DOI:10.19651/j.cnki.emt.2313457

基于 NVIDIA GPU 后向投影 FFBP 算法的加速研究

潘 丰 高 伟 罗 俊 刘文冬 周春元 张 慧
(珠海微度芯创科技有限责任公司 珠海 519000)

摘 要：后向投影(BP)算法,在计算成像过程中未采用近似,成像质量高,任何阵列构型成像均适合。近年来在雷达成像技术领域广泛应用。但在毫米波三维全息成像中,计算效率较低,影响了实时成像的实现。在三维极坐标条件下,快速因式分解后向投影(FFBP)算法,利用子孔径划分的方式进行成像,一定程度上解决了实时成像的问题。本文利用四线程 CPU 与 GPU 加速 CUDA 平台实现 FFBP 算法,并对比分析了多点目标成像,结果基本一致,进而验证加速算法的有效性。进一步,通过电磁仿真软件,对分辨力板建模和仿真,模拟真实目标,并进行 GPU 加速成像,计算时间比四线程 CPU 提高 33.97 倍,适用于三维近场实时成像系统,更好的应用于人体安检领域。

关键词：三维极坐标系;FFBP 算法;图像处理器(GPU);子孔径划分
中图分类号：TN95 **文献标识码**：A **国家标准学科分类代码**：510.4030

Research on accelerating the backward projection FFBP algorithm based on NVIDIA GPU

Pan Feng Gao Wei Luo Jun Liu Wendong Zhou Chunyuan Zhang Hui
(Zhuhai Microcreative Tech. Co.,Ltd., Zhuhai 519000, China)

Abstract: The Back Projection (BP) algorithm does not use approximation in the imaging calculation process, and the imaging quality is high. Any array configuration is suitable for imaging. In recent years, it has been widely used in the field of radar imaging technology. However, in millimeter wave three-dimension holographic imaging, the computational efficiency is low, which affects the implementation of real-time imaging. Under the conditions of three-dimensional polar coordinates, the Fast Factorization Backward Projection (FFBP) algorithm uses sub-aperture division for imaging, which solves the real-time imaging problem to a certain extent. This article implements the FFBP algorithm based on the four threaded CPU and GPU accelerated CUDA platform, and compares and analyzes the imaging of multi-point targets. The results are basically consistent, thereby verifying the effectiveness of the accelerated algorithm. Furthermore, through electromagnetic simulation software, the resolution board is modeled and simulated to simulate real targets, and GPU accelerated imaging is performed. The calculation time is 33.97 times faster than the four threaded CPU, making it suitable for 3D near-field real-time imaging systems and better applied in the field of human security inspection.

Keywords: 3D polar coordinate system;FFBP algorithm;graphics processing unit(GPU);sub aperture division

0 引 言

近年来,随着雷达技术的日新月异,合成孔径雷达(synthetic aperture radar, SAR)技术领域得到迅速发展,高速发展的图像处理器(graphics processing unit)为高效 SAR 成像算法提供新型运算平台。GPU 高性能计算在雷达成像应用领域越来越广泛。

后向投影(backword projection, BP)时域算法源于计算机层析成像技术,算法无需任何近似,是一种精确的时域成像算法。后向投影的时域算法,需循环遍历所有收发阵元坐标,累加整个感兴趣位置的像素空间,计算量较大,时间复杂度较高,在快速实时成像领域应用中受到限制。

基于传统后向投影算法计算复杂度较高的问题,一些学者,优化了后向投影算法,推导了快速后向投影算法。Ulander 等[1]提出因式分解快速后向投影(FFBP)算法,理论推导了三维空间的快速成像,成像计算复杂度降至和频域算法相当的 $O(N^2 \log N)$。潘丰等[2],仿真及实测对比分析了 BP 算法、FBP(factorization backward projection)算

收稿日期:2023-04-24

法、FFBP 算法。杨啸宇等[3],基于 GPU 研究了近场 RMA (range migration algorithm)算法,加速成像计算速度[3]。刘鹏飞等[4-5],研究了极坐标系下的三维因式分解后向投影算法,并对三维 FFBP 算法孔径划分、算法实现及计算效率等问题进行了分析。班阳阳等[6],研究了一种 BP 算法多流异步执行技术,即数据传输模式并行处理的 GPU 优化成像方案,处理速度在一定程度上得到提升。姜晓龙等[7],针对共享存储器的 bank 冲突问题,给出相应解决方案,并设计不同优化方法对 BP 算法进行加速。陈彦民等[8],提出了 GPU 平台实现多雷达自适应主瓣干扰对消算法,提高信噪比同时并快速处理。孟大地等[9],针对 SAR 成像处理算法的结果特征及 CUDA 架构并行处理特点进行深入研究,充分利用 GPU 设备计算资源,提出了一种基于 CUDA 的机载 SAR 实时成像处理方案。苟立婷等[10],利用采集到的回波数据,并优化极坐标格式算法,提出了一种基于图像处理器的圆迹视频 SAR 实时成像算法。刘百玲等[11],提出了一种 CUDA 平台上用 GPU 进行非相参成像的并行化方法,实现成像过程的并行化。景国彬等[12],采用"线程外推"、"归约相加"的方式,提出了一种结合改进的同心圆算法与 GPU 技术的高效 SAR 回波仿真方法。笪敏等[13],研究了改进型距离徙动算法的 GPU 加速实现。姜晓龙等[14],基于 GPU 加速研究了超宽带 SAR 实时成像技术。梁美彦等[15],研究了基于 GPU 加速相位补偿反投影算法。GPU 在提升 SAR 成像速度方面取得了显著效果。

1　FFBP 算法原理

FFBP 算法使用极坐标系下极线波束近似原理来表示其临近区域,用来减少算法计算量。利用两组同心圆弧来表示两个孔径临近位置的回波数据在感兴趣区域的投影,图 1 中扇形区域,在感兴趣距离位置两组回波数据具有相近的投影。回波数据对波束中心线 or 上的投影近似整个波束 or_1r_2 内所有像素单元的投影。这种近似能够产生距离误差,距离 or 愈远的像素单元误差将愈大,距离波束中心线 or 愈近的像素单元误差将愈小。理论层面,若子孔径长度固定时,波束角宽度与近似距离误差成正比,如果子孔径与波束角宽度的乘积固定,则距离误差 ΔR 将控制在如下范围内:

$$|\Delta R| \leqslant \frac{d \cdot \Delta\theta}{4} \tag{1}$$

式中:d 为子孔径长度;$\Delta\theta$ 为波束角宽度。

FFBP 算法将整个合成孔径划分为若干子孔径,每幅子孔径图像被重建在以子孔径中心为原点的局部坐标系。

三维 FFBP 成像,将包含整个成像区域,沿高度向、方位向划分为若干辐射状的三维网络。如图 2 所示,以笛卡尔坐标系原点 O 为原点,以 σ、α_1、α_2 为轴建立三维极坐标系 $O(\sigma,\alpha_1,\alpha_2)$,三维极坐标系 $O(\sigma,\alpha_1,\alpha_2)$ 与笛卡尔坐标系

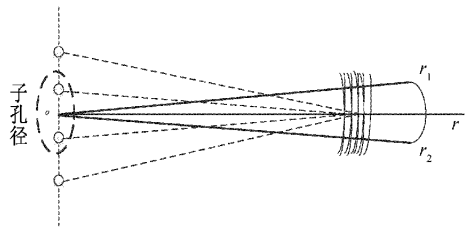

图 1　子孔径波束示意图

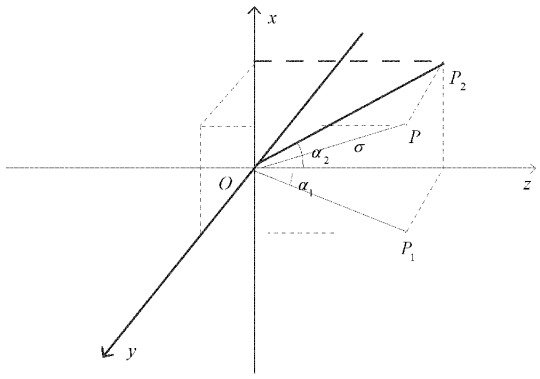

图 2　三维极坐标示意图

$O(x,y,z)$ 满足变换关系如下:

$$(x,y,z) = U(\sigma,\alpha_1,\alpha_2) \tag{2}$$

其中,

$$z = \sigma / \sqrt{\tan\alpha_1^2 + \tan\alpha_2^2 + 1}$$

$$y = z \cdot \tan\alpha_1$$

$$x = z \cdot \tan\alpha_2$$

其中,$-\pi < \alpha_1, \alpha_2 \leqslant \pi$。

反变换关系如下:

$$(\sigma,\alpha_1,\alpha_2) = T^{-1}(x,y,z) \tag{3}$$

其中,$T^{-1}(\cdot)$ 满足下列关系:

$$\sigma = \sqrt{x^2 + y^2 + z^2}$$

$$\alpha_1 = \begin{cases} \arctan(y/z), z > 0 \\ \arctan(y/z) + \pi, z < 0, y \geqslant 0 \\ \arctan(y/z) - \pi, z < 0, y < 0 \\ \pi/2, z = 0, y > 0 \\ 0, z = 0, y = 0 \\ -\pi/2, z = 0, y < 0 \end{cases}$$

$$\alpha_2 = \begin{cases} \arctan(x/z), z > 0 \\ \arctan(x/z) + \pi, z < 0, x \geqslant 0 \\ \arctan(x/z) - \pi, z < 0, x < 0 \\ \pi/2, z = 0, x > 0 \\ 0, z = 0, x = 0 \\ -\pi/2, z = 0, x < 0 \end{cases}$$

在三维极坐标系 $O(\sigma,\alpha_1,\alpha_2)$ 中,需要划分方位角 α_1 和高度角 α_2,角度误差控制在一定范围内,为了不丢失图像信息,方位向角度、高度向角度 $\Delta\alpha$ 均满足如下关系:

$$\Delta\alpha \leqslant \frac{\lambda_{\min}}{4L} \tag{4}$$

其中，$\lambda_{\min}$ 为最高频率对应的最小波长，L 为孔径长度。

对于全孔径三维成像，以二维全孔径中心 $\upsilon_{00}(x_0,y_0)$ 为原点，建立三维极坐标系 $O(\sigma,\alpha_1,\alpha_2)$，成像区域中某点像素值，可通过相干积累获得：

$$h(\upsilon_{00},\sigma,\alpha_1,\alpha_2)=\sum_{m=1}^{P}\sum_{n=1}^{Q}g(t,\upsilon_{mn})\cdot\delta(t,\tau_{mn}) \tag{5}$$

式中：$g(t,\upsilon_{mn})$为二维均匀面阵中第 m 行第 n 列天线的回波数据，τ_{mn} 为像素点到天线的回波延时。

若两维全孔径被分成 $p\times q$ 个$(P/p)\times(Q/q)$大小的子孔径时，分别以各子孔径中心 $\upsilon'_{ij}(x_i,y_j)$为原点，建立三维极坐标系 $O'(\sigma'_{ij},\alpha'_{1,ij},\alpha'_{2,ij})$，则 $P(\sigma,\alpha_1,\alpha_2)$点的像素值可表示为：

$$h(\upsilon_{00},\sigma,\alpha_1,\alpha_2)=\sum_{i=1}^{p}\sum_{j=1}^{q}h(\upsilon'_{ij},\sigma'_{1,ij},\alpha'_{1,ij},\alpha'_{2,ij}) \tag{6}$$

FFBP 算法利用子孔径分级合并的方法，依据多级因式分解的思路划分子孔径，基于初级子孔径长度较小的特征，选取较大的初级波束角宽度，进而减少了第一级成像时的计算量，得到多组低分辨率子图像，在随后多级合成中通过插值的方式对上一级子图像进行累加合并，最终得到高分辨图像。FFBP 算法流程如图 3 所示。

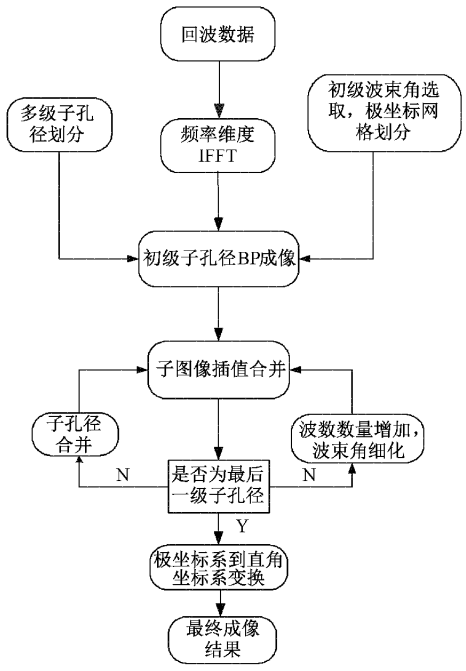

图 3　FFBP 算法流程图

2　四线程 CPU 及 CPU＋GPU 实现

基于四线程 CPU 平台 FFBP 算法实现步骤如下：

1)对回波数据频率维度作逆傅里叶变换(IFFT)脉压处理；

2)初级子孔径 BP 算法合成；

3)第二级至最后一级子孔径合成，主要技术思路为波束角划分及子孔径合并；

4)三维极坐标系转至直角坐标系，即投影到三维成像区域，获得三维复数图像；

基于 GPU 加速 CUDA 平台 FFBP 算法实现步骤如下：

1)将回波数据从 CPU 端复制到 GPU 端；

2)对回波数据进行频率维度 IFFT 脉压处理，调用 cufft 函数库的 cufftExecC2C 函数可实现信号的一维 IFFT，核心代码如下：

```
cufftHandle plan; //创建 FFT 计划
cufftPlan1d(&plan, iNumFrePad, CUFFT_C2C, (iNumNx*iNumNy)); //定义 FFT 计划
cufftExecC2C(plan, (cufftComplex *)pf2SignalIn_dev, (cufftComplex *)pf2SignalOut_dev, CUFFT_INVERSE); //执行 IFFT 逆变换
cufftDestroy(plan); //销毁 IFFT 计划
```

3)编写 cuda 核函数 kernel_PriSubapertureSyn，将步骤(2)处理得到的结果做 BP 处理；

4)编写 cuda 核函数 kernel_SubapertureSyn，将上一步结果，波束角细化，子孔径插值合并，GPU 加速中，用到了 CUDA 通用数学函数中的单精函数，核心代码如下；

```
zm2=RR_zz2_dev[threadIdx.x]/sqrtf(pow(__tanf(thta_Azimuthyy2_dev[blockIdx.x]*Pi/180.0f),2)+powf(__tanf(thta_Heightxx2_dev[blockIdx.y]*Pi/180.0f),2)+1); //z 方向坐标
xm2=zm2*__tanf(thta_Heightxx2_dev[blockIdx.y]*Pi/180.0f); //x 方向坐标
ym2=zm2*__tanf(thta_Azimuthyy2_dev[blockIdx.x]*Pi/180.0f); //y 方向坐标
pp21=floor((rm2-RR_zz1_dev[0])/dz0); //距离向索引位置
pp22=floor((thta_Height2-thta_Heightxx1_dev[0])/deltthta_Height1); //高度向索引位置
pp23=floor((thta_Azimuth2-thta_Azimuthyy1_dev[0])/deltthta_Azimuth1); //方位向索引位置
Href21={__cosf(4*Pi*(rm2-rs_down2)/c_lamata), __sinf(4*Pi*(rm2-rs_down2)/c_lamata)}; //向下像素点位置匹配滤波
Href22={__cosf(4*Pi*(rm2-rs_up2)/c_lamata), __sinf(4*Pi*(rm2-rs_up2)/c_lamata)};//向上像素点位置匹配滤波
```

5)循环重复步骤 4)过程，直至合成至最后一级子孔径；

6)三维极坐标系投影至三维空间直角坐标系，即笛卡尔直角坐标系；

7)将三维复数图像从 GPU 端复制到 CPU 端,进一步对距离向最大值投影到二维图像。

FFBP 算法的 CPU+GPU 实现流程如图 4 所示。

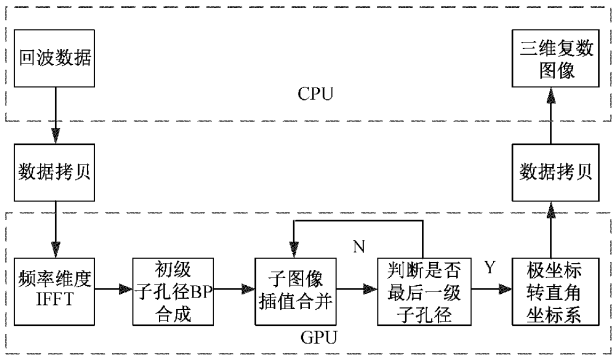

图 4　FFBP 算法的 CPU+GPU 实现流程

3　实验结果与分析

本文仿真环境:计算机配置处理器 12th Gen Inter(R) Core(TM) i7-12700 2.10 GHz,显卡 NVIDIA GeForce RTX 2060,64 bit Microsoft Windows 10 操作系统,Microsoft Visual Studio 2017(社区版),CUDA10.1。

1)为验证算法聚焦性能,基于四线程 CPU 与 GPU 加速平台对三维空间多点目标进行 FFBP 成像,分析对比两种平台的聚焦及成像效率之间差异。

成像仿真参数设置为:步进频连续波,频率范围 32～40 GHz,带宽 8 GHz,步进频率 100 MHz,阵列天线部署为 64×64 均匀收发同置的二维阵列,对水平向和竖直向,均进行 64 阵元,多级因式分解,64=16×4,其中 16=8×2,8=4×2,4=2×2,2=1×2。坐标原点 O 位于阵列中心,水平和竖直方向阵元间距均为 2.5 mm,成像区域为方位向 −0.312 5～0.312 5 m,高度向 −0.312 5～0.312 5 m,距离向 0.1～0.3 m。成像网格间隔划分为方位向 2.5 mm、竖直向 2.5 mm,距离向 4 mm,仿真多点目标[−0.1 0.1 0.2]、[0 0.1 0.2]、[0.1 0.1 0.2]、[−0.1 0 0.2]、[0 0 0.2]、[0.1 0 0.2]、[−0.1 −0.1 0.2]、[0 −0.1 0.2]、[0.1 −0.1 0.2]。

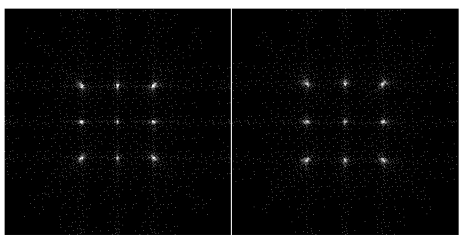

图 5　CPU 平台及 GPU 加速 CUDA 平台多点目标 XOY 面成像

左图为四线程 CPU 编程,右图为 GPU 加速 CUDA 平台编程的成像结果,由图 5 可知,得出表 1 数据。

表 1　多点目标两种平台图像评价指标

计算平台	图像熵	图像对比度
CPU(四线程)	2.790 2	1.054 2
GPU 加速	2.893 3	1.053 8

由图 5 及表 1 可知两种平台多点目标的图像聚焦效果均较好,并且基本一致,图像熵与图像对比度数值基本相同,从而验证 FFBP 算法的可靠性。

表 2　多点目标两种平台计算时间

计算平台	时间/ms
CPU(四线程)	5 332
GPU 加速	312

由表 2 可知两种平台多点目标的计算效率不同。GPU 加速 CUDA 平台编程与四线程 CPU 编程相比,加速比为 17 倍,进而验证 GPU 加速的有效性。

2)电磁仿真软件,分辨力板建模和仿真。

仿真模型参数设置为:步进频连续波,频率范围 30～38 GHz,带宽 8 GHz,步进频率 125 MHz;阵列天线部署为 96×96 均匀收发同置的二维阵列,对水平和竖直向,均进行 96 阵元,多级因式分解,96=24×4,其中 24=8×3,8=4×2,4=2×2,2=1×2。坐标原点 O 位于阵列中心,阵元扫描间隔 0.004 m,成像区域为方位向 −0.25～0.25 m,高度向 −0.25～0.25 m,距离向 0.2～0.4 m。成像网格间隔划分为方位向 2.5 mm、竖直向 2.5 mm,距离向 5 mm。

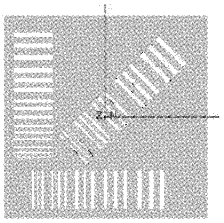

图 6　仿真分辨力板模型示意图

如图 6 为分辨力板仿真模型,分别为水平、竖直、斜 45 方向 3、4、5、6、7 mm 的分辨力。

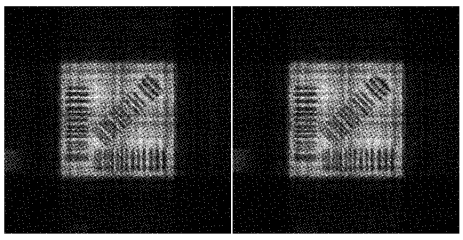

图 7　CPU 平台及 GPU 加速 CUDA 平台分辨力板仿真目标 XOY 面成像

由图 7 及表 3 可知,两种平台分辨力板仿真目标的图像聚焦效果,基本一致,图像熵与图像对比度数值相同,从

而证明了 GPU 实现仿真目标真实数据的可靠性。

表 3　分辨力板仿真目标两种平台图像评价指标

计算平台	图像熵	图像对比度
CPU(四线程)	3.976 6	0.222 3
GPU 加速	3.975 8	0.222 7

表 4　分辨力板仿真目标两种平台计算时间

计算平台	时间/ms
CPU(四线程)	7 440
GPU 加速	219

由表 4 可知，两种平台分辨力板仿真目标数据的计算效率不同。GPU 加速 CUDA 平台编程与四线程 CPU 编程相比，加速比为 33.97 倍。

4　结　　论

本文针对 FFBP 算法，基于 GPU 加速 CUDA 编程与四线程 CPU 编程的多点目标相比，图像对比度和图像熵，基本相同，聚焦效果较好，从而验证 FFBP 加速算法的正确性。在两种平台下，验证了分辨力板仿真模型的成像结果及计算效率，GPU 加速 CUDA 编程加速比是四线程 CPU 编程的 33.97 倍。本文由于是通过因式分解的方式分解阵元，故不同的阵元数量，需尝试不同的因式分解，寻求最佳的加速比。体现了算法的准确性和高效性，为雷达信号实时成像奠定了基础，进而应用于毫米波安检安防领域。

参考文献

[1] ULANDER L M H, HELLSTEN H, STENSTROM G. Synthetic-aperture radar processing using fast factorized back-projection[J]. IEEE Transactions on Aerospace and Electronic Systems, 2003, 39(3): 760-776.

[2] 潘丰，王文静，孟祥新，等. 基于 SAR 成像的 FBP 算法，FFBP 算法三维快速重建[J]. 电子测量技术，2020，43(4):6.

[3] 杨啸宇，高敬坤，邓彬，等. 基于 GPU 的毫米波雷达近场阵列成像技术研究[J]. 电子测量技术，2018，41(11):5.

[4] 刘鹏飞，陆必应，孙鑫，等. 三维快速因式分解后向投影算法[J]. 现代雷达，2016，38(1):36-41.

[5] 刘鹏飞. 穿墙三维成像技术研究[D]. 长沙：国防科技大学，2014.

[6] 班阳阳，张劲东，陈家瑞，等. 后向投影成像算法的 GPU 优化方法研究[J]. 雷达科学与技术，2014(6): 659-665.

[7] 姜晓龙，王建，宋千，等. 基于 GPU 的后向投影 SAR 成像算法[J]. 雷达科学与技术，2014(4):350-357.

[8] 陈彦民，韩阔业，江冕，等. 基于 GPU 平台的多雷达自适应主瓣干扰对消算法实现[J]. 国外电子测量技术，2018(10):103-108.

[9] 孟大地，胡玉新，石涛，等. 基于 NVIDIA GPU 的机载 SAR 实时成像处理算法 CUDA 设计与实现[J]. 雷达学报，2013，2(4):481-491.

[10] 苟立婷，李勇，朱岱寅，等. 基于 GPU 的圆寂视频 SAR 实时成像算法[J]. 雷达科学与技术，2019(10): 550-563.

[11] 刘百玲，江海清，倪书爱. 基于 GPU 的 ISAR 成像算法实现[J]. 电子测量技术，2015(8):76-78.

[12] 景国彬，张云冀，李震宇，等. 基于 GPU 的 SAR 回波仿真高效实现方法[J]. 系统工程与电子技术，2016(11): 2493-2498.

[13] 笪敏，孟祥新，李霆，等. 基于 CPU＋GPU 混合架构的改进型距离徙动算法设计与实现[J]. 电子测量技术，2020，43(3):5.

[14] 姜晓龙. 基于 GPU 的超宽带 SAR 实时成像技术研究[D]. 长沙：国防科学技术大学，2014.

[15] 梁美彦，任竹云，李光辉，等. GPU 加速条件下基于相位补偿反投影算法的太赫兹 ISAR 成像[J]. 红外与毫米波学报，2022，41(2):448.

作者简介

潘丰，硕士研究生，工程师，主要研究方向为合成孔径雷达成像算法、GPU 高性能计算等。

E-mail：PanfengXD_1992@163.com

高伟，博士研究生，高级工程师，主要研究方向为信号处理、雷达成像算法、雷达系统设计等。

E-mail：gaowei@microcreative.org

罗俊，博士研究生，高级工程师，主要研究方向为芯片设计、射频、雷达算法、雷达系统设计等。

E-mail：luojun@microcreative.org

刘文冬，博士研究生，高级工程师，主要研究方向为微波器件、天线设计等。

E-mail：lwd@microcreative.org

周春元，博士研究生，高级工程师，主要研究方向为硅基毫米波的相控阵电路设计研究等。

E-mail：zcy@microcreative.org

张慧，博士研究生，高级工程师，主要研究方向为数字芯片设计研发、系统设计等。

E-mail：zhanghui@microcreative.org