] 苏延川, 崔炳喆, 郭玉霞. MUSIC 算法高分辨测角性能影响因素分析[J]. *电子质量*, 2022(000-004), 2022.
- [6] 康亚芳, 王静, 张清泉, 行小帅. 基于 2D-MUSIC 算法的 DOA 估计[J]. *海南师范大学学报(自然科学版)*, 27(03):266-270, 2014.
- [7] 朱泽锋, 陈庚. 基于 2D-MUSIC 的 MIMO 雷达目标估计[J]. *舰船电子工程*, 42(02):55-58, 2022.
- [8] F. Belfiori, W. V. Rossum, P. Hoogeboom. 2D-MUSIC technique applied to a coherent FMCW MIMO radar[C]. *IET International Conference on Radar Systems*. IET, 2013.
- [9] G. O. Manokhin, Z. T. Erdyneev, A. A. Geltser, et al. MUSIC-based algorithm for range-azimuth FMCW radar data processing without estimating number of targets[C]. *Microwave Symposium*. IEEE, 2016.
- [10] N. Jorge, J. W. Stephen. *Numerical Optimization*[M]. New York. Springer, 2006.

附录

MATLAB 源程序:

1. 问题一 (1) **code_q1.m**: 使用 FMCW-MUSIC 算法求解目标物体位置

```
close all;
```

```
clear all;
```

```
Ts = 1.25e-7; Fs = 1/Ts; T = 3.2e-5;  
Nf = 32; L = 0.0815; gamma = 78.986e12;  
Na = 86; d = L / (Na-1); f0 = 78.8e9;  
c = 2.99792458e8; N = 256;  
K = 2;
```

```
% 读取信号数据
```

```
load('data_q1.mat');
```

```
% 建立窗口
```

```
l1 = 10; p1 = Na - l1 + 1;
```

```
l2 = 10; p2 = N - l2 + 1;
```

```
X = zeros(l1*l2, p1*p2);
```

```
count = 0;
```

```
for i = 1 : Na - l1 + 1
```

```
    for j = 1 : N - l2 + 1
```

```
        count = count + 1;
```

```
        m_sub = Z(i:i+l1-1, j:j+l2-1);
```

```
        x_sub = reshape(m_sub, [l1*l2, 1]);
```

```
        X(:,count) = x_sub;
```

```
    end
```

```
end
```

```
% 计算协方差矩阵
```

```

J = fliplr(eye(l1*l2,l1*l2));
XXH = X*X';
C = 1/(2*p1*p2) * (XXH + J/XXH*det(XXH)*J);
% C = 1/(p1*p2) * (XXH );

% 分解子空间
[U,S] = svd(C);
W = U(:,K+1:l1*l2);
WWH = W*W';

% 选取搜索步长
nR = 1000; resoR = 0.01;
ntheta = 2001; resotheta=0.05;

t0 = tic;
count = 0;
SS = zeros(nR,ntheta);
for iRk = 1:1:nR
    Rk = iRk * resoR;
    for itheta = 1:1:ntheta
        theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
        a_sub = zeros(l1,l2);
        for i = 1:l1
            tau = 2/c * (Rk + (i-1)*d* sin(theta));
            for j = 1:l2
                a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
            end
        end
    end
end

```

```

a_sub = reshape(a_sub,[11*12,1]);
SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
count = count+1;
end
end
toc(t0)

% 寻找局部极值点
SSabs= abs(SS);
pt = [];
SSabsmax= max(max(SSabs))
for i = 2:size(SSabs,1)-1
    for j = 2:size(SSabs,2)-1
        if SSabs(i,j)>SSabsmax/2 && SSabs(i,j)>SSabs(i-1,j)...
            && SSabs(i,j)>SSabs(i+1,j)...
            && SSabs(i,j)>SSabs(i,j-1)...
            && SSabs(i,j)>SSabs(i,j+1)
                pt=[pt;[i,j]];
            end
        end
    end
end

% 绘制定位结果
figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
yticks([-50,-40,-30,-20,-10,0,10,20,30,40,50]);
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
set(gca,'YDir','reverse');

```



```

ylabel('方位角 [度]');
xlabel('距离 [米]');
plot([6.5,6.5,7.5,7.5,6.5],[-5,5,5,-5,-5],'r');
set(gcf,'Position',[100,100,400,500]);

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
(pt(:,2) -(ntheta+1)/2)*resotheta ,pt(:,1)*resoR
set(gca,'YDir','reverse');
ylim([-5,5]);
xlim([6.5,7.5]);
caxis([400,480]);
set(gcf,'Position',[100,100,350,350]);
legend('目标物体');
ylabel('方位角 [度]');
xlabel('距离 [米]');

for i = 1:size(pt,1)
    fprintf('目标物体%d 坐标: %f °, %f m\n',i,(pt(i,2) -(ntheta+1)/2)*resotheta,pt(i,1)*resoR);
end

```

2. 问题一（2） **code_q1_acc.m**: 使用改进的高效率 **FMCW-MUSIC** 算法求解目标物体位置

```

close all;
clear all;

Ts = 1.25e-7; Fs = 1/Ts; T = 3.2e-5;
Nf = 32; L = 0.0815; gamma = 78.986e12;

```

```

Na = 86; d = L / (Na-1); f0 = 78.8e9;
c = 2.99792458e8; N = 256;
K = 2;

% 读取信号数据
load('data_q1.mat');

% 进行快速傅里叶变换实现测距
f_abs = abs(fft(Z(1,:)));
[M,f_peaks] = findpeaks(f_abs);
dist_peaks = (f_peaks-1)/N/Ts*c/2/gamma;

% 建立窗口
l1 = 10; p1 = Na - l1 + 1;
l2 = 10; p2 = N - l2 + 1;
X = zeros(l1*l2, p1*p2);

count = 0;
for i = 1 : Na - l1 + 1
    for j = 1 : N - l2 + 1
        count = count + 1;
        m_sub = Z(i:i+l1-1, j:j+l2-1);
        x_sub = reshape(m_sub,[l1*l2,1]);
        X(:,count) = x_sub;
    end
end

% 计算协方差矩阵
J = fliplr(eye(l1*l2,l1*l2));
XXH = X*X';
C = 1/(2*p1*p2) * (XXH + J/XXH*det(XXH)*J);
% C = 1/(p1*p2) * (XXH );

% 分解子空间
[U,S] = svd(C);
W = U(:,K+1:l1*l2);
WWH = W*W';

% 选取搜索步长
nR = 1000; resoR = 0.01;
ntheta = 2001; resotheta=0.05;

t0 = tic;

```

```

SS = zeros(nR,ntheta);

% 选取多层次搜索的步长系数及阈值
step_R_scale = [10,2,1];
step_theta_scale = [20,4,1];
step_thr_scale=[0.04,2,1];

% 开始进行多层次分辨率的搜索（搜索层级 1）
count = 0;
for iRk = 1:step_R_scale(1):nR
    Rk = iRk * resoR;
    skip_Rk = 1;

    % 使用频谱峰值的距离约束距离搜索范围
    for iii = 1:size(dist_peaks,1)
        if abs(Rk - dist_peaks(iii,1))<0.1
            skip_Rk =0; break
        end
    end
    if skip_Rk==1
        continue
    end

    for itheta = 1:step_theta_scale(1):ntheta
        theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
        a_sub = zeros(l1,l2);
        for i = 1:l1
            tau = 2/c * (Rk + (i-1)*d* sin(theta));
            for j = 1:l2
                a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
            end
        end
        a_sub = reshape(a_sub,[l1*l2,1]);
        SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
        count = count +1;

    % 进入搜索层级 2
    if abs(SS(iRk,itheta))>step_thr_scale(1)
        Rk_bound = step_R_scale(1)/2/step_R_scale(2);
        theta_bound = step_theta_scale(1)/2/step_theta_scale(2);
        for iiRk = (floor(-Rk_bound):ceil(Rk_bound))*step_R_scale(2)
            for iitheta = (floor(-theta_bound):ceil(theta_bound))*step_theta_scale(2)
                Rk = (iRk+iiRk) * resoR;
                theta = (itheta + iitheta -(ntheta+1)/2)*resotheta/180.0*pi;
            end
        end
    end
end

```



```

for i = 2:size(SSabs,1)-1
    for j = 2:size(SSabs,2)-1
        if SSabs(i,j)>SSabsmax/2 && SSabs(i,j)>SSabs(i-1,j)...
            && SSabs(i,j)>SSabs(i+1,j)...
            && SSabs(i,j)>SSabs(i,j-1)...
            && SSabs(i,j)>SSabs(i,j+1)
                pt=[pt;[i,j]];
            end
        end
    end
end

% 绘制定位结果
figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
yticks([-50,-40,-30,-20,-10,0,10,20,30,40,50]);
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
set(gca,'YDir','reverse');
ylabel('方位角 [度]');
xlabel('距离 [米]');
plot([6.5,6.5,7.5,7.5,6.5],[-5,5,5,-5,-5], 'r');
set(gcf,'Position',[100,100,400,500]);

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
(pt(:,2) -(ntheta+1)/2)*resotheta ,pt(:,1)*resoR
set(gca,'YDir','reverse');
ylim([-5,5]);
xlim([6.5,7.5]);
caxis([400,480]);
set(gcf,'Position',[100,100,350,350]);
legend('目标物体');
ylabel('方位角 [度]');
xlabel('距离 [米]');

for i = 1:size(pt,1)
    fprintf('目标物体%d 坐标:  %f °,  %f m\n',i,(pt(i,2) -(ntheta+1)/2)*resotheta,pt(i,1)*resoR);
end

```

3. 问题二 code_q2_acc.m: 使用改进的 FMCW-MUSIC 算法求解噪声环境下目标物体位置

```
close all;
clear all;

Ts = 1.25e-7; Fs = 1/Ts; T = 3.2e-5;
Nf = 32; L = 0.0815; gamma = 78.986e12;
Na = 86; d = L / (Na-1); f0 = 78.8e9;
c = 2.99792458e8; N = 256;
K = 2;

% 读取信号数据
load('data_q2.mat'); Z = Z_noisy;

% 进行快速傅里叶变换实现测距
f_abs = abs(fft(Z(1,:)));
[M,f_peaks] = findpeaks(f_abs,'MinPeakProminence',100);
dist_peaks = (f_peaks-1)/N/Ts*c/2/gamma;

% 建立窗口
l1 = 20; p1 = Na - l1 + 1;
l2 = 10; p2 = N - l2 + 1;
X = zeros(l1*l2, p1*p2);

count = 0;
for i = 1 : Na - l1 + 1
    for j = 1 : N - l2 + 1
        count = count + 1;
        m_sub = Z(i:i+l1-1, j:j+l2-1);
        x_sub = reshape(m_sub,[l1*l2,1]);
        X(:,count) = x_sub;
    end
end

% 计算协方差矩阵
J = fliplr(eye(l1*l2,l1*l2));
XXH = X*X';
% C = 1/(2*p1*p2) * (XXH + J/XXH*det(XXH)*J);
C = 1/(p1*p2) * (XXH );
```

```

% 分解子空间
[U,S] = svd(C);
W = U(:,K+1:11*12);
WWH = W*W';

% 选取搜索步长
nR = 1000; resoR = 0.01;
ntheta = 2001; resotheta=0.05;

t0 = tic;
SS = zeros(nR,ntheta);

% 选取多层次搜索的步长系数及阈值
step_R_scale = [10,2,1];
step_theta_scale = [20,4,1];
step_thr_scale=[0.02*10*10/11/12,1*10*10/11/12,1];

% 开始进行多层次分辨率的搜索（搜索层级 1）
count = 0;
for iRk = 1:step_R_scale(1):nR
    Rk = iRk * resoR;
    skip_Rk = 1;

    % 使用频谱峰值的距离约束距离搜索范围
    for iii = 1:size(dist_peaks,1)
        if abs(Rk - dist_peaks(iii,1))<0.1
            skip_Rk =0; break
        end
    end
    if skip_Rk==1
        continue
    end

    for itheta = 1:step_theta_scale(1):ntheta
        theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
        a_sub = zeros(11,12);
        for i = 1:11
            tau = 2/c * (Rk + (i-1)*d* sin(theta));
            for j = 1:12
                a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
            end
        end
        a_sub = reshape(a_sub,[11*12,1]);
        SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
    end
end

```

```
count = count +1;
```

```
% 进入搜索层级 2
```

```
if abs(SS(iRk,itheta))>step_thr_scale(1)
```

```
    Rk_bound = step_R_scale(1)/2/step_R_scale(2);
```

```
    theta_bound = step_theta_scale(1)/2/step_theta_scale(2);
```

```
    for iiRk = (floor(-Rk_bound):ceil(Rk_bound))*step_R_scale(2)
```

```
        for iitheta = (floor(-theta_bound):ceil(theta_bound))*step_theta_scale(2)
```

```
            Rk = (iRk+iiRk) * resoR;
```

```
            theta = (itheta + iitheta -(ntheta+1)/2)*resotheta/180.0*pi;
```

```
            a_sub = zeros(11,12);
```

```
            for i = 1:11
```

```
                tau = 2/c * (Rk + (i-1)*d* sin(theta));
```

```
                for j = 1:12
```

```
                    a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
```

```
                end
```

```
            end
```

```
            a_sub = reshape(a_sub,[11*12,1]);
```

```
            SS(iRk+iiRk,itheta+iitheta)=1/(a_sub' * WWH *a_sub);
```

```
            count = count +1;
```

```
% 进入搜索层级 3
```

```
if abs(SS(iRk+iiRk,itheta+iitheta))>step_thr_scale(2)
```

```
    Rk_bound = step_R_scale(2)/2/step_R_scale(3);
```

```
    theta_bound = step_theta_scale(2)/2/step_theta_scale(3);
```

```
    for iiiRk = (floor(-Rk_bound):ceil(Rk_bound))*step_R_scale(3)
```

```
        for iiiieta = (floor(-theta_bound):ceil(theta_bound))*step_theta_scale(3)
```

```
            Rk = (iRk+iiRk+iiiRk) * resoR;
```

```
            theta = (itheta + iitheta + iiiitheta -(ntheta+1)/2)*resotheta/180.0*pi;
```

```
            a_sub = zeros(11,12);
```

```
            for i = 1:11
```

```
                tau = 2/c * (Rk + (i-1)*d* sin(theta));
```

```
                for j = 1:12
```

```
                    a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
```

```
                end
```

```
            end
```

```
            a_sub = reshape(a_sub,[11*12,1]);
```

```
            SS(iRk+iiRk+iiiRk,itheta + iitheta + iiiitheta)=1/(a_sub' * WWH *a_sub);
```

```
            count = count +1;
```

```
        end
```

```
    end
```

```
end
```

```
end
```



```

        end
    end
end
toc(t0)

% 寻找局部极值点
SSabs= abs(SS);
pt = [];
SSabsmax= max(max(SSabs))
for i = 2:size(SSabs,1)-1
    for j = 2:size(SSabs,2)-1
        if SSabs(i,j)>SSabsmax/2 && SSabs(i,j)>SSabs(i-1,j)...
            && SSabs(i,j)>SSabs(i+1,j)...
            && SSabs(i,j)>SSabs(i,j-1)...
            && SSabs(i,j)>SSabs(i,j+1)
                pt=[pt;[i,j]];
            end
        end
    end
end

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
yticks([-50,-40,-30,-20,-10,0,10,20,30,40,50]);
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta ,'rx');
xlabel('方位角 [度]');
ylabel('距离 [米]');
plot([8.1,8.1,8.3,8.3,8.1],[-3,3,3,-3,-3],'r');
set(gcf,'Position',[100,100,400,500]);
set(gca,'YDir','reverse');

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta ,'rx');
(pt(:,2) -(ntheta+1)/2)*resotheta ,pt(:,1)*resoR
ylim([-3,3]);
xlim([8.1,8.3]);
% set(gcf,'Position',[100,100,300,300]);
set(gcf,'Position',[100,100,350,350]);%colorbar;
legend('目标物体');
ylabel('方位角 [度]');
xlabel('距离 [米]');

```

```
set(gca,'YDir','reverse');
```

```
for i = 1:size(pt,1)
```

```
    fprintf('目标物体%d 坐标: %f°, %f m\n',i,(pt(i,2)-(ntheta+1)/2)*resotheta,pt(i,1)*resoR);
end
```

4. 问题三 code_q3.m: 使用连续跟踪 FMCW-MUSIC 算法求解动态物体位置

```
close all;
```

```
clear all;
```

```
Ts = 1.25e-7; Fs = 1/Ts; T = 3.2e-5;
```

```
Nf = 32; L = 0.0815; gamma = 78.986e12;
```

```
Na = 86; d = L / (Na-1); f0 = 78.8e9;
```

```
c = 2.99792458e8; N = 256;
```

```
K = 2;
```

```
tracking_mode = 0; % 0 为局部搜索, 1 为梯度下降
```

```
load('data_q3.mat');
```

```
all_pt={};
```

```
for ttt = 1:32
```

```
    close all;
```

```
    Z = reshape(Z_time(ttt,,:),[86,256]);
```

```
    % 建立窗口
```

```
    l1 = 20; p1 = Na - l1 + 1;
```

```
    l2 = 10; p2 = N - l2 + 1;
```

```
    X = zeros(l1*l2, p1*p2);
```

```
    count = 0;
```

```
    for i = 1 : Na - l1 + 1
```

```
        for j = 1 : N - l2 + 1
```

```
            count = count + 1;
```

```
            m_sub = Z(i:i+l1-1, j:j+l2-1);
```

```
            x_sub = reshape(m_sub,[l1*l2,1]);
```

```
            X(:,count) = x_sub;
```

```
        end
```

```
    end
```

```
    J = fliplr(eye(l1*l2,l1*l2));
```

```
    XXH = X*X';
```

```
    C = 1/(2*p1*p2) * (XXH + J/XXH*det(XXH)*J);
```

```

% C = 1/(p1*p2) * (XXH );

[U,S] = svd(C);
W = U(:,K+1:l1*l2);
WWH = W*W';

t0 = tic;

nR = 1000; resoR = 0.01;
ntheta = 2001; resotheta=0.05;

SS = zeros(nR,ntheta);
count = 0;

% 第一个 chirp 的处理，不考虑位置先验信息
if ttt == 1
    % 进行快速傅里叶变换实现测距
    f_abs = abs(fft(Z(1,:)));
    [M,f_peaks] = findpeaks(f_abs,'MinPeakProminence',100);
    dist_peaks = (f_peaks-1)/N/Ts*c/2/gamma;
    for iRk = 1:1:nR
        Rk = iRk * resoR;
        % 对于第一个 chirp 来说，可以使用 FFT 得到的测距值来辅助
        skip_Rk = 0;
        % 使用频谱峰值的距离约束距离搜索范围
        for iii = 1:size(dist_peaks,1)
            if abs(Rk - dist_peaks(iii,1))>0.1
                skip_Rk = 1; break
            end
        end
        if skip_Rk==1
            continue
        end
        for itheta = 1:1:ntheta
            theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
            a_sub = zeros(l1,l2);
            for i = 1:l1
                tau = 2/c * (Rk + (i-1)*d* sin(theta));
                for j = 1:l2
                    a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
                end
            end
            a_sub = reshape(a_sub,[l1*l2,1]);

```

```

        SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
        count = count+1;
    end
end
SSabs= abs(SS);
pt = [];
SSabsmax= max(max(SSabs))
for i = 2:size(SSabs,1)-1
    for j = 2:size(SSabs,2)-1
        if SSabs(i,j)>SSabsmax/2 && SSabs(i,j)>SSabs(i-1,j)...
            && SSabs(i,j)>SSabs(i+1,j)...
            && SSabs(i,j)>SSabs(i,j-1)...
            && SSabs(i,j)>SSabs(i,j+1)
                pt=[pt;[i,j]];
            end
        end
    end
end
all_pt{ttt} = pt;
end

```

% 第 2~Nf 个 chirp 的处理，考虑之前获得的位置先验信息

```

if ttt > 1 && tracking_mode ==0 %使用局部搜索的方法进行连续跟踪
    pt = all_pt{ttt-1};
    pt_new = pt;
    t1= tic;count = 0;
    % 限定搜索范围，局部搜索
    for ip = 1:size(pt,1)
        for iRk = floor(pt(ip,1) - 0.1/resoR):1:ceil(pt(ip,1) + 0.1/resoR)
            Rk = iRk * resoR;
            for itheta = floor(pt(ip,2) - 0.1/Rk*180/pi/resotheta):1:ceil(pt(ip,2) +
0.1/Rk*180/pi/resotheta)
                theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
                a_sub = zeros(l1,l2);
                for i = 1:l1
                    tau = 2/c * (Rk + (i-1)*d* sin(theta));
                    for j = 1:l2
                        a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
                    end
                end
                a_sub = reshape(a_sub,[l1*l2,1]);
                SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
                count = count+1;
            end
        end
    end
end

```

```

SSabs =abs(SS);
max_value = 0;
for iRk = floor(pt(ip,1) - 0.1/resoR):1:ceil(pt(ip,1) + 0.1/resoR)
    for itheta = floor(pt(ip,2) - 0.1/Rk*180/pi/resotheta):1:ceil(pt(ip,2) +
0.1/Rk*180/pi/resotheta)
        if SSabs(iRk,itheta) > max_value
            max_value = SSabs(iRk,itheta);
            pt_new(ip,1)= iRk;
            pt_new(ip,2)= itheta;
        end
    end
end
end
toc(t1),count
all_pt{tnt}= pt_new;
end

```

if tnt > 1 && tracking_mode ==1 %使用梯度下降的方法进行连续跟踪

```

pt = all_pt{tnt-1};
pt_new = pt;
% 寻找下降梯度最大的方向
try_direction = [0,0;0,1;0,-1;-1,0;1,0];
t1 = tic;
count = 0;
for ip = 1:size(pt,1)
    search_traj = [];
    search_traj =[search_traj;pt_new(ip,:)];
    while 1
        max_direction_zzz=0;
        max_value =0;
        for zzz = 1:5
            iRk = pt_new(ip,1) + try_direction(zzz,1);
            itheta = pt_new(ip,2) + try_direction(zzz,2);
            if abs(SS(iRk,itheta))<0.0001 % 如果该点还没有被计算过
                Rk = iRk * resoR;
                theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
                a_sub = zeros(11,12);
                for i = 1:11
                    tau = 2/c * (Rk + (i-1)*d* sin(theta));
                    for j = 1:12
                        a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
                    end
                end
            end
        end
    end
end

```

```

        a_sub = reshape(a_sub,[11*12,1]);
        SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
        count = count +1;
    end
    if abs(SS(iRk,itheta)) > max_value
        max_value = SS(iRk,itheta);
        max_direction_zzz = zzz;
    end
end
if max_direction_zzz == 1
    break
else
    pt_new(ip,1) = pt_new(ip,1) + try_direction(max_direction_zzz,1);
    pt_new(ip,2) = pt_new(ip,2) + try_direction(max_direction_zzz,2);
    pt_new(ip,:)
    abs(SS(pt_new(ip,1),pt_new(ip,2)))
    search_traj=[search_traj;pt_new(ip,:)];
end
end
end
toc(t1),count
all_pt{ttt}= pt_new;
end
end

% 绘制目标物体运动轨迹
figure;
mycolormap_r=interp1([1,32,64],[255,255,192],1:64,'linear','extrap');
mycolormap_g=interp1([1,32,64],[180,0,0],1:64,'linear','extrap');
mycolormap_b=interp1([1,32,64],[180,0,0],1:64,'linear','extrap');
mycolor_red=[mycolormap_r',mycolormap_g',mycolormap_b']/255;

mycolormap_b=interp1([1,32,64],[255,255,192],1:64,'linear','extrap');
mycolormap_g=interp1([1,32,64],[180, 0, 0],1:64,'linear','extrap');
mycolormap_r=interp1([1,32,64],[180, 0, 0],1:64,'linear','extrap');
mycolor_blue=[mycolormap_r',mycolormap_g',mycolormap_b']/255;

for i = 1:32
    scatter(all_pt{i}(1,1)*resoR,(all_pt{i}(1,2) -
(ntheta+1)/2)*resotheta ,30,'MarkerEdgeColor',mycolor_red(i*2,:),'Marker','x'); hold on;
    scatter(all_pt{i}(2,1)*resoR,(all_pt{i}(2,2) -
(ntheta+1)/2)*resotheta ,30,'MarkerEdgeColor',mycolor_blue(i*2,:),'Marker','x'); hold on;
end

```

```

plot([5,5,7,7,5],[-15,15,15,-15,-15],'magenta');

ylim([-50,50]);
xlim([0,10]);
set(gcf,'Position',[100,100,400,400]);
ylabel('方位角 [度]');
xlabel('距离 [米]');
set(gca,'YDir','reverse');

figure;
for i = 1:32
    hdl1 = scatter(all_pt{i}(1,1)*resoR,(all_pt{i}(1,2) -
(ntheta+1)/2)*resotheta ,30,'MarkerEdgeColor',mycolor_red(i*2,:),'Marker','x'); hold on;
    hdl2 = scatter(all_pt{i}(2,1)*resoR,(all_pt{i}(2,2) -
(ntheta+1)/2)*resotheta ,30,'MarkerEdgeColor',mycolor_blue(i*2,:),'Marker','x'); hold on;
end
legend([hdl1,hdl2],{'目标物体 1','目标物体 2'})
ylim([-15,15]);
xlim([5.0,7.0]);
set(gcf,'Position',[100,100,350,400]);
ylabel('方位角 [度]');
xlabel('距离 [米]');

set(gca,'YDir','reverse');

```

5. 问题四 **code_q4.m**: 使用改进的 FMCW-MUSIC 算法在结构性噪声下求解目标物体位置

```

close all;
clear all;

Ts = 1.25e-7; Fs = 1/Ts; T = 3.2e-5;
Nf = 32; L = 0.0815; gamma = 78.986e12;
Na = 86; d = L / (Na-1); f0 = 78.8e9;
c = 2.99792458e8; N = 256;
K = 2;

load('data_q4.mat'); Z = Z_antnoisy;

% 进行快速傅里叶变换实现测距
f_abs = abs(fft(Z(1,:)));

```

```

[M,f_peaks] = findpeaks(f_abs,'MinPeakProminence',100);
dist_peaks = (f_peaks-1)/N/Ts*c/2/gamma;

% 建立窗口
l1 = 20; p1 = Na - l1 + 1;
l2 = 10; p2 = N - l2 + 1;
X = zeros(l1*l2, p1*p2);

count = 0;
for i = 1 : Na - l1 + 1
    for j = 1 : N - l2 + 1
        count = count + 1;
        m_sub = Z(i:i+l1-1, j:j+l2-1);
        x_sub = reshape(m_sub,[l1*l2,1]);
        X(:,count) = x_sub;
    end
end
J = fliplr(eye(l1*l2,l1*l2));
XXH = X*X';
% C = 1/(2*p1*p2) * (XXH + J/XXH*det(XXH)*J);
C = 1/(p1*p2) * (XXH);

[U,S] = svd(C);
W = U(:,K+1:l1*l2);
WWH = W*W';

t0 = tic;

nR = 1000; resoR = 0.01;
ntheta = 2001; resotheta=0.05;

SS = zeros(nR,ntheta);
count = 0;

for iRk = 1:1:nR
    Rk = iRk * resoR;
    skip_Rk = 1;
    % 使用频谱峰值的距离约束距离搜索范围
    % 在此为了压制结构性噪声的影响，选取比较小的范围
    for iii = 1:size(dist_peaks,2)
        if abs(Rk - dist_peaks(1,iii))<0.03
            skip_Rk=0; break
        end
    end
end
end

```



```

if skip_Rk==1
    continue
end
for itheta = 1:1:ntheta
    theta = (itheta-(ntheta+1)/2)*resotheta/180.0*pi;
    a_sub = zeros(11,12);
    for i = 1:11
        tau = 2/c * (Rk + (i-1)*d* sin(theta));
        for j = 1:12
            a_sub(i,j) = exp(1i*2*pi*(f0*tau + gamma*tau * (j)*Ts));
        end
    end
    end

    a_sub = reshape(a_sub,[11*12,1]);
    SS(iRk,itheta)=1/(a_sub' * WWH *a_sub);
    count = count+1;
end
end
toc(t0)
SSabs= abs(SS);

% 寻找局部极值点
pt = [];
SSabsmax= max(max(SSabs))
for i = 2:size(SSabs,1)-1
    for j = 2:size(SSabs,2)-1
        if SSabs(i,j)>SSabsmax/2 && SSabs(i,j)>SSabs(i-1,j)...
            && SSabs(i,j)>SSabs(i+1,j)...
            && SSabs(i,j)>SSabs(i,j-1)...
            && SSabs(i,j)>SSabs(i,j+1)...
            && SSabs(i,j)>SSabs(i+1,j+1)...
            && SSabs(i,j)>SSabs(i-1,j-1)...
            && SSabs(i,j)>SSabs(i+1,j-1)...
            && SSabs(i,j)>SSabs(i-1,j+1)
            pt=[pt;i,j];
        end
    end
end
figure;
[XX,YY] = meshgrid((1:nR)*resoR,((1:ntheta)-(ntheta+1)/2)*resotheta);
s = surf(XX,YY,abs(SS));
s.EdgeColor = 'none';
yticks(-50:10:50);
xlabel('距离 [米]');

```

```

ylabel('方位角 [度]');
view(-45,37);
set(gca,'YDir','reverse');

figure;
[XX,YY] = meshgrid((1:nR)*resoR,((1:ntheta)-(ntheta+1)/2)*resotheta);
s = surf(XX,YY,abs(SS'));
s.EdgeColor = 'none';
xlim([5.5,6.5]);
ylim([-5,5]);
yticks(-5:1:5);
xlabel('距离 [米]');
ylabel('方位角 [度]');
view(-45,37);
set(gca,'YDir','reverse');

```

```

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
yticks([-50,-40,-30,-20,-10,0,10,20,30,40,50]);
% scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
set(gca,'YDir','normal');
ylabel('方位角 [度]');
xlabel('距离 [米]');
plot([5.5,5.5,6.5,6.5,5.5],[-5,5,5,-5,-5], 'r');
set(gcf,'Position',[100,100,400,500]);
set(gca,'YDir','reverse');

```

```

figure;
imagesc([1*resoR,nR*resoR],[(1-(ntheta+1)/2)*resotheta,( ntheta - (ntheta+1)/2)*resotheta],...
    SSabs'); hold on;
% scatter(pt(:,1)*resoR,(pt(:,2) -(ntheta+1)/2)*resotheta , 'rx');
set(gca,'YDir','reverse');

```

```

(pt(:,2) -(ntheta+1)/2)*resotheta ,pt(:,1)*resoR
set(gca,'YDir','normal');
ylim([-5,5]);
xlim([5.5,6.5]);
% set(gcf,'Position',[100,100,300,300]);
set(gcf,'Position',[100,100,400,350]);colorbar;
% legend('目标物体');
ylabel('方位角 [度]');
xlabel('距离 [米]');

```

```
set(gca,'YDir','reverse');
```

```
for i = 1:size(pt,1)
```

```
    fprintf('目标物体%d 坐标: %f °, %f m\n',i,(pt(i,2) -(ntheta+1)/2)*resotheta,pt(i,1)*resoR);  
end
```