Website：ycyk.brit.com.cn

基于多核CPU的雷达导引头数字化实时仿真研究

苏灏杨[1,2]，夏伟杰[1,2]，吴　雪[1,2]，王　宇[1,2]
（1 雷达成像与微波光子技术教育部重点实验室 南京 210016；
2 南京航空航天大学电子信息工程学院 南京 210016）

摘要：雷达导引头仿真系统在导引头精确打击目标的过程中起着重要的作用。随着仿真系统的复杂度不断提升以及数据处理需求的日益增长，传统的串行计算仿真技术已难以满足雷达导引头数字仿真系统对实时性的严格要求。针对现有雷达导引头仿真过程中耗时过长的问题，本文提出了一种全流程的数字化实时仿真方法。首先，将传统全流程仿真架构的核心部分——接收控制系统指令、接收回波仿真数据、SAR（Synthetic aperture Radar，合成孔径雷达）成像处理、成像结果上传与界面动态更新进行流水并行化。其次，利用OpenMP（开放式多处理）多核并行模型，对SAR成像算法主要步骤进行多核CPU（Central Processing Unit，中央处理器）并行处理。然后，引入高性能数学计算库FFTW3（西方最快傅里叶变换第3版）快速实现成像算法的傅里叶变换，加快SAR成像算法处理速度。最后仿真结果表明：该全流程的设计方法相较于传统的串行仿真，加速比达到100倍左右，同时加速前后的SAR图像相似度接近于1。在处理精度和效果一致的前提下，该方法能够完成雷达导引头系统的全流程实时仿真，具有较好的工程应用前景。

关键词：雷达导引头；SAR仿真；多核CPU；并行计算；实时仿真
中图分类号：TN955　　**文献标志码：**A　　**文章编号：**2095-1000(2025)02-0092-08
DOI：10.12347/j.ycyk.20241129002
引用格式：苏灏杨，夏伟杰，吴雪，等. 基于多核CPU的雷达导引头数字化实时仿真研究[J]. 遥测遥控，2025，46(2)：92–99.

Research on Digital Real-Time Simulation of Multi-Core CPU-Based Radar Seeker

SU Haoyang，XIA Weijie，WU Xue，WANG Yu
（1 Key Laboratory of Radar Imaging and Microwave Photon Technology, Ministry of Education, Nanjing 210016
2 College of Electronic Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016）

Abstract: The radar seeker simulation system is crucial for the process of the seeker striking the target accurately. As simulation systems become more complex and data processing demands grow, traditional serial computing methods can no longer satisfy the strict real-time requirements of radar seeker digital simulations. To address the challenge of lengthy simulation times in the radar seeker simulation process, this paper proposes a full-process digital real-time simulation method. Firstly, the core components of the traditional full-process simulation architecture—receiving and controlling system commands, simulating echo data reception, SAR imaging processing, uploading imaging results, and dynamically updating the user interface—are restructured into a pipeline-based parallelization framework. Secondly, the SAR imaging algorithm's primary steps are parallelized using the OpenMP multi-core parallel programming model on multi-core CPUs. Furthermore, the high-performance mathematical computing library FFTW3 is introduced to quickly realize the Fourier transform of the imaging algorithm to accelerate the processing speed of the SAR imaging algorithm. Finally, the simulation results show that compared with the traditional serial simulation, the acceleration ratio of the whole process design method reaches about 100 times, and the similarity of SAR images before and after acceleration is close to 1. Under the premise of consistent processing accuracy and effect, this approach enables full-process real-time simulation of the radar seeker system, showcasing promising prospects for practical engineering applications.

Keywords: Radar seeker; SAR simulation; Multi-core CPUs; Parallel computing; Real-time simulation

基金项目：国家自然科学基金（62301260）
收稿日期：2024-11-29；**修回日期：**2025-01-09

Citation: SU Haoyang, XIA Weijie, WU Xue, et al. Research on Digital Real-Time Simulation of Multi-Core CPU-Based Radar Seeker[J]. Journal of Telemetry, Tracking and Command, 2025, 46(2): 92–99.

0 引言

在当今日益复杂的战争环境中，精确制导武器已成为关键的作战力量，而雷达导引头作为这些武器的核心设备，发挥着至关重要的作用[1]。通过雷达导引头，制导武器可以准确捕捉目标位置与特征，并引导火力对敌方目标进行高效、精确的打击，从而大幅提高作战效能，确保迅速摧毁敌方装备。

在雷达研发过程中，评估设备性能与技术参数是至关重要的环节。然而，传统的雷达导引头整机测试往往需要与弹体及其它载体协同完成，这一过程不仅消耗大量资源，且操作复杂、成本高昂[2]。随着计算机技术和仿真技术的不断进步，雷达导引头系统的研究逐步从传统的实物测试转向数字化仿真与分析。技术人员逐渐能够实时模拟雷达导引头的工作过程，准确获取目标与导弹的飞行轨迹，从而对仿真生成的数据进行全面分析与深入研究，为系统性能优化提供重要支持[3]。这不仅大大减少了实物实验的资源消耗，还有效保障了雷达导引头的性能，从而为进一步提升作战能力提供了有力支持。而传统雷达导引头数字化仿真面临的突出问题之一是实时性，仿真时间越长，所需的人力资源和试验成本就越高，并且难以实时复现导引头在实际条件下的运动状态[4]。

近年来，随着现代战争对精确打击和高效侦察需求的提高，雷达导引头数字化仿真研究备受关注。国内学者在仿真建模和高性能计算优化方面取得显著成果。张军涛等人使用Matlab实现了雷达导引头动态工作过程的仿真，并对其关键模块的性能进行了分析[5]。赵佳琪等人通过设计多通道雷达导引头信号处理系统[6]，为提高仿真效率提供了支持。在并行计算方面，国内研究结合OpenMP（开放式多核处理）并行模型，提出了基于数据分块与循环并行的策略[7–9]，显著加速了仿真执行。耿昭谦等人从高性能计算软硬件技术出发，分析了CPU（中央处理器）并行处理的优势，并展望了其在雷达信号处理中的应用前景[10]。罗政等人进一步研究了多核和众核架构下的并行算法，为性能优化提供了重要的理论依据[11]。国外研究在多核CPU、GPU（Graphics Processing Unit，图形处理器）及SAR（合成孔径雷达）成像加速方面取得突破，评估SAR成像算法在不同软硬件上的性能，并利用CPU、GPU实现高效并行化处理[12–13]，显著提升了成像效率。

在现有的雷达仿真加速方案中，尽管各方案分别探讨了仿真软件的实现和雷达信号处理算法的加速策略，但尚未从整体上对雷达导引头仿真闭环系统的全流程加速问题进行系统化分析。尤其在确保逻辑交互正确性的前提下，现有方案未能有效结合并行与串行操作来提升整体仿真效率。为解决这一问题，本文提出了一种雷达导引头全流程的数字化实时仿真方法。相比于传统的仿真加速方法，该方法基于核心运算模块的独立性和并行性，设计了全流程并行化架构，并对传统SAR成像算法进行了改进。该方法充分发挥了各运算模块的独立性与并行性，最终实现了雷达导引头系统的实时仿真。

1 全流程并行化架构设计

传统的雷达导引头仿真系统流程如图1所示，该系统以固定周期作为仿真时间步进，完成整个闭环仿真流程。考虑到雷达参数和图像数据传输的高可靠性需求，本文仿真系统采用TCP/IP协议，实现雷达导引头、控制系统与散射源仿真系统之间的高效数据交互。仿真过程从控制系统指令发出开始，整个流程包括回波配置参数计算、散射源回波仿真、雷达导引头成像处理以及界面更新显示等操作，这些操作构成了一个周期性的闭环节拍仿真。

在传统雷达导引头仿真系统的架构设计中，与任务1相比，任务2中的SAR成像处理和任务3中的界面显示的仿真耗时更长，这是导致仿真系统执行速度缓慢的主要原因。当雷达导引头的工作次数增加时，成像任务的负荷也随之加大，若继续采用串行处理流程，将会增加时间和内存资源的浪费，严重影响仿真系统的实时性。为了解决这一问题，流水线设计方式为并行处理系统提供了一种有效的方法。该技术通过将输入任务分解为一系列相互独立的子任务，并将其按顺序传

递至流水线中的不同处理阶段，同时执行，从而实现任务在时间上的重叠，达到操作级并行处理的效果[14]。

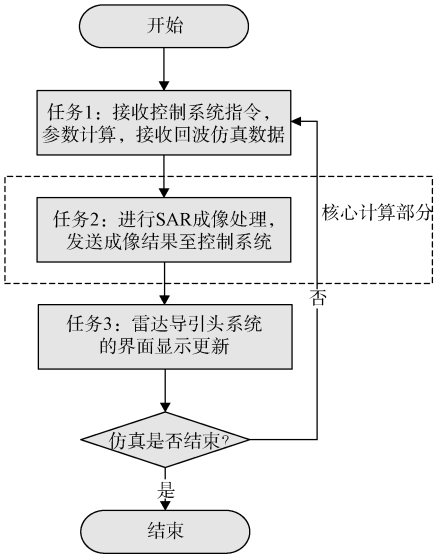


图 1 传统的雷达导引头仿真系统流程

Fig. 1 Traditional radar seeker simulation system process

为了进一步优化雷达导引头的仿真性能，引入线程池技术，实现流水并行化的全流程，并行处理各个模块，实现仿真速度的提升。线程池作为一个线程管理容器，预先创建多个线程并将其置于空闲状态，等待任务到来时动态分配。任务执行完成后，线程并不会被销毁，而是返回线程池中，继续等待新的任务分配，减少了线程频繁创建和销毁带来的时间和空间上的系统资源浪费[15]。通过创建成像算法的线程池，本文实现了全流程的流水并行设计，如图 2 所示。

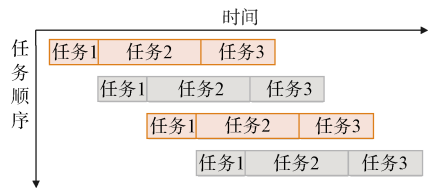


图 2 全流程流水并行设计图

Fig. 2 Whole process flow parallel design

该线程池通过循环利用已创建的线程技术，将执行成像任务与新任务的创建步骤进行了有效分离。当任务 1 执行完毕时，雷达导引头系统将成像数据放入线程池任务队列中，同时开始接收下一张图像数据，此时成像算法与指令接收、参数计算、回波数据接收及界面显示等过程流水并行进行。此过程不断重复，当成像算法计算出一张图像时，该图像数据将存入共享内存，在与控制系统的指令交互中按固定节拍发送，以提高整个雷达导引头闭环过程的仿真速度。为了有效地管理线程池，在仿真系统架构中设计的成像线程池详情如图 3 所示。

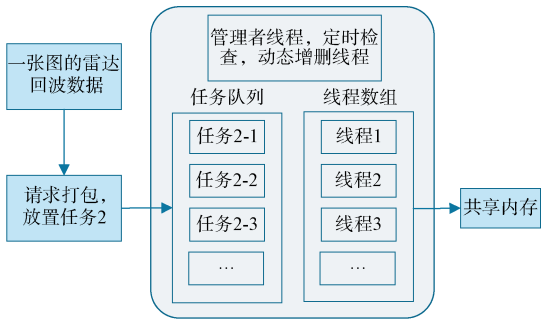


图 3 雷达 SAR 成像处理的线程池管理

Fig. 3 Thread pool management for radar SAR imaging processing

当成像数据传入线程池任务队列时，线程池管理器首先检查是否有空闲的成像算法线程可用。如果有，则将任务 2 分配给该线程；如果没有且线程数量未达上限，则创建新线程并分配任务 2。当系统中的线程数已达到最大限制，而任务 2 的队列尚未满载时，任务 2 将被暂时存入队列，等待空闲线程处理。线程在完成任务 2 后，将从队列中获取下一个待处理任务。

然而，在传统 SAR 成像算法的仿真过程中，由于场景范围较大、分辨率要求高、脉冲数量多以及采样点密集，导致雷达导引头仿真系统在执行单次任务 2 时的处理速度较为缓慢。仅依赖于流程的并行化无法满足系统对实时性的严格要求。作为任务 2 的核心计算环节，成像处理的效率亟须通过进一步的并行优化来提升，以确保系统能够在规定的时间内高效地完成仿真任务。

2 SAR 成像算法原理及并行加速

2.1 传统的 SAR 成像算法原理

由于导引头 SAR 天线尺寸较小，其全孔径方位分辨率通常较高，超出了匹配参考图的精度要求。为了保证后续景象匹配、弹体定位等处理的正常进行，同时满足制导率要求，针对修正 INS（Inertial Navigation System，惯性导航系统）误差的 SAR 成像算法不宜过于复杂。因此，本文采用

大斜视子孔径频域成像算法，该算法适用于子孔径成像，同时内部运算只包含复乘和FFT（快速傅里叶变换）运算，不涉及插值处理，易于工程实现[16-17]。

在成像模式下，散射源返回的基带信号通过大斜视子孔径频域成像处理算法进行处理。该算法通过距离向处理、方位向处理和图像处理等步骤，最终生成图像及相关的处理结果参数。具体的处理流程如图4所示。

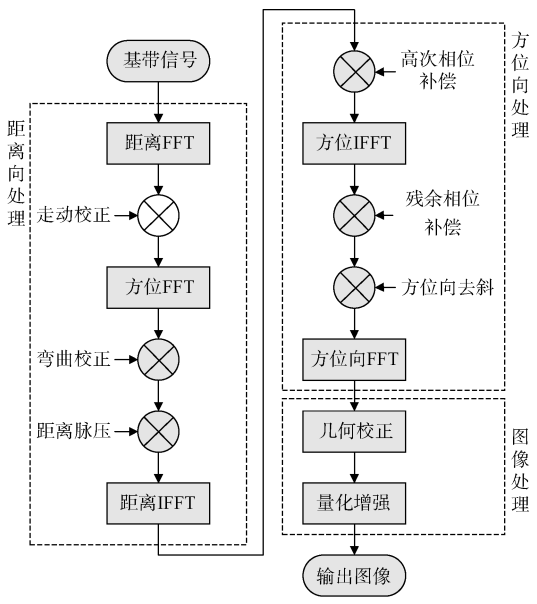


图4　成像信号处理流程

Fig. 4　Imaging signal processing flow

假设雷达导引头信号脉内调制体制为线性调频，则其基带信号时域的表达式如下：

$$S_{\mathrm{t}}(t_{\mathrm{r}},t_{\mathrm{n}})=w_{\mathrm{r}}\left(t_{\mathrm{r}}-\frac{2R(t_{\mathrm{n}})}{c}\right)w_{\mathrm{n}}(t_{\mathrm{n}}-t_0)\cdot \exp\left(\mathrm{j}\pi u\left(t_{\mathrm{r}}-\frac{2R(t_{\mathrm{n}})}{c}\right)^2\right)\exp\left(-\mathrm{j}\frac{4\pi R(t_{\mathrm{n}})}{\lambda}\right) \tag{1}$$

式中，w_{r}表示距离窗函数，t_{r}表示距离时间，w_{n}表示方位窗函数，t_{n}表示方位时间，$R(t_{\mathrm{n}})$表示无近似瞬时斜距，c为光速，t_0为波束中心穿过目标的时刻，λ为波长，u为调频斜率。设f_{r}为距离频率，f_{c}为信号载频，对公式（1）在距离方向上进行傅里叶变换，得到了该信号在距离频域中的表示形式，如下所示：

$$S_{\mathrm{ft}}(f_{\mathrm{r}},t_{\mathrm{n}})=W_{\mathrm{r}}(f_{\mathrm{r}})w_{\mathrm{n}}(t_{\mathrm{n}}-t_0)\cdot \exp\left(-\mathrm{j}\pi\frac{f_{\mathrm{r}}^2}{u}\right)\exp\left(-\mathrm{j}\frac{4\pi(f_{\mathrm{c}}+f_{\mathrm{r}})R(t_{\mathrm{n}})}{c}\right) \tag{2}$$

式中，W_{r}表示距离频率窗函数。信号的线性距离走动校正函数可以表示为：

$$H_{\mathrm{lrwc}}(f_{\mathrm{r}},t_{\mathrm{n}})=\exp\left(-\mathrm{j}\frac{4\pi v\sin\theta_0(f_{\mathrm{c}}+f_{\mathrm{r}})t_{\mathrm{n}}}{c}\right) \tag{3}$$

式中，波束射线指向的斜视角为θ_0，信号的距离弯曲校正函数可以表示为：

$$H_{\mathrm{rcc}}(f_{\mathrm{r}},f_{\mathrm{n}};R_0)=\exp\left(\mathrm{j}2\pi f_{\mathrm{r}}\frac{2R_0\cos\theta_0}{c}\frac{(f_{\mathrm{c}}/c)-(f_{\mathrm{n}}+f_{\mathrm{Dc}})\sin\theta_0/2v}{\sqrt{(f_{\mathrm{c}}/c)^2-((f_{\mathrm{n}}+f_{\mathrm{Dc}})/2v)^2}}\right)\cdot \exp\left(\mathrm{j}2\pi f_{\mathrm{r}}\frac{2R_0\sin^2\theta_0}{c}\right)\exp\left(-\mathrm{j}2\pi f_{\mathrm{r}}\frac{2R_0}{c}\right) \tag{4}$$

式中，R_0表示雷达与波束照射中心之间的斜距，$f_{\mathrm{Dc}}=2v\sin\theta_0/\lambda$为多普勒中心频率。信号的距离脉压和二次距离脉压函数可以通过以下方式表示：

$$H_{\mathrm{rc\_src}}(f_{\mathrm{r}};R_0)=\exp\left(\mathrm{j}\pi\frac{f_{\mathrm{r}}^2}{K}\right)\cdot \exp\left(-\mathrm{j}\pi f_{\mathrm{r}}{}^2\frac{2R_0\cos\theta_0}{c^2}\frac{(f_{\mathrm{c}}\sin\theta_0/c-(f_{\mathrm{a}}+f_{\mathrm{Dc}})/2v)^2}{\left((f_{\mathrm{c}}/c)^2-((f_{\mathrm{a}}+f_{\mathrm{Dc}})/2v)^2\right)^{\frac{3}{2}}}\right) \tag{5}$$

通过上述距离向处理函数，将信号转换到二维频域。接着，再将二维频域上的信号与式（3）和式（4）相乘，以进一步处理信号，最终得到距离向处理后的信号在二维频域中的表示形式：

$$S_{\mathrm{tf}}(t_{\mathrm{r}},f_{\mathrm{n}};R_0)=\sin\mathrm{c}\left(B_{\mathrm{r}}\left(t_{\mathrm{r}}-\frac{2(R_0+x_m\sin\theta_0)}{c}\right)\right)\cdot \exp\left(-\mathrm{j}4\pi R_0\cos\theta_0\sqrt{(f_{\mathrm{c}}/c)^2-((f_{\mathrm{n}}+f_{\mathrm{Dc}})/2v)^2}\right)\cdot \exp\left(-\mathrm{j}\frac{2\pi f_{\mathrm{n}}(R_0\cos\theta_0+x_m)}{v}\right)\cdot \exp\left(-\mathrm{j}\frac{2\pi f_{\mathrm{Dc}}(R_0\sin\theta_0+x_m)}{v}\right)W(f_{\mathrm{n}}) \tag{6}$$

式中，B_{r}为信号带宽，$R_0+x_m\sin\theta_0$为距离向处理后的目标位置。对二维频域表达式进行方位向的高阶相位补偿和方位压缩后，得到聚焦后的信号。

最后，采用反向投影的快速几何校正方法，将聚焦后的两维数据从斜距平面映射到地距平面，并进行量化，从而得到最终输出的图像。

2.2　算法主要并行化过程

随着多核处理器技术的不断进步，多核多线程编程技术逐渐成熟，极大地推动了多核计算机

在运算性能上的提升，多核CPU与GPU技术被广泛应用于并行处理中。相比于GPU技术，在多核CPU上进行实时信号处理，不仅提高了灵活性，还降低了成本，更加适合应对复杂流程控制问题。因此，多核CPU作为数据处理核心的方案为相对更优的方案[18]。

OpenMP是一种支持C、C++和Fortran（公式翻译器语言）的并行编程模型，专为多核CPU和共享内存架构设计，提供了一种高层次的并行化标准。研究人员可以利用OpenMP简洁高效地实现多线程并行编程，相较于其他多线程编程工具，OpenMP在降低代码入侵度和提升灵活性方面具有明显优势[19]。因此，本文采用OpenMP模型对成像算法中多次重复的计算过程进行了优化，编程框架采用如图5所示的多线程派生连接（Fork-join）模式进行并行计算。在程序执行过程中，当主线程进入并行区域时，它会派发或调度空闲线程，生成一组子线程，从而形成新的线程团队。每个子线程负责执行分配的并行任务，任务完成后，所有子线程将在并行区域结束时汇合并返回主线程。

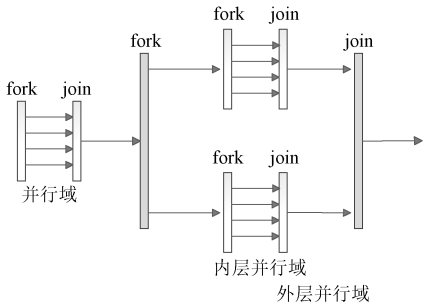


图5 OpenMP并行原理图

Fig. 5 OpenMP parallel schematic

在本文设计的雷达导引头仿真系统中，生成一幅图像所需的回波数据包含8 192个脉冲和4 096个采样点，构成了一个大小为8 192×4 096的二维矩阵。针对成像算法中的脉冲压缩过程，本文结合OpenMP模型，设计了图6所示的线程资源分配方案。该方案通过for循环，可以启用m个线程并行处理m个等大小的小矩阵，充分利用了多线程并行计算的优势，成像算法的其他步骤也使用了相同的优化策略。

3 傅里叶变换计算的优化

当成像任务较多时，雷达成像处理各个模块中存在大量傅里叶变换运算任务，仅通过OpenMP模型并行成像算法步骤，难以达到实时仿真的目的。

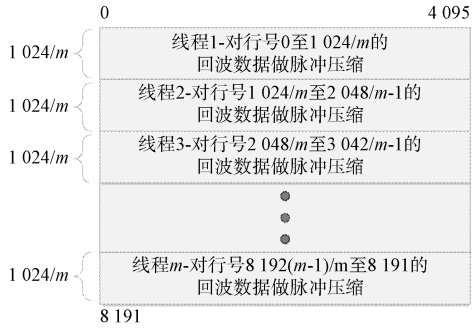


图6 脉冲压缩算法资源分配

Fig. 6 Pulse compression algorithm resource allocation diagram

为进一步提高傅里叶变换运算效率，对传统的编程环境下的FFT计算方式，引入了第三方库FFTW3。FFTW3能够适应不同系统软硬件环境，并支持多线程并行和分布式存储并行，其计算性能优于其他FFT计算库。FFTW3不仅具备跨平台的优势，还支持多种数据格式的FFT运算，表现出极高的速度和全面性，非常适合应用于复杂的计算场景[20]。

在FFTW3函数库中，为了尽可能提升运算效率，其计算过程不会采用在所有硬件平台上统一的算法流程。相反，FFTW3会根据底层硬件的特点，动态调整函数流程中的部分细节，从而最大限度地优化性能表现。整个计算过程可分为两个主要阶段。首先，FFTW3函数库通过一个“计划者”模块分析当前硬件平台，学习并选择最优的计算方式，随后创建一个“计划”来存储这些信息。在后续的计算中，函数库将根据“计划”中存储的优化方案执行数据运算，以实现最快的计算速度。由于一个“计划”通常会被多次重复使用，尽管初次创建计划可能需要消耗一定的资源，但其重复应用时的高效性能够大大弥补这一开销。具体的运算步骤如图7所示。

针对成像算法中的脉冲压缩过程，结合OpenMP模型框架和FFTW库，启用m个线程并行处理，每次对一行数据进行快速傅里叶变换，将滤波器系数的FFT结果相乘后，再执行IFFT。其余成像处理模块步骤的FFT也使用了相同方式优化，以下是多线程脉冲压缩的计算过程实现，如表1所示。

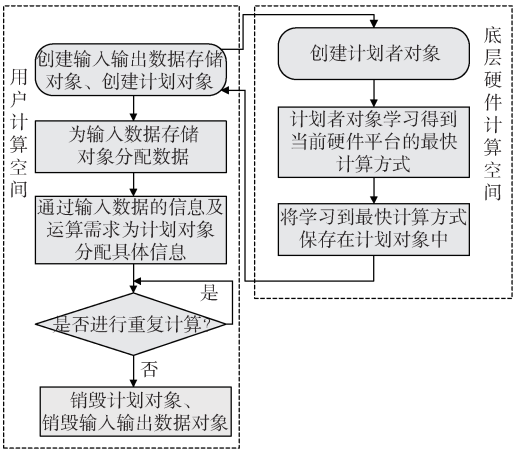


图 7　FFTW3 函数库计算流程

Fig. 7　FFTW3 function library calculation process

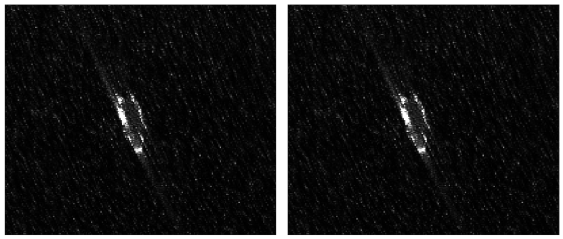
表 1　多线程脉冲压缩计算过程表格

Table 1　Multi-threaded pulse compression calculation process table

名称：多线程脉冲压缩过程的 FFTW 使用方式
输入： 1. 构造二维复数数据的矩阵 PC_FFT，大小为 $r_{row}\times c_{col}$，设置的并行线程数为 m； 2. 使用数组 data_rows，用于存储每行 FFT 数据；创建 FFT 的计划，用于执行 FFT 和 IFFT； 循环迭代： 1. 使用 OpenMP 进行并行化； 2. 对每行的 data_rows 数据使用 FFTW3 库中的 fftw_plan_dft_1d 函数执行 FFT 过程； 3. 结果与原数据逐点复数相乘，执行 IFFT； 输出： 每行数据的脉冲压缩结果，更新到矩阵 Result

4　系统仿真结果

本文的仿真实验是在 Microsoft Visual Studio（微软可视化工作室）2017 开发环境中进行的，设置 SAR 成像模式下仿真工况：雷达平台的初始位置设为 $(-2\,000, -10\,000, 10\,000)m$，雷达平台运动速度矢量设为 $(0, 1\,000, 0)m/s$，雷达平台的姿态角均设为 0°；目标设置在原点处且保持静止。该工况下参数计算得到的信号脉冲个数为 8 192，脉冲重复频率为 8 192 Hz，载频为 15 GHz；脉宽设为 20 μs；带宽为 60 MHz；采样率为 75 MHz，采样波门前沿为 12 930 m，成像分辨率为 5 m；相干处理周期为 500 ms；仿真场景设置如下，海况为三级海况，电磁波俯仰入射角设为 45°，方位入射角为 10°，场景大小为 2 048×2 048，分辨率为 4 m，并向场景中添加航母模型一个。以上条件在加速前和加速后，雷达导引头仿真处理出的 SAR 成像舰船目标的成像结果清晰，仿真加速前和加速后的成像效果分别如图 8 所示。



（a）仿真加速前　　（b）仿真加速后

（a）Before simulation acceleration　（b）After simulation acceleration

图 8　雷达导引头对海面舰船的 SAR 成像

Fig. 8　SAR imaging of ships on the sea surface by radar seeker

针对上述目标场景和仿真参数，本文使用了串行仿真单核 CPU、共享内存实现全流程并行仿真[21]、全流程并行仿真与 8 核 CPU 的加速方法，然后对核心计算模块进行了逐一测试，对不同部分的仿真耗时进行了评估。当 SAR 成像算法中线程数 m 为 8 时，见表 2 和表 3，全流程仿真加速效果显著，全流程开环（导引头读指令文件，使用递增的帧数模拟工作，只与散射源仿真系统进行交

表 2　主要步骤计算耗时表格

Table 2　The main step calculates the time consumption table

主要步骤	串行仿真单核 CPU/s	并行仿真共享内存/s	并行仿真 8 核 CPU/s
SAR 成像处理	710.3	120.7	50.1
回波计算	56.4	56.3	56.3
数据交互	231.9	4.2	2.9
界面显示	1 683.9	16.2	2.4
全流程开环	5 612.5	141.1	55.6
全流程闭环	6 301.2	643.9	370.2

表 3　主要步骤计算加速比

Table 3　The main step is to calculate the acceleration ratio

主要步骤	并行仿真共享内存	并行仿真 8 核 CPU
SAR 成像处理	5.9	14.2
数据交互	55.2	80.0
界面显示	103.9	701.6
全流程开环	39.8	100.9
全流程闭环	9.9	17.0

互）、闭环（导引头实时接收控制系统发送的指令文件，同时与散射源仿真系统进行交互，控制系统需要根据成像结果修正指令）的加速比分别为100.9、17.0。除各主要部分计算外，还有一些关于成像参数的相关计算、回波参数配置，这些部分通过优化内存、改变软件的编译环境为快速堆栈等方式，仿真速度也得到了提升。

本文的全流程仿真共生成60张图像，每张图像的脉冲数、脉冲重复频率相同。数据传输基于万兆速率进行，每帧数据交互的延迟控制在约1 ms以内，该延迟相较于雷达的工作周期几乎可以忽略不计。界面显示通过Windows API（视窗应用程序编程接口）中的PostMessage（投递消息）消息机制[22]实现，以确保界面的快速响应。在采用本文所提出的并行处理方法时，回波计算和数据交互的总时间小于60 s，能够满足实时将回波数据传输至SAR成像处理的需求。为评估该方法的实时性，本文通过计算SAR图像出图显示的总时间，即全流程开环时间，与脉冲回波计算时间和数据交互时间总和的比值，将其作为全流程仿真的平均实时性值。

根据表4的数据，使用8核CPU并行仿真的方法对全流程进行加速时，60张SAR图像的平均实时性值为0.939，满足雷达导引头系统对实时性的要求。此外，本文还采用了归一化交叉相关系数图像匹配算法[23]，计算出加速前后所有SAR图像的平均相似度为0.973，接近于1，表明加速前后的图像不会影响控制系统的图像匹配。因此，本文所提出的方法不仅显著提高了雷达导引头仿真系统的处理速率，而且能够在满足实时成像处理需求的同时，确保达到实时仿真效果。

表4 各方法的结果对比

Table 4 Comparison of the results of each method

结果对比	串行仿真 单核CPU	并行仿真 共享内存	并行仿真 8核CPU
平均实时性值	19.468	2.332	0.939
平均相似度	1	0.983	0.973

5 结束语

本文提出了一种雷达导引头系统的全流程实时仿真方法，对全流程架构进行了流水并行化设计，同时通过对SAR成像处理的核心计算部分完成多线程计算，最后将串行仿真与并行仿真（8核CPU）的时间进行对比，实验表明：在全流程开环情况下加速比达到100左右，全流程闭环情况下加速比达到17左右，全流程开环的平均实时性数值为0.939、图像的平均相似度为0.973。通过对最终的实验数据分析得出，本文提出的全流程实时仿真方法有效缩短了算法核心计算部分的运行时间，能够显著提升雷达导引头仿真系统的性能。

在以后的研究中，还可以考虑以下问题：算法中不同数据精度之间的速度差异，以及在可接受的误差范围内，降低数据精度以提升算法运行速度的可能性[24]；在进行大量的向量运算、矩阵运算时，相比于传统的C++编程方法，使用MKL（数学核心库）运行库进行运算，能够进一步提升运算效率[25]。

参考文献

[1] 陈卓. 雷达导引头仿真与关键技术研究[D]. 成都: 电子科技大学, 2019.

[2] 张晓东, 张林让. 基于GPU的相控阵雷达并行仿真技术[J]. 计算机仿真, 2019, 36(12): 20-24.
ZHANG Xiaodong, ZHANG Linrang. Parallel simulation technology of phased array radar based on GPU[J]. Computer Simulation, 2019, 36(12): 20-24.

[3] 严煜宇, 王学田, 高洪民, 等. 反舰导弹作战全数字仿真软件设计与实现[J]. 微波学报, 2020, 36(S1): 27-30.
YAN Yuyu, WANG Xuetian, GAO Hongmin, et al. Design and implementation of full-digital simulation software for anti-ship missile operations[J]. Microwave Journal, 2020, 36(S1): 27-30.

[4] 张彦彬, 丁晟, 高雁, 等. 基于CPU+GPU混合架构的实时成像系统设计与实现[J]. 太赫兹科学与电子信息学报, 2019, 17(1): 146-151.
ZHANG Yanbin, DING Sheng, GAO Yan, et al. Design and implementation of real-time imaging system based on CPU + GPU hybrid architecture[J]. Journal of Terahertz Science and Electronic Information, 2019, 17(1): 146-151.

[5] 张军涛, 李尚生, 刘晨飞, 等. 基于Matlab的反舰导弹雷达导引头动态工作过程仿真[J]. 舰船电子工程, 2022, 42(3): 96-99.
ZHANG Juntao, LI Shangsheng, LIU Chenfei, et al. Dynamic working process simulation of anti-ship missile radar seeker based on Matlab[J]. Ship Electronic Engineering, 2022, 42(3): 96-99.

[6] 赵佳琪. 多通道雷达导引头信号处理系统设计与实现[D]. 西安: 西安电子科技大学, 2024.

[7] 周剑. 多核CPU上雷达信号处理的设计与实现[D]. 南京: 南京理工大学, 2020.

[8] 常艳, 何涛, 朱占宇. 基于CPU+GPU混合架构的雷达信号处理方法[J]. 火力与指挥控制, 2024, 49(7): 80-85, 90.
CHANG Yan, HE Tao, ZHU Zhanyu. Radar signal processing method based on CPU+GPU hybrid architecture [J]. Fire and Command Control, 2024, 49(7): 80-85, 90.

[9] 秦华, 周沫, 察豪, 等. 软件雷达信号处理的多GPU并行技术[J]. 西安电子科技大学学报, 2013, 40(3): 145-151.
QIN Hua, ZHOU Mo, CHA Hao, et al. Multi-GPU parallel technology for software radar signal processing[J]. Journal of Xidian University, 2013, 40(3): 145-151.

[10] 耿昭谦, 朱虎明, 李旭明, 等. 基于高性能计算的雷达信号处理研究综述[J]. 电子科技, 2021, 34(9): 1-6.
GENG Zhaoqian, ZHU Huming, LI Xuming, et al. Overview of radar signal processing research based on high-performance computing[J]. Electronics, 2021, 34(9): 1-6.

[11] 罗政. 基于多核众核架构的并行雷达信号处理算法研究[D]. 西安: 西安电子科技大学, 2018.

[12] CRUZ H, VÉSTIAS M, MONTEIRO J, et al. A review of synthetic-aperture radar image formation algorithms and implementations: A computational perspective[J]. Remote Sensing, 2022, 14(5): 1258.

[13] DENHAM M, ARETA J, TINETTI F G. Synthetic aperture radar signal processing in parallel using GPGPU[J]. The Journal of Supercomputing, 2016, 72(2): 451-467.

[14] 王国庆, 申俊杰, 张旭峰. 基于DSP的宽带雷达多片流水分段脉压处理平台设计[J]. 现代电子技术, 2008(9): 41-44.
WANG Guoqing, SHEN Junjie, ZHANG Xufeng. Design of DSP-based wideband radar multi-chip pipeline segmented pulse compression processing platform[J]. Modern Electronic Technology, 2008(9): 41-44.

[15] 谢文杰, 艾赛江, 张映昊. 雷达数据实时处理软件多线程技术的改进[J]. 计算机应用, 2018, 38(S2): 250-253.
XIE Wenjie, AI Saijiang, ZHANG Yinghao. Improvement of multithreading technology in radar data real-time processing software[J]. Computer Applications, 2018, 38(S2): 250-253.

[16] 李震宇, 梁毅, 邢孟道, 等. 弹载合成孔径雷达大斜视子孔径频域相位滤波成像算法[J]. 电子与信息学报, 2015, 37(4): 953-960.
LI Zhenyu, LIANG Yi, XING Mengdao, et al. Phase filtering imaging algorithm in frequency domain for high squint subaperture of missile-borne synthetic aperture radar[J]. Journal of Electronics and Informatics, 2015, 37(4): 953-960.

[17] 曹晔, 闫海鹏, 张剑琦, 等. 高动态条件下舰船目标SAR成像算法研究[J]. 遥测遥控, 2019, 40(4): 40-48.
CAO Ye, YAN Haipeng, ZHANG Jianqi, et al. Study on SAR imaging algorithm of ship targets under high dynamic conditions[J]. Telemetry and Remote Control, 2019, 40(4): 40-48.

[18] 杨思军. 基于多核CPU的雷达信号并发处理架构设计[J]. 舰船电子对抗, 2021, 44(5): 62-66.
YANG Sijun. Design of radar signal concurrent processing architecture based on multi-core CPU[J]. Ship Electronic Countermeasures, 2021, 44(5): 62-66.

[19] 陆士浩. 基于多核CPU的雷达信号处理系统的设计与实现[D]. 南京: 南京理工大学, 2023.

[20] 魏梦瑶. 基于X86架构CPU的雷达信号处理算法研究[J]. 电子科技, 2017, 30(5): 55-57.
WEI Mengyao. Research on radar signal processing algorithms based on X86 architecture CPU[J]. Electronics, 2017, 30(5): 55-57.

[21] 滕琨. 基于GPU+CPU的雷达仿真系统设计与实现[D]. 西安: 西安电子科技大学, 2019.

[22] 寇惠云. 基于PC的成像系统显控终端设计及目标检测算法研究[D]. 成都: 电子科技大学, 2015.

[23] 张寅, 范君杰, 闫钧华, 等. 多成像模式下舰船目标SAR成像仿真[J]. 现代防御技术, 2022, 50(2): 113-120.
ZHANG Yin, FAN Junjie, YAN Junhua, et al. SAR imaging simulation of ship targets in multi-imaging mode[J]. Modern Defense Technology, 2022, 50(2): 113-120.

[24] MENG Dadi, HU Yuxin, SHI Tao, et al. Airborne SAR real-time imaging algorithm design and implementation with CUDA on NVIDIA GPU: Airborne SAR real-time imaging algorithm design and implementation with CUDA on NVIDIA GPU[J]. Journal of Radars, 2014, 2(4): 481-491.

[25] 邱瑾. 软件化雷达算法组件设计与实现[D]. 西安: 西安电子科技大学, 2020.

[作者简介]

苏灏杨　2001年生，硕士研究生。
夏伟杰　1979年生，教授，博士生导师。
吴　雪　2000年生，硕士研究生。
王　宇　2001年生，硕士研究生。

(本文编辑：傅　杰)