文章编号：1672-2892(2012)01-0098-05

基于 GPU 的 SAR 方位向信号分解的高效实现方法

张相广，吴长朋，高叶盛，王开志，郁文贤

(上海交通大学 电子工程系，上海 200240)

摘　要：提出了一种基于图形处理器(GPU)的 SAR 方位向信号分解的高效实现方法。SAR 方位向信号可以通过四参数 Chirplet 分解方法来分解。此方法的关键难题是计算量过大,计算量主要由 2 部分组成：构建 Chirp 原子库，以及 SAR 方位向信号在过完备库上分解的计算量。与传统的 CPU 相比，GPU 更加适用于密集型和大量数据并行化的计算。提出将算法的核心部分移植到 GPU 上进行并行计算，充分挖掘其运算潜能。结果表明：该方法与传统的基于 CPU 的算法相比有两位数以上的效率提升。

关键词：四参数 Chirplet 分解；合成孔径雷达；Chirp 原子库；图形处理器

中图分类号：TN957.51　　**文献标识码：**A

A highly efficient GPU-based method for decomposing SAR azimuth signal

ZHANG Xiang-guang，WU Chang-peng，GAO Ye-sheng，WANG Kai-zhi，YU Wen-xian

(Department of Electronic Engineering，Shanghai Jiaotong University，Shanghai 200240，China)

Abstract: A highly efficient Graphics Processing Unit(GPU) based method for decomposing SAR azimuth signal is presented. SAR azimuth signal is decomposed by the four-parameter Chirplet decomposition method. This method needs excessive computation mainly to construct Chirp atomic dictionary and decompose SAR azimuth signal in the over-complete dictionary. Compared with CPU, GPU is well suited for data parallel computations with high arithmetic intensity. A new method is introduced that takes full advantage of GPU's computation capability by transferring the key part of the algorithm to GPU.The experimental results shows that this method is more than ten times as fast as traditional CPU based algorithms.

Key words: four-parameter Chirplet decomposition; Synthetic Aperture Radar; Chirp atomic dictionary; Graphics Processing Unit

SAR 方位向信号可视为 1 组不同参数的 Chirp 信号的集合,如果将这些信号逐个分解,并分别进行聚焦处理,就能实现 SAR 精确成像。因此，四参数 Chirplet 分解方法能够按能量由大到小顺序，把 SAR 方位向信号分解成多个 Chirp 分量。此方法的关键是如何准确地估计 4 个参数。原子字典是完备的,字典的大小决定图像的分辨率,因此对信号的每一次分解都要进行大量的内积运算以决定在这一步中应该选取原子库中哪一个 Chirp 原子做分解。传统的基于 CPU 的分解方法面临挑战，因此有必要探索新的四参数 Chirplet 分解运算平台。

最近,图形处理器(Graphic Processing Unit,GPU)正在快速发展,在浮点计算能力和存储器带宽上相对于 CPU 都具有明显的优势。GPU 凭借多核共同驱动及很高的内存读写带宽，使其不仅应用于图形处理方面，在通用计算方面也可以提供更多的运算资源[1]。2007 年，NVIDIA 公司推出的统一计算平台 CUDA[2]是一种不需要借助图形学 API 就可以使用类 C 语言进行通用计算的开发环境和软件体系，增强了 GPU 的可编程性。

本文提出基于 GPU 的 SAR 方位向信号分解方法,将分解计算量最大的两大部分(Chirp 原子库的构建和 SAR 方位向信号在过完备库上分解)移植到 GPU 上进行并行计算，优化基于 GPU 的算法结构，使其与 GPU 的架构和 CUDA 编程模型相适应，充分发挥 GPU 强大的运算资源，提高运算效率。

1　时频原子分解算法

原子分解算法[3]也叫匹配追踪算法。

收稿日期：2011-06-28；修回日期：2011-09-03

令 H 表示 Hilbert 空间；定义过完备原子库：$D=\{g_\gamma\}_{\gamma\in\Gamma}$，其中，$\Gamma$ 是参数组 γ 的集合，g_γ 为由参数组 γ 定义的原子，且 $\|g_\gamma\|=1$。对于任意信号 $y\in H$，可以由它在 D 中原子上的正交投影来近似表示：

$$y=\langle y,g_\gamma\rangle g_\gamma+Ry \tag{1}$$

式中：$\langle y,g_\gamma\rangle g_\gamma$ 为 y 在 g_γ 方向上的正交投影；Ry 为投影后 y 的残余量。同样，也分解 Ry，如此反复，可得：

$$y=\sum_{n=0}^{M-1}\langle R^n y,g_{\gamma n}\rangle g_{\gamma n}+R^M y \tag{2}$$

在每一步分解中，都应该选择 $g_{\gamma n}$，使得 $\left|\langle R^n y,g_{\gamma n}\rangle\right|$ 最大，从而使残余 $R^{n+1}y$ 最小，即最佳的时频原子为：

$$g_{\gamma n}=\arg\max_{\gamma\in\Gamma}\left|\langle R^n y,g_\gamma\rangle\right| \tag{3}$$

如果字典 D 是完备的，则迭代结果将会收敛于 $y(t)$。

$$y(t)=\sum_{n=0}^{\infty}\langle R^n y,g_{\gamma n}\rangle g_{\gamma n} \tag{4}$$

常用的是 Gabor 时频原子字典和 Chirp 时频原子字典[4−5]，它们均由高斯单位能量函数变换得来，本文所用的 Chirp 原子为：

$$g_\gamma(t)=\frac{1}{\sqrt{d}}\mathrm{rect}\left(\frac{t-u}{d}\right)\mathrm{e}^{\mathrm{j}f(t-u)}\mathrm{e}^{\mathrm{j}\frac{k}{2}(t-u)^2} \tag{5}$$

式中：$\gamma\triangleq(d,u,f,k)$ 是原子的参数集；参数 d 是 Chirp 原子持续时间；f 是中心频率；k 是线性调频率；u 是时间中点；$\mathrm{rect}(t)$ 为矩形窗。

$$\mathrm{rect}(t)=\begin{cases}1, & t\in(0,1]\\ 0, & \text{others}\end{cases} \tag{6}$$

2　基于 GPU 的四参数 Chirplet 分解

2.1 算法流程

在具体的计算机实现中，数据都是离散化处理的。假定待分解的 SAR 方位向信号共有 N 个方位向上的采样 $S(k)$，$k=0,1,\cdots,N-1$。假定 Chirp 原子三参数 (d,f,k) 的取值范围分别为 $[d_{\min},d_{\max})$，$[f_{\min},f_{\max})$，$[k_{\min},k_{\max})$，三参数的间隔分别为 $step\_d$，$step\_f$，$step\_k$。出于方便的考虑，先把 u 设置为 0，本文所指的 Chirp 原子库是基于 $u=0$ 的，Chirp 原子时延在后面会另外考虑。

字典的长度 $szDict=\left(\frac{d_{\max}-d_{\min}}{step\_d}\right)\left(\frac{f_{\max}-f_{\min}}{step\_f}\right)\left(\frac{k_{\max}-k_{\min}}{step\_k}\right)$。这样就得到了离散化的 γ 参数集合 $\gamma_m\triangleq(d_m,f_m,k_m)$ 和连续时间 Chirplet $g_{\gamma m}(t)$，其中 $m\in[1,szDict]$。

第 1 步，对每个 Chirplet $g_{\gamma m}(t)$ 进行采样率为 f_s 的时域采样，第 m 个 Chirp 原子的采样点个数 $s_m=d_m\cdot f_s$。

把 $t=\left(-\frac{s_m}{2}+k\right)\cdot\frac{1}{f_s}$，$k\in[0,s_m-1]$ 代入式(5)得到离散形式的 Chirp 原子为 $h_{\gamma_m'}(k)$。则离散 Chirp 原子库 $D_{chirp}=\left\{h_{\gamma_m'}(k)\right\}$，其中 $\gamma_m'\triangleq(s_m,f_m,k_m)$，$m\in[1,szDict]$。这一步即是构建 Chirp 原子库。

第 2 步，设定 Chirp 原子时间中心 u 的搜索步长即间隔为 $step\_u$。

第 m 个 Chirp 原子的搜索次数 $shift_m=\frac{N-s_m+1}{step\_u}$。即第 m 个 Chirp 原子可以往后延时 $u_n=n\cdot step\_u$ 个离散时间点，再与待分解信号做内积运算，其中 $n\in[0,shift_m-1]$。

第 3 步，设定循环索引 $i=1$，剩余信号 $R_i(k)=S(k)$，$k=0,1,\cdots,N-1$。

第 4 步，通过 Chirplet 对 $R_i(k)$ 进行分解，寻找该信号在所有 Chirp 原子上投影的最大值。

$$\{s_{mi},u_{ni},f_{mi},k_{mi}\}=\arg\max\left|<R_i(k),h_{\gamma_m'}(k-u_n)>\right|$$

设 $c_i=\max\left|<R_i(k),h_{\gamma_m'}(k-u_n)>\right|$。$u_n$ 表示 Chirp 原子所有可能的中心时间，其取值范围按照第 2 步中的规律来确定。其中，$<R(k),h(k)>$ 定义为：$<R(k),h(k)>=\sum_{k=n_0}^{n_1}R(k)h^*(k)$。

第 5 步，分解信号 $R_i(k)$，得到残值 $R_{i+1}(k)=R_i(k)-c_i h_{\gamma_{mi}'}(k-u_{ni})$。

第 6 步，若剩余能量 $\|R_{i+1}(k)\|_2<\varepsilon$，则停止分解，否则令 $i=i+1$，转至步骤 4。

设总共分解了 I 次，那么分解的结果就是得到了 I 组向量：$\gamma_m^* \triangleq (s_{mi}, u_{ni}, f_{mi}, k_{mi})$，其中 $i \in [1, I]$。

2.2 算法的 CUDA 实现

在 CUDA 编程模式中，CPU 是主机(host)，而 GPU 是设备(device)。

CUDA 是一个超多线程编程环境。CUDA 通过如下的层次结构来管理同时运行在 GPU 中的多个线程：由若干个线程构成线程块(Block)，若干个线程块构成线程网格(Grid)。组成线程块网格的线程块和组成线程块的线程分别通过线程块索引和线程索引进行访问。根据需要，线程索引的维数可以选择为一维、二维、三维的，线程块索引的维数可以选择为一维、二维的。包含在同一个线程块网格中的所有线程并发执行同一段 CUDA 子程序，即内核函数(kernel)，kernel 函数由主机调用并在设备上运行。

CUDA 线程可以在执行时间内访问多种存储器。文献[6]介绍了 CUDA 的存储器属性。各种不同存储器的访问速度和存储量是不同的。

在上述流程中，第 1 步构建庞大的、冗余的 Chirp 原子库和第 4 步从过完备 Chirp 库中检索出最佳匹配 Chirp 原子所需的内积运算的计算量非常大，其运行时间决定了整个原子分解所需的时间。但是各个采样值是相互独立的，每次的内积运算也是相互独立的。因此本文结合高效的 GPU 运算平台，在 GPU 上同时运行多个线程，对采样值的计算和内积运算进行并行处理，最大可能地提升效率。图 1 为使用 CUDA 实现的四参数 Chirplet 分解的处理流程。

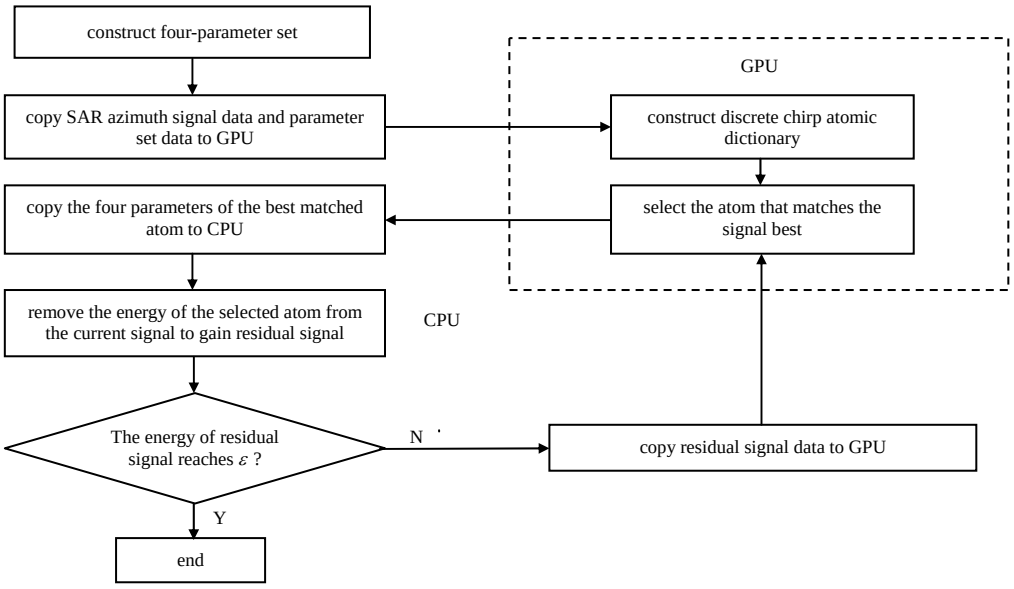

Fig.1 Block diagram of implementation of four-parameter Chirplet decomposition

图 1 四参数 Chirplet 分解处理流程

在构建 Chirp 原子库过程中，1 个 CUDA 线程负责 1 个采样点。由于各个 Chirp 原子的采样点个数不一定相同，取 $s_{\max} = \max(s_m), m \in [1, szDict]$，即 $s_{\max}$ 取为所有 s_m 中的最大值。Kernel 函数由 $szDict \times s_{\max}$ 个线程并发执行，同一行的线程负责计算 1 个 Chirp 原子的采样值，多余的线程进行空操作。线程块网格的大小是 $(szDict / b_x) \times (s_{\max} / b_y)$，其中 (b_x, b_y) 为 1 个线程块的尺寸。为了最大程度地发挥 GPU 的并行计算能力，需要恰当地确定线程块的尺寸。由于冗余原子库的采样值的数据量很大，因此把这些数据放到全局内存中，供后续使用。

在检索最佳原子的过程中，每次信号分解需要进行的内积运算总次数为 $shift_1 + \cdots + shift_m + \cdots + shift_{szDict}$，其中 $shift_m$ 是第 m 个 Chirp 原子和待分解信号需要做的内积运算次数。取 $shift_{\max} = \max(shift_m)$，$m \in [1, szDict]$。另外一个 kernel 函数负责内积运算，该 kernel 函数由 $szDict \times shift_{\max}$ 个线程并发执行。同一行的线程负责同一个 Chirp 原子所要做的内积运算，多余的线程空操作。线程块网格的大小是：$(szDict / d_x) \times (shift_{\max} / d_y)$，其中 (d_x, d_y) 为线程块的尺寸。同样，块的尺寸需要恰当的确定。内积运算得到的互相关系数的数据量也是非常大的，因此也把它放在 GPU 的全局内存中。注意到如果线程块中的每一个线程单独从全局内存中取出数据(包括待分解信号数据和 Chirp 原子数据)，将导致同一个数据被多个线程重复读出，这样的话大量的时间会浪费在读取低速的全局内存上。根据 CUDA 规范，同一个块中的所有线程都在同一个处理器上执行，因此它们可以通过共享内存共享数据。在共享内存中为线程块分配 2 个大小为线程块中线程个数的数组，用于存放待分解信号数据和 Chirp 原子数据，当线

程开始内积运算时，线程块中的每一个线程从全局内存中读出 1 个待分解信号数据和 1 个 Chirp 信号数据存入到共享内存中，这样的话，这些数据可以被线程块中的所有线程共享，线程块中的线程根据这些数据进行计算并叠加。当线程块中的所有线程完成计算以后，重复上述步骤，把后续的数据读取到共享内存中，并且对它们的计算所得结果进行叠加，直到处理完所有的待分解信号数据和 Chirp 信号数据。这样可以大大减少全局内存的访问次数，节约了时间。至此已经完成了内积运算，并且所有的互相关系数数据已经存入到全局内存中，可供后续使用。

由于 GPU 使用 PCI-E 总线与主机通信，输入输出受到 IO 带宽的限制，所以要尽量减少设备与主机之间传输的数据量。如果把上述庞大的互相关系数数据传输到主机，再检索出其中的最大值是不合适的。基于这一点，使用另外一个 kernel 函数检索出所有的互相关系数中的最大值及其所对应四参数 $(s_{mi}, u_{ni}, f_{mi}, k_{mi})$，只需要把此四参数传回到主机即可。

3　实验结果与效率测试

为了检验基于 GPU 的 SAR 方位向信号分解方法的性能，实验中分别实现了运行在单个 GPU 上和运行在 4 个 GPU 上的分解方法，并且与传统的基于 CPU 的分解算法进行效率比较。表 1 列出了各种不同条件下在 GPU 和 CPU 平台上的运行时间，其中在计算 GPU 上实现的时间时考虑了读写显存的时间以及 CUDA 初始化的时间，精度是 double 型的。测试使用的硬件平台分别是:CPU,intel Xeon E554,主频 2.53 GHz;GPU,NVIDIA Tesla C1060，处理器内核频率 1.296 GHz。

设原始 SAR 方位向信号总能量为 E。本文进行了 4 组测试。在表 1 中,条件 1 是:Chirp 原子库 $D_{\text{chirp}}=\{h_{\gamma_m'}(k)\}$，原子数目为 32 768，中心时刻延迟 u 的搜索步长为 25，能量阈值 $\varepsilon=50\%E$；条件 2 是：Chirp 原子 D_{chirp} 原子数目为 65 536，中心时刻延迟 u 的搜索步长为 25，能量阈值 $\varepsilon=50\%E$；条件 3 是：Chirp 原子库 D_{chirp} 原子数目为 16 384，中心时刻延迟 u 的搜索步长为 50，能量阈值 $\varepsilon=50\%E$；条件 4 是：Chirp 原子库 D_{chirp} 原子数目为 16 384，中心时刻延迟 u 的搜索步长为 25，能量阈值 $\varepsilon=50\%E$。由表 1 可以看出，基于 GPU 的分解方法相比基于 CPU 的分解方法具有非常明显的速度提升，分解的时间得到了极大的缩短。

表 1 基于 GPU 与基于 CPU 的分解时间比较(单位：s)

Table1 Comparison of decomposing time based on GPU and CPU(unit: s)

condition	1	2	3	4
1 CPU	5 520.803 016	9 620.458 65	1 421.595 820	2583.817 883
4 CPU	1 468.994 219	2 584.000 711	385.941 772	676.946 231
1 GPU	314.680 056	544 .238 534	82.171 705	147.749 583
4 GPU	91.113 33	149.328 900	21.217 660	39.195 561

为了验证基于 GPU 的 SAR 方位向信号分解的有效性，按照上述各种条件，将基于 GPU 实现的残余信号的能量下降情况绘成图，如图 2 所示，对应的横坐标是分解循环的次数 c，纵坐标是经过 c 次循环后残余信号的能量 e，并且与基于 CPU 实现的残余信号的能量曲线进行比较。如图所示，基于 GPU 实现的分解算法和基于 CPU 实现的分解算法所得到的能量下降曲线完全相同，即基于 GPU 的 SAR 方位向信号分解方法是完全有效的。

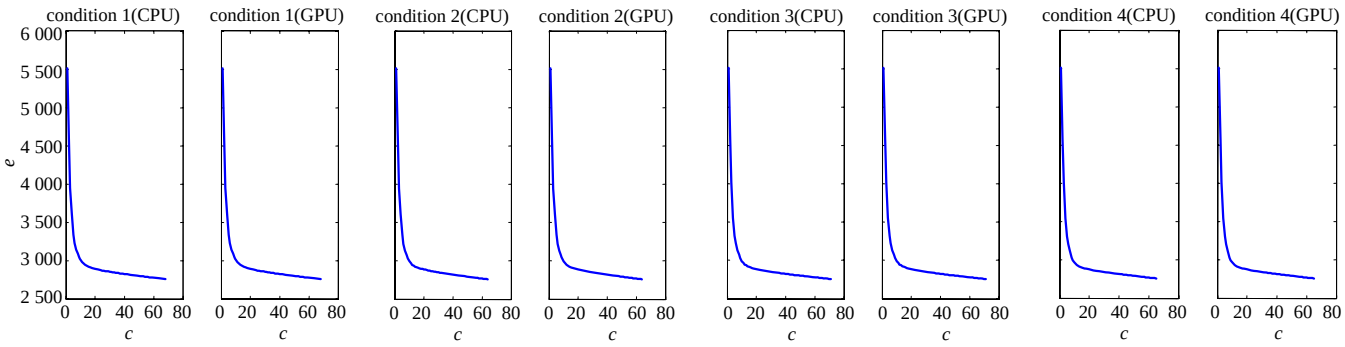

Fig.2 Comparison of changing of the residual signal's energy based on CPU and GPU

图 2 基于 CPU 和 GPU 的残余信号能量变化比较

4　结论

本文提出的基于 GPU 的 SAR 方位向信号分解方法引入了 CUDA 这种并行计算架构，利用 GPU 并行计算引

擎来解决复杂计算问题[7]，极大地发挥 GPU 的通用计算潜能。该方法与传统的基于 CPU 的分解方法相比，耗费更少的资金和电力[8]。本文专注于 SAR 方位向信号的分解。今后的研究将致力于把该算法融入到整个 SAR 成像系统中，把 CUDA 并行架构应用在 SAR 成像系统的更广泛的领域中，获得新一代的 SAR 成像算法系统，以得到实时的 SAR 成像。

参考文献：

[1] Macedonia M. The GPU enters computing's mainstream[J]. IEEE Computer, 2003,36(10):106–108.

[2] NVIDIA. CUDA programming guide version 3.0[M/OL]. [S.l.]:NVIDIA Corp, 2009. [2011-06-28]. http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf.

[3] Stéphanc G Mallat,Zhang Zhifeng. Matching Pursuit With Time–Frequency Dictionaries[J]. IEEE Transactions on Signal Processing, 1993,41(12):3377–3415.

[4] Gribonval R. Fast Matching Pursuit with a Multiscale Dictionary of Gaussian Chirps[J]. IEEE Trans. Signal Processing, 2001,49(5):994–1001.

[5] Yin Qinye,Qian Shie,Feng Aigang. A Fast Refinement for Adaptive Gaussian Chirplet Decomposition[J]. IEEE Trans. Signal Processing, 2002,50(6):1298–1306.

[6] Svetlin A,Manavski,Giorgio Valle. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment[J]. BMCBioinformatics, 2008,9(z2):S10.

[7] 俞惊雷,柳彬,王开志,等. 一种基于 GPU 的高效合成孔径雷达信号处理器平[J]. 信息与电子工程, 2010,8(4):415–419. (YU Jinglei,IU Bin,WANG Kaizhi,et al. A highly efficient GPU-based signal processor of Synthetic Aperture Radar[J], 2010,8(4):415–419.)

[8] Govindaraju Naga,Gray Jim,Kumar Ritesh,et al. GPUTeraSort:high performance graphics co-processor sorting for Large database management[C]// International Conference on Management of Data. Chicago:[s.n.], 2006:325–336.

作者简介：

张相广(1984–)，男，浙江台州人，硕士，主要研究方向为 GPU 高性能运算和信号时频分解.email:strong0109@yahoo.cn.

吴长朋(1986–)，男，山东菏泽人，硕士，主要研究方向为图像解译、高分 SAR 图像特征提取.

高叶盛(1983–)，男，浙江绍兴人，博士，主要研究方向为信号时频分析、SAR 精确成像.

郁文贤(1964–)，男，上海市人，博士，教授，主要研究方向为低分辨率雷达目标波形动态识别理论与系统技术、高分辨率雷达图像目标识别理论与技术.

王开志(1977–)，男，四川绵阳人，博士，副教授，主要研究方向为微波成像、微波全息图像、信号的时频分析与分数阶算子理论.

(上接第 92 页)

[9] 陶荣辉,陈惠连. 一种雷达脉冲的去交错和识别新算法[J]. 信息与电子工程, 2005,3(1):18–21. (TAO Ronghui,CHEN Huilian. A new algorithm for radar pulse deinterleaving and recognition[J]. Information and Electronic Engineering, 2005,3(1):18–21.)

作者简介：

何艾玲(1985–)，女，四川省彭山县人，硕士，研究实习员，主要从事雷达对抗信号处理技术研究. email:aileensum@foxmail.com.

蔡英武(1971–)，男，四川省乐山市人，硕士，研究员，研究方向为雷达信号处理、电子对抗总体设计研究.

陶荣辉(1969–)，男，四川省达州市人，硕士，副研究员，主要研究方向为电子对抗中的信号处理.

李合生(1970–)，男，四川省巴中市人，博士后，研究员，主要研究方向为信号与信息处理、电子对抗和模式识别等.