第 38 卷 第 4 期
2020 年 12 月

天 文 学 进 展
PROGRESS IN ASTRONOMY

Vol. 38, No. 4
Dec., 2020

doi: 10.3969/j.issn.1000-8349.2020.04.05

混合 w-facets 成像并行算法研究

于 昂[1]，劳保强[2]，王俊义[1]，安 涛[2]

(1. 桂林电子科技大学 信息与通信学院，桂林 541004； 2. 中国科学院 上海天文台，上海 200030)

摘要： 大视场成像是低频射电干涉阵列数据处理的核心技术，也是实现未来平方公里阵射电望远镜 (square kilometre array, SKA) 宏伟科学目标的基础手段。为了改善 uv-faceting 和 w-projection 这两种大视场成像算法，研究了它们之间的混合算法 (称为 w-facets)；此外，为了加速该混合算法，提出基于多核 CPU 和 GPU 的并行算法。验证性实验表明，与 uv-faceting 算法相比，该混合算法每束光能降低 4 mJy 噪声水平，图像的动态范围最高提升 2.34 dB；并且，在图像质量最好时，与目前在澳大利亚默奇森宽视场阵列 (Murchison widefield array, MWA) 项目数据处理中广泛使用的 w-stacking 算法的结果基本一致。性能测试结果表明，基于多核 CPU 的并行算法在一定线程数范围内，具有良好的可拓展性，当分面数与线程数相等时，加速效果最佳；基于 GPU 的并行算法加速比高达 201.8 倍，是多核 CPU 的并行方法最佳结果的 8.9 倍左右。这些研究成果能够为即将开展的大视场成像相关科学任务，提供有力的技术支撑和参考价值。

关 键 词： 射电天文；大场成像；混合算法；并行优化

中图分类号： P164 **文献标识码：** A

1 引 言

为了更深入地探索宇宙奥秘，射电望远镜朝着更高灵敏度、更高分辨率和更快巡天速度的方向发展。预计于 2021 年开建的平方公里阵 (square kilometre array, SKA) 射电望远镜，建成后将成为人类有史以来最大的综合孔径射电望远镜。该观测设备将为宇宙演化、引力的本质、生命起源、地外文明等重大前沿科学研究提供强有力的数据[1]。全数字化的 SKA 望远镜，规模巨大，将产生空前庞大的数据。SKA 第一阶段 (SKA1) 每年将约有 600 PB 的数

收稿日期： 2020-04-20； **修回日期：** 2020-07-07

资助项目： 国家重点研发计划 (2018YFA0404603)；国家自然科学基金 (U1831204, 10973028, 10833005, 10878003, 61966007)；创新群体 (10821302)；973 (2007CB815402/403)；重点实验室开发基金 (10821302, CRKL180201)；广西高校卫星导航与位置感知重点实验室开发基金；广西云计算与大数据协同创新中心 (1716)；认知无线电与信息处理教育部重点实验室主任基金 (CRKL180106)；广西无线宽带通信与信号处理重点实验室主任基金 (CRKL180106, GXKL06180107)

通讯作者： 劳保强，lbq@shao.ac.cn

据产品传输到 SKA 区域中心，这些数据将有助于天文学家进行深度处理与分析[2]。最终用于发表的 SKA 数据产品需要经过无数次的反复处理与验证，因此 SKA 数据处理软件与算法的运行速度和有效性将大大影响天文学家的工作效率，甚至影响科学产出。

大视场成像是 SKA1 低频阵列 (SKA1-low) 数据处理的重要环节，如 SKA1 的首要科学任务宇宙黎明的探索，将利用 SKA1-low 进行宇宙再电离成像观测；SKA1 十大科学目标之一宇宙磁场，也将利用 SKA1-low 进行大规模偏振成像观测。目前，应用比较广泛的大视场成像算法有 faceting[3–5], w-projection[6], w-stacking[7], snapshots[8] 和 w-snapshot[9]。faceting 和 snapshots 主要是利用数据分割技术，使得分割后的数据满足二维傅里叶变换近似成像方法。这种分割技术的成像方法由于需要循环处理多份分割数据，将消耗较多的计算时间和 CPU 资源。其中，faceting 又分为图像域 faceting 和 uv-faceting，图像域 faceting 需要对分割后的数据图像进行拼接，将会有图像边缘效应；snapshots 最后也需要进行图像拼接，同样存在类似的问题。uv-faceting 由于分割后的数据都在同一个网格上栅格化，经过二维逆傅里叶变换后将获得整个视场的图像，不存在边缘效应。

w-projection, w-stacking 和 w-snapshot 这三种方法主要是通过修正 w 项来完成大视场成像，相比分割技术的方法，成像质量相对高一些。w-projection 主要通过可见度数据与 w 核函数的卷积，将三维可见度数据投影到 $w = 0$ 平面，最后利用二维傅里叶变换进行成像。由于数据处理时，卷积核函数将存放在内存中，w-projection 的卷积核是三维的 w 核函数，因此相比传统的二维卷积核将要消耗更多的内存。w-stacking 可以说是 w-projection 的升级版，该算法在图像域进行 w 项校正，并使用多线程技术优化算法，减少了卷积核的内存使用，但是需要使用大量的内存存储 w 层的傅里叶变换数据。由于使用多线程加速，运行速度相比 w-projection 有很大的改善。w-stacking 算法已经封装在 wsclean 软件中，主要用于 SKA 先导望远镜默奇森宽视场阵列 (Murchison widefield array, MWA)[10] 的数据处理。由于 w 核可以单独计算，并根据内存来选择任意 w 平面数，使得 w-projection 比较容易与其他算法进行结合使用。w-snapshot 算法即是 snapshots 和 w-projection 结合的产物，结合后的算法改善了 snapshots 边缘的畸变问题，相比 w-projection 有更好的可拓展性。w-snapshot 主要封装在 ASKAPsoft 软件中，用于 SKA 先导望远镜澳大利亚平方公里阵 (Australian square kilometre array pathfinder, ASKAP)[11]的数据处理。

由于混合算法具备一定的优势，本文主要研究 uv-faceting 和 w-projection 的混合算法，助力 SKA1 科学数据处理。本文的混合算法，一方面，可以改善 uv-faceting 成像质量问题，包括噪声水平和动态范围等；另一方面，可以减少内存的使用，提高 w-projection 的可拓展性。为了提高该混合算法的运行速度，本文提出了两种并行优化方法：基于多核 CPU 的并行算法和基于 GPU 的并行算法。

2 混合 w-facets 成像算法

本文的混合成像算法，主要是将 uv-faceting 算法[4, 5]和 w-projection 算法[12] 结合以完

成低频干涉阵列大视场成像，称之为混合 w-facets 成像算法，其中 facets 表示分面。该算法首先采用 uv-faceting 中的视场和数据分割技术获得分面数据；然后通过相位旋转技术获得每个分面的可见度数据，通过基线旋转技术获得每个分面的 u 和 v 坐标数据。其次，对每个分面的数据进行栅格化，栅格化的卷积核函数不再使用传统的卷积核函数，而是 w-projection 算法中的 w 核函数。每个分面的栅格化处理均在同一个网格中进行，因此栅格化后的数据进行逆傅里叶变换，只输出一个天空图像，即一个图像结果，也称为脏图。相比图像域 faceting 算法，本文的方法无需进行图像拼接，避免了图像拼接引起的图像边缘畸变，同时也减少了运行时间。此外，与 w-projection 结合后再与只使用 uv-faceting 相比，本文算法的成图质量和图像的动态范围将会进一步提高。

近年，Wortmann[13, 14]提出的 w-towers 算法整体思想上与本文的 w-facets 算法类似，但具体设计思路不同。主要不同为：(1) w-towers 的 facets 分割方法与图像域 faceting 方法相同，而 w-facets 是基于 uvw 域分割的。(2) w-facets 方法中每个 facet 的数据在同一个大网格上进行栅格化，子网格实际上是按照 facets 数目进行严格均分而获得；而 w-towers 需要经过更复杂的计算，且每个子网格是由所有 facets 的数据构成的。(3) 本文采用的是 w-projection 方法进行混合，w-towers 采用的是 w-stacking 方法。下面详细介绍本文算法的公式推导过程。

在综合孔径成像原理中，经过射电干涉仪测量获得的可见度数据 $V(u,v,w)$ 与天空亮度分布函数 $I(l,m)$ (即天空图像) 存在如下关系[15]：

$$V(u,v,w)=\iint \frac{I(l,m)}{\sqrt{1-l^2-m^2}}\exp\{-2\pi\mathrm{i}[ul+vm-w(1-\sqrt{1-l^2-m^2})]\}\mathrm{d}l\mathrm{d}m \quad , \tag{1}$$

其中，u,v,w 是射电望远镜的基线在 $u-v-w$ 坐标系下的投影坐标，(l,m) 为 u 轴和 v 轴观测方向上的单位向量的方向余弦，i 是虚数单位。

在传统的小视场观测下，$2\pi\mathrm{i}w(1-\sqrt{1-l^2-m^2})$ (w 项) 近似等于 $\pi\mathrm{i}w(l^2+m^2)$，由于 l 和 m 趋近于 0，w 项与 $(ul+vm)$ 相比可以忽略，即 w 项趋近于 0[16]。因此，天空亮度分布函数与可见度数据近似为傅里叶变换对关系。理论上，对二维的可见度数据 $V(u,v)$ 进行二维傅里叶变换，即可获得天空亮度分布函数。在低频干涉阵列的大视场观测下，由于非共面基线效应的存在，w 项远远大于 0，因此二维傅里叶变换近似求解天空图像的方法不再适用。

为了使用二维傅里叶变换近似方法，可以将视场切割为 N_{facets} 个相同大小的小视场 (也称为分面)，当 N_{facets} 满足下式条件，每个小视场将可以采用近似方法重建天空图像：

$$N_{\mathrm{facets}} \geqslant \frac{B_{\max}}{\lambda D^2} \quad , \tag{2}$$

其中，$B_{\max}$ 是低频射电干涉阵列台站两两之间距离 (基线) 的最长值，max 表示取最大值，λ 是参考频率对应的波长，D 是台站的直径范围。

每个小视场的可见度数据可以通过原始可见度数据进行相位旋转获得，令第 j 个分面的

相位中心坐标为 (l_j, m_j, n_j)，则该分面的可见度数据 V_j 为[4, 5]：

$$V_j = V(u,v,w)\exp\{2\pi\mathrm{i}[ul_j + vm_j - w(1-\sqrt{1-l_j^2-m_j^2})]\} = \iint \frac{I(l,m)}{\sqrt{1-l^2-m^2}}\exp\{-2\pi\mathrm{i}[u(\Delta l_j) + v(\Delta m_j) + w(\sqrt{1-l^2-m^2} - \sqrt{1-l_j^2-m_j^2})]\}\mathrm{d}l\mathrm{d}m \quad , \tag{3}$$

其中，$\Delta l_j = l - l_j$, $\Delta m_j = m - m_j$，(l_j, m_j, n_j) 可以利用分面相位中心的赤经 RA 和赤纬 DEC，以及原始相位中心的 RA 和 DEC，通过坐标转换公式计算出来[5]。

式 (3) 末尾的 2 个平方根项，经过泰勒一级近似展开，有：

$$\sqrt{1-l^2-m^2} - \sqrt{1-l_j^2-m_j^2} \approx \frac{\partial h}{\partial l}\Delta l_j + \frac{\partial h}{\partial m}\Delta m_j = -\frac{1}{\sqrt{1-(l_j^2+m_j^2)}} l_j\Delta l_j + m_j\Delta m_j \quad . \tag{4}$$

将式 (4) 代入式 (3) 后，得：

$$V_j = \iint I(\Delta l_j, \Delta m_j)\exp[-2\pi i(u_j\Delta l_j + v_j\Delta m_j)]\mathrm{d}l_j\mathrm{d}m_j \quad , \tag{5}$$

其中，u_j 和 v_j 是第 j 个小视场的基线坐标，计算公式为：

$$\begin{cases} u_j = u - w\dfrac{l_j}{n_j} \\ v_j = v - w\dfrac{m_j}{n_j} \end{cases} \quad . \tag{6}$$

随后，每个小视场的可见度数据和 (u_j, v_j) 数据在同一个网格上进行栅格化，再对整个网格数据进行二维逆傅里叶变换，即获得整个视场的脏图。其中，栅格化处理时，采用 w-projection 的 w 核函数与每个小视场的可见度数据进行卷积来完成。w 核函数的表达式为[17]：

$$C_{\mathrm{w-kernel}}(u,v,w) = C(u,v) \otimes \tilde{G}(u,v,w) \quad , \tag{7}$$

其中，$C(u,v)$ 是传统的卷积核函数，一般选取性能较优的球谐函数；$\otimes$ 表示卷积运算；$\tilde{G}(u,v,w)$ 是 $G(l,m,w)$ 的傅里叶变换，$G(l,m,w)$ 的表达式为[6]：

$$G(l,m,w) = \exp[-2\pi\mathrm{i}w(\sqrt{1-l^2-m^2}-1)] \quad . \tag{8}$$

从 w 核函数的表达式可以看出，栅格化的卷积将多一个维度，也就是这个 w 核函数将由多个二维卷积核组成。值得注意的是，每个二维卷积核的大小不固定，将会随着 w 值和视场的增大而增大[6]。通常，在 $w = 0$ 时，二维卷积核的大小为 7×7 或者 9×9[6]。由于一维卷积核函数是一种窗函数，所以一维卷积核函数的长度又称为全支撑大小。当 $w > 0$ 时，全支撑大小可以由 $0.5\pi w I_{\mathrm{width}}^2$ 计算得到，其中 I_{width} 是脏图的最长边大小或完整图像的最

长边大小，单位为 rad。从上述分析可知，如果 w 平面数和图像过大则消耗的内存较多；w 平面数取值过小，则将导致混叠效应产生。因此，w 平面数取值应合理，我们给出每个分面较佳的平面数计算公式：

$$N_{\text{wplanes}} = 0.5\frac{w_{\max}}{\lambda} \cdot \frac{I_{\text{width}}}{\sqrt{N_{\text{facets}}}} \quad , \tag{9}$$

其中，$w_{\max}$ 是 w 的最大取值。

uv-faceting 与 w-projection 结合后，N_{facets} 在不满足公式 (2) 的情况下也能完成大视场成像，因为当 $N_{\text{facets}} = 1$ 时，即进行 w-projection 成像。此外，N_{wplanes} 与需要处理的图像大小会随着 N_{facets} 的增加而减小，w 核函数的最大全支撑大小也随之减小，因此本文的混合算法将大大减少卷积核内存的消耗。

Algorithm 1 基于多核 CPU 的成像并行算法

Input: V 可见度数据，(u,v,w) 基线坐标，N_{wplanes} w 平面数，网格数组栅格存放栅格化结果全零数组 (完整图像大小)

Output: 脏图

```
#pragma omp parallel for schedule (static)
for j in N_facets do
    for r in row_count do
        for c in channel_count do
            V,(u, v, w),RA_0, DEC_0 = read_data(r,c);
            RA_j, DEC_j = calculate_facets_phase(RA_0, DEC_0, N_facets)
            V_j=phase_rotation(V, u, v, w,RA_j, DEC_j);
            u_new,v_new=baseline_rotation(u,v,w);
            u_scale, v_scale=scale_uv(u_new,v_new);
            w_index = w/w_max × (N_wplanes − 1)
            //w 核栅格化
            for sv in 0 to c_fullsupport[w_index] do
                for su in 0 to c_fullsupport[w_index] do
                    conv_weight = C_w−kernal[w_index][sv][su];
                    grid[v_scale + sv][u_scale + su] += V_j× conv_weight;
                end for
            end for
        end for
    end for
end for
dirty_image = IFFT(grid)
return dirty_image
```

3 并行算法

3.1 基于多核 CPU 的成像并行算法

混合 w-facets 成像算法的数据处理过程与传统的射电干涉成像基本相同，大体包括 4 个部分：数据读取，卷积核函数的创建，栅格化和逆傅里叶变换。其中，栅格化至少包含 5 层循环处理过程，相对其他过程，需要消耗更多的运行时间。

因此，本文提出了一种粗粒度并行处理方法，实现每个分面数据的并行处理。采用开放式多处理 (open multi-processing, OpenMP) 并行技术，在最顶层加入并行任务调度，即可实现该循环的并行。本文主要采用静态调度方式，具体并行过程：当分配 m 个 CPU 核处理 $N_{\rm facets}$ 个分面数据时，每个核 (或线程) 将处理 $N_{\rm facets}/m$ 个分面的数据。当分配的 CPU 核数与 $N_{\rm facets}$ 相等时，并行效果最佳。这种并行方法避免了并行化过程中可能出现的同步和锁的影响，是一种简单实用的并行方法。

基于多核 CPU 的混合 w-facets 成像并行算法的实现见 Algorithm 1 伪代码。算法的输入数据为校准后的数据，一般为宽带数据，有多个不同频率通道。最外的三层循环分别为：分面处理循环、按行处理循环和频率通道处理循环。内层的两个循环，主要完成栅格化处理；最外层，是根据 CPU 核数来进行分面处理的并行。最外三层循环，每一次将从输入数据中读取可见度、基线坐标 $u-v-w$ 和原始相位中心 $\rm RA_0$ 和 $\rm DEC_0$ 等主要数据；接着，根据分面数目 $N_{\rm facets}$ 和原始相位中心计算出当前分面的相位中心 $\rm RA_{\it j}$ 和 $\rm DEC_{\it j}$。将公式 (3) 进行相位旋转，得到当前分面的可见度数据 V_j；根据式 (6) 进行基线旋转，获得当前分面的基线坐标数据 $u_{\rm new}$ 和 $v_{\rm new}$，并利用傅里叶变换缩放原理对旋转后的基线数据进行缩放。然后，通过 w 的最大值 $w_{\rm max}$ 和 $N_{\rm wplanes}$ 计算出当前 w 在 w 核函数的索引位置 $w_{\rm index}$。

最后，通过二层循环完成栅格化，每次循环将通过 $w_{\rm index}$ 获取 w 核函数的值，再与当前可见度数据进行乘累加。$c_{\rm fullsupport}$ 是 w 核中一维卷积核函数全支撑大小数组，通过 $w_{\rm index}$ 获取当前 w 所需的全支撑大小。网格数组栅格的大小为完整图像的大小，经过逆傅里叶变换后将得到唯一脏图。

3.2 基于 GPU 的成像并行算法

GPU 是一种可以一次并行运行数千个线程来实现快速计算的处理器，其计算能力是 CPU 的百倍。统一计算设备架构 (compute unified device architecture, CUDA) 的内部主要框架是以 GPU 中的线程块为组进行划分，每块由一个多核处理器控制，这意味着只需要控制多核处理器就可以同时控制和处理上万个线程。本文将利用 CUDA 编程实现基于 GPU 的混合 w-facets 成像并行方法。该方法主要将输入数据拷贝到 GPU 设备中, 然后在 GPU 中进行并行的栅格化处理。该方法让主机负责整个系统的内存管理、逻辑控制以及图像形成，而让 GPU 负责大量的并行计算，包括卷积核函数的计算，栅格化过程的处理等，分工明确。

在 GPU 中进行栅格化处理时，使用原理为：同一基线连续测量的任意两个数据之间的时间间隔足够短，相同基线测量的数据集经过栅格化处理后将落在相同的网格点上。同时借

鉴 Romein[18] 的研究成果，将栅格化分成与卷积核函数相同大小的子网格，线程被分配用来处理每个子网格中的可见度数据，并依靠基线轨迹，将数据进行累加并存储到 GPU 寄存器中。当基线轨迹移动到下一个网格点时，将当前网格点内的值进行累加取均值，并在处理下一个网格点之前，将当前网格点的处理结果从寄存器写入全局内存中[19]，这样能减少 GPU 内存的使用。

该方法利用基线进行并行化，将同一基线不同时刻的观测数据作为一组，每个线程负责处理一组数据。本文的加速方法所需的最小总线程数为：$cs_{\max}\times cs_{\max}\times N_{\text{baselines}}\times N_{\text{facets}}$，其中 $cs_{\max}$ 是 $c_{\text{fullsupport}}$ 数组中的最大值，$N_{\text{baselines}}$ 为基线个数。此外，当卷积核全支撑大小较小时，每个线程将处理 1 个分面中 1 个基线的 1 次栅格化循环 (即 1 个卷积点)，将最大限度地进行并行化，循环次数减少到只有 2 层循环；当卷积核全支撑大小较大时，每个线程将处理 1 个分面中 1 个基线的多个卷积点，因为每个线程块的最大线程数目通常在 256 ∼ 1 024 之间。相对文献 [12]，本文增加了分面的并行处理。为避免两个或多个线程发生竞争现象，我们对全局内存的写入方式设置为原子性。

由于实际观测的数据通常是按照时间顺序记录在表格文件里的，同一时刻包含不同基线的数据，为了进行基线并行化，必须对输入数据进行重新排序。首先将输入的数据按照不同基线分组并组成一排，然后对相同基线即一组内的数据集按照观测时间先后排序。对每组基线进行编号，该编号是 0 至 $N_{\text{baselines}}-1$ 的自然数。天线在数据文件中有唯一编号，因此可以通过下式计算出基线的编号 $i_{\boldsymbol{B}}$[12]：

$$i_{\boldsymbol{B}}=\left(-i_{\min}^{2}+2i_{\min}N_{\text{ant}}+i_{\min}\right)/2+\left|i_{\text{ant1}}-i_{\text{ant2}}\right| \quad , \tag{10}$$

其中，i_{ant1} 和 i_{ant2} 分别是该基线中天线 1 和天线 2 的编号，N_{ant} 天线个数，$i_{\min}$ 是天线 1 和天线 2 的编号最小值。

由于输入的数据一般已经进行过射频干扰的去除和校准处理，处理之后有些数据将会被删除，每组基线内的数据长度或时间步数 (时间戳数目 N_{t}) 将会不同。因此，每组基线主要通过该组数据开始索引和时间戳数目循环 (遍历) 处理完该组的数据[12]。基线的开始索引可以通过下式计算得到[12]：

$$\boldsymbol{B}_{\text{start\_index}}[i_{\boldsymbol{B}}]=\begin{cases}0 & i_{\boldsymbol{B}}=0\\ \displaystyle\sum_{k=0}^{i_{\boldsymbol{B}}-1}T[k] & i_{\boldsymbol{B}}>0\end{cases} \quad , \tag{11}$$

其中，T 是由所有组基线时间戳数目构成的数组，数组的索引与每组基线的编号一致。

图 1 给出基于 GPU 的混合 w-facets 成像并行算法流程。图中主要分为主机处理、GPU 处理和具体模块处理三个部分来描述。在单个 GPU 节点中算法优化的过程为：首先，主机读取数据集；然后主机与 GPU 设备进行通讯，完成 GPU 设备的初始化工作，包括选取性能最好的 GPU 卡和数据处理所需的内存的分配等；接着，将存储在主机内存中的数据进行重新排序，并将重排后的数据发送到已分配好的 GPU 内存中。其次，在 GPU 中以基线并行的方式进行数据处理，计算并行参数：根据线程索引计算出该线程所需要处理的基线起

始索引 $i_{\boldsymbol{B}}$、栅格化的卷积核全支撑索引 (su 和 sv) 和分面索引 (j)。具体的计算方法，见式 (12)。然后，利用计算好的索引，按照频率通道 (c) 和时间戳数目 ($N_{\rm t}$) 两层循环，加载该线程所需要处理的数据，以及获取 w 核函数，并进行乘累加，从而完成基线的栅格化处理。循环结束后，即结束该线程的处理任务。最后，将栅格化后的数据返回到主机内存中，并对该数据进行逆傅里叶变换 (IFFT) 从而得到脏图。

$$\begin{cases} sv = th_{id}\%cs_w^2/cs_w \\ su = th_{id}\%cs_w^2/cs_w \\ j = th_{id}/cs_w^2/N_{\rm baselines} \\ i_{\boldsymbol{B}} = th_{id}\%cs_w^2\%N_{\rm baselines} \end{cases}, \tag{12}$$

其中，cs_w 是 w 值对应的卷积核全支撑大小，th_{id} 是线程索引。

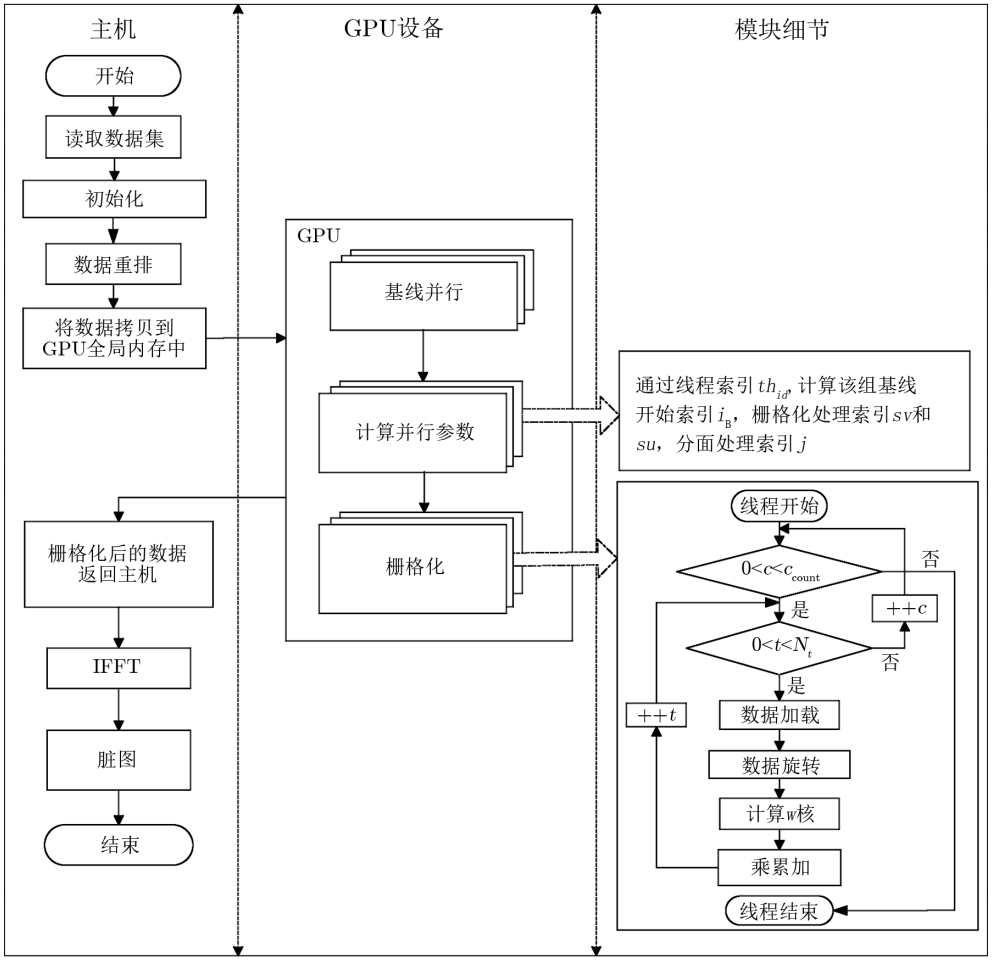


图 1 基于 GPU 的混合 w-facets 成像并行算法流程

4　实验与分析

4.1　算法验证

本节采用的数据来自 SKA 先导望远镜 MWA 的银河系与河外星系全天巡天 (galactic and extragalactic all-sky MWA survey, GLEAM) 项目中于 2013 年 8 月 9 日 22:13:18 开始并历时 112 s 的快照观测数据，该快照观测的相位中心为 (RA=03h13min54.40s, $DEC = -55°04'02.82''$)，观测频率范围是 200 ∼ 231 MHz。首先我们对该观测数据进行 RFI 消除、频率与时间积分并转换为 Measurement Set 格式，然后进行校准处理，处理完后获得的数据将用于本文算法进行成像。

使用的测试环境是上海天文台 SKA 数据中心原型机的 GPU 节点，该节点具体的硬件指标参数如表 1 所示。

表 1　上海天文台 SKA 数据中心原理样机的 GPU 节点指标参数

类型	数量	型号	总计
CPU	2	Intel(R) Xeon(R) Gold 6140, 2.30 GHz	36 核心
内存	16	DDR4-32G-2666-ECC	512 GB
硬盘	4	1.92 TB-SSD-2.5	7.68 TB
GPU 卡	8	Nvidia Tesla V100 SXM2 32 GB	80 TFLOPS (双精度)

根据 GLEAM 巡天数据处理的经验，我们设置需要计算的脏图的大小为：$4\,000 \times 4\,000$，每个像素的大小为 $25''$，因此脏图的最大宽度 I_{width} 约为 0.48 rad。该快照数据的 $w_{\max}$ 值约为 1 035 倍波长，根据式 (9) 我们计算出单独使用 w-projection 时，最佳的 N_{wplanes} 约为 248。如表 2 所示，是不同成像算法的测试结果对比。在成像质量对比方面，我们主要给出的是均方根 (RMS)、动态范围 (DR) 和射电源识别数目。通过从这三方面的对比中发现，二维傅里叶变换方法 (2D FFT，保持 $N_{\mathrm{facets}} = 0$ 和 $N_{\mathrm{wplanes}} = 0$) 成像质量最差，w-stacking 方法成像质量最好。uv-faceting 方法 (保持 $N_{\mathrm{wplanes}} = 0$) 的成像质量比 2D FFT 好，随着 N_{facets} 的增大成像质量越好。w-projection 方法 (保持 $N_{\mathrm{facets}} = 0$) 在 $N_{\mathrm{wplanes}} \geqslant 31$ 时，成像质量均比 2D FFT 方法好；在较小 N_{wplanes} 时，w-projection 比 uv-faceting 的成像质量要好，成像质量也随着 N_{wplanes} 的增大质量越高。对于 w-facets 方法的成像结果，在 $N_{\mathrm{facets}} = 16$ 和 $N_{\mathrm{wplanes}} = 62$ 时质量最高；在相同的取值时，比单独使用 uv-faceting 或 w-projection 的质量都要高，并且最接近 w-stacking 的成像质量。此时，w-facets 的图像动态范围比 uv-faceting 提高了 2.34 dB，图像 RMS 降低了 4 mJy·beam^{-1}。在 $N_{\mathrm{facets}} = 4$ 和 $N_{\mathrm{wplanes}} = 124$ 时，w-facets 的成像结果比相同取值的 uv-faceting 成像质量好，比 w-projection 的差；在 $N_{\mathrm{facets}} = 64$ 和 $N_{\mathrm{wplanes}} = 31$ 时，w-facets 的成像结果比 w-projection 的都要好，与 uv-faceting 成像结果相近。成像质量的分析表明，uv-faceting 与 w-projection 结合后，成像质量能够得到进一步提高。

在内存使用方面，我们记录了不同成像方法的卷积核内存使用情况或傅里叶变换 (FFT) 内存使用情况。w-stacking 统计的是 FFT 内存使用量，其余方法均是统计卷积核内存使用情况，因为 w-stacking 的 w 项校正在图像域进行，卷积核消耗的内存小 (小于 1 MB)。由表 2 可知，2D FFT 和 uv-faceting 的卷积核消耗的内存均小于 1 MB，w-projection 方法的卷积核内存消耗随着 $N_{\rm wplanes}$ 的增大而增大，且远远大于 1 MB；w-facets 方法的卷积核内存消耗随着 $N_{\rm facets}$ 的增大而减少，当 $N_{\rm facets}=64$ 时，内存消耗小于 1 MB；w-stacking 方法由于使用了较大的 w 层，FFT 内存消耗较大，高达 48 GB。分析表明，将 w-projection 与 uv-faceting 结合后，内存的消耗大大减少。

对于运行时间，我们给出的是使用 20 个 CPU 核的运行时间和使用 1 个 GPU 卡的运行时间，所有运行时间均是测量 10 次取平均值。由于基于多核 CPU 的并行是针对分面，因此仅使用 w-projection 方法时，程序是串行执行的，比 uv-faceting 方法消耗更多的时间。由于使用 20 个 CPU 核，uv-faceting 方法在 $N_{\rm facets}=16$ 时能够充分并行，所以消耗的时间都比其他的分面数少。w-facets 方法在 $N_{\rm facets}=16$ 和 $N_{\rm wplanes}=62$ 时，耗时最短，与相同 w 平面数的 w-projection 方法耗时相近，但耗时比 uv-faceting 方法均长。上述分析表明，uv-faceting 结合了 w-projection 之后，运行时间会变长。使用了 GPU 加速后，大多数方法的运行时间整体大大缩减了；uv-faceting 和 w-facets 的运行时间依然随着 $N_{\rm facets}$ 的增大而变长；w-projection 方法随着 $N_{\rm wplanes}$ 增大，运行时间无较大变化，因为 GPU 的加速方法主要是针对栅格化进行并行，因此分面数越大需要的处理时间越多。w-facets 方法中 GPU 的运行速度比 20 个 CPU 核快 7 ∼ 33 倍。

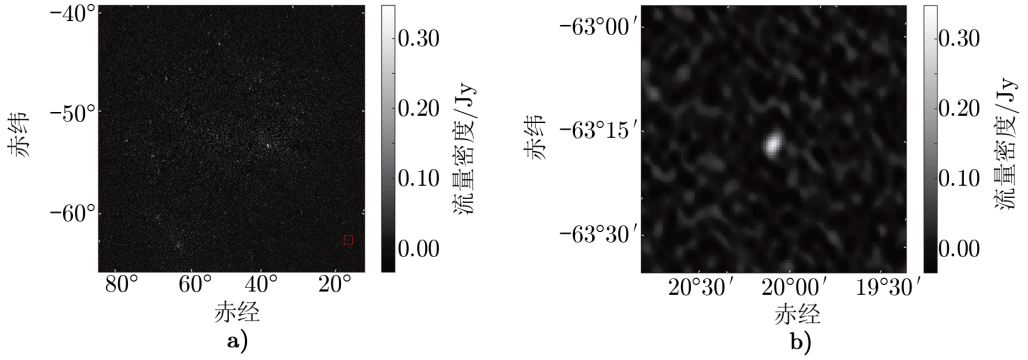
表 2　测试结果对比

成像设置	RMS /mJy	DR	射电源识别数	卷积核内存或 FFT 内存	运行时间 (CPU/GPU)/s
2D FFT	42	345.03	800	< 1MB	173.98/23.99
uv-faceting(4 facets)	41	331.31	838	< 1MB	192.46/29.13
uv-faceting(16 facets)	40	368.93	930	< 1MB	189.05/72.03
uv-faceting(64 facets)	36	439.33	1 091	< 1MB	699.37/251.19
w-projection(31 wplanes)	36	384.02	878	226 MB	597.23/23.32
w-projection(62 wplanes)	35	419.85	923	453 MB	735.39/24.18
w-projection(124 wplanes)	35	432.72	993	960 MB	897.19/24.23
w-projection(248 wplanes)	35	439.07	1 057	1813 MB	1 068.52/24.31
w-facets(4 facets,124 wplanes)	40	357.28	904	91 MB	976.46 /28.97
w-facets(16 facets,62 wplanes)	36	467.99	1 151	4 MB	763.03/71.56
w-facets(64 facets,31 wplanes)	37	391.49	1 018	< 1MB	1 767.21/251.26
w-stacking(376 wlayer)	33	468.32	1185	48 GB	138.05/无

注：RMS 表示均方根值；DR 表示动态范围；FFT 表示快速傅里叶变换；CPU 表示使用 20 个 CPU 核的运行时间；GPU 表示使用 1 个 GPU 卡的运行时间。

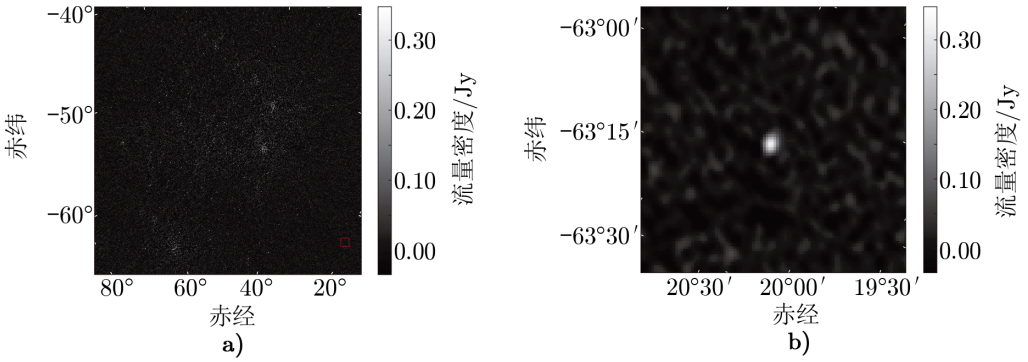
如图 2 和图 3 所示，分别为基于 GPU 的混合算法的成像结果和使用 w-stacking 算法的成像结果。为了便于比较，我们截取出距离原始图像较远位置的射电源，并得到如图

2b) 中和图 3b) 中的两个图像。图 2b) 图像的峰值为每束光 309.5 mJy，RMS 值为每束光 23.7 mJy。图 3b) 图像的峰值为每束光 311.9 mJy，RMS 值为每束光 21.8 mJy。结合表 2 与整个图像的对比，表明本文的 GPU 混合算法能够对距离较远的射电源成图，且成图结果与 w-stacking 获得的结果基本一致。基于多核 CPU 的混合算法结果与 GPU 混合算法结果相同，这里不作展示。



注：a) 是使用本文算法获得的完整图像；b) 是从 a) 中截取红色方框部分的图像。

图 2　基于 GPU 的混合 w-facets 成像并行算法的成像结果 (N_{facets}=16, N_{wplanes}=62)



注：a) 是使用该算法获得的完整图像；b) 是从 a) 中截取红色方框部分的图像。

图 3　w-stacking 成像结果

4.2　性能测试与分析

性能测试部分采用模拟数据进行，主要使用牛津大学开发的 SKA 射电望远镜模拟软件 (OSKAR) 仿真并生成 SKA1-low 观测数据。主要的参数设置为：天线采用 SKA1-low 第四阶段的完整规模天线阵列，台站个数为 512，基线个数为 130 816，最长基线为 65 km，观测相位中心为 (RA=201.36, DEC=43.02)，偏振个数为 4，波束个数为 1，观测的起始频率为 50 MHz，频率带宽为 300 MHz，频率通道数为 64，每个频率通道的带宽为 4 687.5 kHz；积分时间为 2 min，起始观测时间设置为 2015 年 01 月 01 日 18 点整，总观测时长为 2 min。观

测相位中心指向的天区包含一个巨大的射电星系 NGC 5128，也称为半人马座 A (Centaurus A)。它不仅是距离地球最近的射电星系 (距离大约为 11.09 光年)，而且是南半球最大的星系。本模拟观测使用的天空模型是由包含 Centaurus A 的 MWA GLEAM 巡天图像制作得到[20]。该模型共有 23 811 个独立成分，包含 23 336 个高斯源和 2 475 个点源。所有成分的流量强度和谱指数分布在 74 ∼ 231 MHz 频率范围内。最终模拟观测产生的数据量大小约为 36 GB。该数据曾经作为 SKA 科学数据处理器工作包中基于执行框架的大视场成像流水线的性能测试[21]，因此比较适合用于本文的性能测试。

实验设置的成像参数是：脏图大小为 1 024 × 1 024，每个像素大小为 14.06′，$N_{\rm facets} = 36$。该模拟数据的 $w_{\max} = 61\,026$ 倍波长，根据脏图的大小和分面数，我们计算出最佳的 $N_{\rm wpalanes} = 355$。该部分的实验采用的测试环境与验证性实验相同。如图 4 所示，是性能测试的结果，主要使用了不同 CPU 核数 (线程数) 测试基于多核 CPU 方法的可拓展性，使用一个 GPU 卡测试基于 GPU 方法的加速性能。图中每个测试点的运行时间均是连续测试 10 次取平均值。我们根据图 4 计算出加速比情况，如表 3 所示。根据多核 CPU 的方法，当使用的线程数为 36 时，线程数与 $N_{\rm facets}$ 值相等，因此消耗的运行时间最短；使用 1 个线程时，相当于串行程序，运行时间最长；线程数为 1 ∼ 12，消耗的运行时间近似线性减少；线程数大于 12 后，没有明显的线性关系，在线程数为 24 和 36 时，运行时间出现下降。使用 V100 GPU 时，单精度加速比为 201.8，运行速度是 36 个线程数的 8.9 倍左右；双精度加速比为 173.5，运行速度是 36 个线程数的 7.7 倍左右。上述分析表明，基于多核 CPU 的方法在少量线程数内有一定的可拓展性，当线程数与 $N_{\rm facets}$ 相等时，运行速度最快；基于 GPU 的方法在运行速度上比多核 CPU 方法有明显的优势。

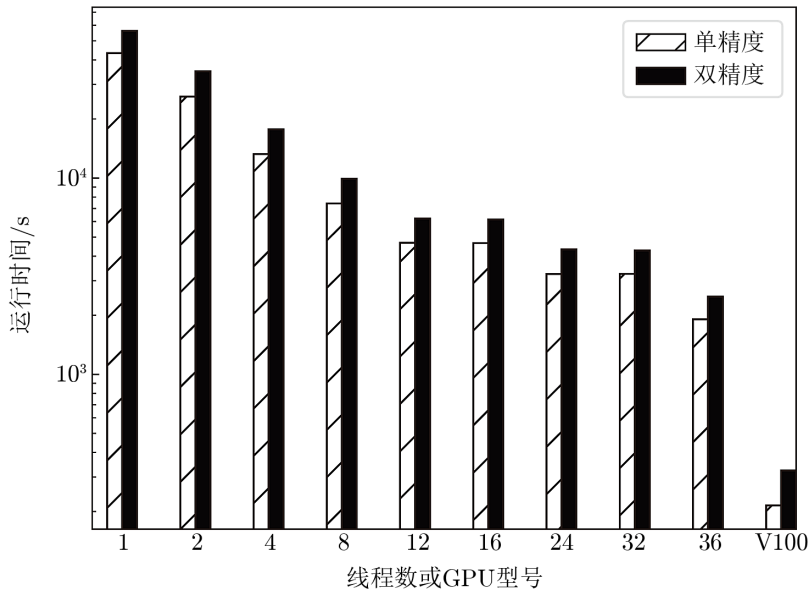


图 4 混合 w-facets 成像并行算法性能测试结果

表 3　加速比情况

CPU 线程数/GPU 型号	单精度加速比	双精度加速比
1	1	1
2	1.7	1.6
4	3.3	3.2
8	5.8	5.7
12	9.3	9.0
16	9.3	9.1
24	13.3	13.0
32	13.3	13.1
36	22.7	22.6
V100	201.8	173.5

5　总结与展望

由于低频射电望远镜阵列对成像具有高动态范围和高分辨率的要求，因此面对海量天文观测数据时，既须保证大视场成像的成图质量也必须具有较快的运行速度。故本文研究了 uv-faceting 与 w-projection 相混合的算法，称为 w-facets 算法，为将来开展低频射电望远镜阵列巡天观测的数据处理提供技术手段。与 uv-faceting 算法相比，w-facets 算法能够进一步减少噪声水平，提高成图的动态范围，改善成像质量；与 w-projection 算法相比，w-facets 算法大大减少了内存的消耗，提高了可拓展性。w-facets 在 uv-faceting 的基础上使用 w 核进行栅格化，运行时间比 uv-faceting 慢很多，但与 w-projection 的运行时间相近。为此，本文提出了两种并行优化方法：基于多核 CPU 的混合 w-facets 成像并行方法和基于 GPU 的混合 w-facets 成像并行方法。通过图像验证性实验和性能测试实验表明，本文提出的两种方法均能够正确实现混合 w-facets 算法成像，并提升混合算法的成像速度。基于多核 CPU 的并行加速方法运行速度最大提升约 22.7 倍，基于 GPU 的并行加速方法运行速度最大可以提升约 201.8 倍；这表明基于 GPU 的并行加速方法比基于多核 CPU 的并行加速方法更高效。但是通过本文测试分析可知，基于多核CPU的并行加速方法效果略差，因为采用的是粗粒度的并行方法，后续工作将考虑通过更细粒度的 CPU 并行方案进行优化。

参考文献：

[1] Bourke T, Braun R, Fender R, et al. Advancing Astrophysics with the Square Kilometre Array, AASKA14, 2015, 215: 174.
[2] Dewdney P E, et al. SKA1: Design Baseline Description, SKA-TEL-SKO-0001075, 2019
[3] Cornwell T J, Perley R A. A&A, 1992, 261: 353

[4] Kogan L, Greisen E W. AIPS Memo, 2009, 113: 1
[5] 劳保强, 安涛, 于昂, 等. 天文学报, 2019, 60: 12
[6] Cornwell T J, Golap K, Bhatnagar S. IEEE Journal of Selected Topics in Signal Processing, 2008, 2: 647
[7] Offringa A R, McKinley B, Hurley-Walker N, et al. MNRAS, 2014, 444: 606
[8] Sault R J, Staveley-Smith L, Brouw W N. A&A, 1996, 120: 375
[9] Cornwell T J, Voronkov M A, Humphreys B. International Society for Optics and Photonics, 2012, 8: 8500
[10] Wayth R B, Tingay S J, Trott C M, et al. PASA, 2018, 35: 33
[11] McConnell D, Allison J R, Bannister K, et al. PASA, 2016, 33: e042
[12] Lao B Q, An T, Yu A, et al. Science Bulletin, 2019, 64: 28
[13] Wortmann P. http://www.mrao.cam.ac.uk/ pw410/wtowers.pdf, 2017
[14] Wortmann P. http://www.mrao.cam.ac.uk/ pw410/distributed_imaging2.pdf, 2019
[15] Perley R A, Schwab F R, Bridle A H. Astronomical Society of the Pacific Conference Series, 1989, 6: 259
[16] Thompson A R, Moran J W, Swenson Jr G W, eds. Interferometry and Synthesis in Radio Astronomy, Cham Switzerland: Springer, 2017: 94
[17] Bannister K W, Cornwell T J. MNRAS, 2013, 430: 2390
[18] Romein J W. ACM International Conference Supercomputing, 2012: 321.
[19] Muscat D. master thesis, Republic of Malta: University of Malta, 2014: 19
[20] Hurley-Walker N, Callingham J R, Hancock P J, et al. MNRAS, 2017, 464: 1146
[21] Lao Baoqiang, An Tao, Wu Chen, et al. http://ska-sdp.org/sites/default/files/attachments/sdp_memo_78_scalability_testing_using_daliuge_v1.0_0.pdf, 2018

Research on Parallel Algorithms for Hybrid w-facets Imaging

YU Ang[1], LAO Bao-qiang[2], WANG Jun-yi[1], AN Tao[2]

(1. School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China; 2. Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030, China)

Abstract: Large field of view imaging is the core technology of low-frequency radio arrays data processing, and it is also the basic means to realize the grand scientific goals of the future square kilometre array (SKA) radio telescope. In order to improve the two large-field imaging algorithms uv-faceting and w-projection, a hybrid algorithm (called w-facets) between them is studied. In addition, in order to speed up the hybrid algorithm, a parallel algorithm based on multi-core CPU and GPU is proposed. Confirmatory experiments show that the hybrid algorithm can reduce the noise level by 4mJy at most compared to uv-faceting, and the dynamic range of the image is improved by 2.34 dB. Moreover, when the image quality of w-factes is the best, it is basically consistent with the results of the w-stacking algorithm which is widely used in the data processing of the Murchison Widefield Array (MWA) project in Australia. The performance test results show that the parallel algorithm based on multi-core CPU has good scalability within a certain number of CPU

threads. When the number of facets is equal to the number of threads, the acceleration effect is the best. The GPU-based parallel algorithm has an acceleration ratio of up to 201.8 times, which is about 8.9 times the best result of the parallel method of the multi-core CPU. These research results can provide strong technical support and reference value for future scientific tasks related to large-field imaging in SKA.

Key words: radio astronomy; wide-field imaging; hybrid algorithm; parallel optimization

. . . . . .

《天文学进展》 2021 年征订启事

《天文学进展》是天文学类中文核心期刊。刊物为季刊，2021 年本刊于 3、6、9、12 月中旬出版，每期定价 50 元，全年 200 元（含邮寄费和包装费）。凡需订阅 2021 年《天文学进展》，请到所在地邮局进行订购。

统一刊号：CN 31-1340/P

邮发代号：4-819

欢迎订阅！谢谢支持!

《天文学进展》编辑部

2020年12月