Accelerating Radio Astronomical Imaging

LIU Honghao LIU Yixin

June 28 2021

In this report, we discussed the derivation of imaging algorithm and notations in section I. In section II, the implementation and some details are discussed. Section III lists the possible improvements that can be done later.

1 Notation and Derivation

The sky images are generated by processing visibility received by antennas, see equation 1. U V and W are 3D coordinates in frequent domain and they are wavelengths.

$$u = \frac{f}{c} \cdot u' \qquad v = \frac{f}{c} \cdot v' \qquad w = \frac{f}{c} \cdot w'$$

u' v' and w' are original coordinates. u v and w are effective coordinates which multiply the frequency - f and divide speed of light - c.

$$I(l,m) = \frac{1}{n} \sum_{k=0}^{M} V(u_k, v_k, w_k) e^{i(u_k l + v_k m + w_k(n-1))} \tag{1}$$

l m and n are direction cosines of the point in the celestial coordinate. Based on the property of direction cosine, $l^2 + m^2 + n^2 = 1$. V indicates visibility in corresponding channels. M is the number of coordinates with different wavelengths received by antennas. I denotes the sky image.
The equation 1 can be rewritten as equation 2. And it can be regarded as 3D Fourier transform multiplying a scaling ratio. However, the fast Fourier transform can not be adopted due to the random distributed points. What is more, l and m should be mapped to pixels.

$$I(l,m) = \frac{1}{n} \sum_{u} \sum_{v} \sum_{w} V(u,v,w) e^{i(ul + vm + w(n-1))} \tag{2}$$

According to the convolution theorem, the Fourier transform of the convolution of two function equals the multiplication of Fourier transform of those two function, see equation 3.

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g) \tag{3}$$

Therefore, the visibility can be gridded into mesh coordinates by a convolution function (mask function). The equation 2 is transformed into equation 4.

$$I(l,m) = \frac{1}{n\hat{\phi}(a,b,c)} \sum_a \sum_b \sum_c G(a,b,c) e^{i(al+bm+c(n-1))} \tag{4}$$

$$G(a,b,c) = \sum_u \sum_v \sum_w V(u,v,w)\phi(a-u, b-v, c-w) \tag{5}$$

$\phi(x)$ is the mask function and we adopt the exponential semicircle function proposed by Barnett. [1] (β is a hyper-parameter). G is result after convolution between function V and ϕ. $\hat{\phi}$ is the Fourier transform of mask function ϕ.

$$\phi(x) = \begin{cases} e^{\beta(\sqrt{1-x^2}-1)} & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\phi}(a) = \mathcal{F}(\phi(x)) = \int_x \phi(x) e^{iax} dx$$

For the radius to pixels mapping, we suppose that the resolution of a image is (N_1, N_2). N_1 and N_2 are width and height respectively. And the pixel sizes (radius per pixel - ps) are used for convert radius to resolution (eg. $ps_x = \frac{l}{N_1}$). Then, the equation 2 can be converted into equation 9.

$$\begin{aligned} I(l,m) &= I(x \cdot ps_x, y \cdot ps_y) && (6) \\ &= \frac{1}{n} \sum_u \sum_v \sum_w V(u,v,w) e^{i(uxps_x + vyps_y + w(n-1))} && (7) \\ &= \frac{1}{n} \sum_{\hat{u}} \sum_{\hat{v}} \sum_w V(\frac{\hat{u}}{ps_x}, \frac{\hat{v}}{ps_y}, w) e^{i(\hat{u}x + \hat{v}y + wz)} && (8) \\ &= \frac{1}{n} \sum_w \mathcal{F}(V(\frac{\hat{u}}{ps_x}, \frac{\hat{v}}{ps_y}, w)) e^{iwz} && (9) \end{aligned}$$

Combining equation 9 and equation 4. Then, $F(a,b,c)$ is 2D Fourier transform and can be calculated by fast Fourier transform. Fast Fourier transform can not applied on w axis, because z is randomly distributed not like x and y.

$$\begin{aligned} I(l,m) &= I(x \cdot ps_x, y \cdot ps_y) && (10) \\ &= \frac{1}{n\hat{\phi}(a,b,c)} \sum_a \sum_b \sum_c G(a,b,c) e^{i(ax+by+cz)} && (11) \\ &= \frac{1}{n\hat{\phi}(a,b,c)} \sum_c F(x,y,c) e^{icz} && (12) \end{aligned}$$

$$F(x,y,c) = \sum_a \sum_b G(a,b,c) e^{i(ax+by)} \tag{13}$$

$$G(a,b,c) = \sum_{\hat{u}} \sum_{\hat{v}} \sum_w V(\frac{\hat{u}}{ps_x}, \frac{\hat{v}}{ps_y}, w)\phi(a-\hat{u}, b-\hat{v}, c-w) \tag{14}$$

2 Implementation

We implemented explicit gridder, Non-uniform Fourier transform and some other opertaions in CUDA.

2.1 Explicit Gridder

The mapping and decomposition is based on the output data, which means each pixel corresponds to one thread. Each thread completes their own computation and get its value of this pixel. The correctness has been tested.

2.2 Non-uniform Fourier transform

The randomly distributed points to non-randomly distributed points Fourier transform is implemented in three stages (Conv, FFT and Deconv).

Stage 1: Conv. The location of random points which are not in $[-\pi, \pi)$ are shifted into this range. Due to its periodization, the value does not change. Then, calculating the discrete convolution.

$$G(a,b) = \sum_{j=0}^{M} c_j \phi(a - u_j, b - v_j) \qquad a|b \in (1, 2, 3 \cdots n) \tag{15}$$

The value of n (grid size) is determined by upsampling factor and image size (or non randomly distributed points). The width of mask function is set based on Barnett's work [1].
Stage 2: FFT. The CUDA fourier transform library `cufft` is used for computing discrete 2D fast Fourier transform.

$$F(x,y) = \sum_{a} \sum_{b} G(a,b) e^{i(ax+by)} \tag{16}$$

Stage 3: Deconv. The integral of mask function for deconvolution is computed by Gaussian quadrature [2]. α_i is the mask width. And the value is the correction factor for deconvolution.

$$\hat{\phi}(a,b) = \int_{-\alpha_1}^{\alpha_1} \int_{-\alpha_2}^{\alpha_2} \phi(u,v) e^{-i(ua+vb)} du dv \tag{17}$$

The 2D Non-uniform Fourier transform $\mathcal{F}(f(u,v)) = F(x,y)/\hat{\phi}(a,b)$.

For the implementation of convolution, we assigned one thread for each input coordinate. The thread calculates convolutional values $(\phi(d) \cdot c_j)$ for grid points and adds the values to corresponding grid points $(G(a,b))$ which are in the mask width.

However, the concurrent write should be avoided. Figure 1 gives an example, the $\phi(d) \cdot c_j$ should be added into $g(a_i)$, $g(a_{i+1})$ and $g(a_{i+2})$. d denotes the distance between x_j and a_i. For deconvolution, each thread calculates its own correction factor.

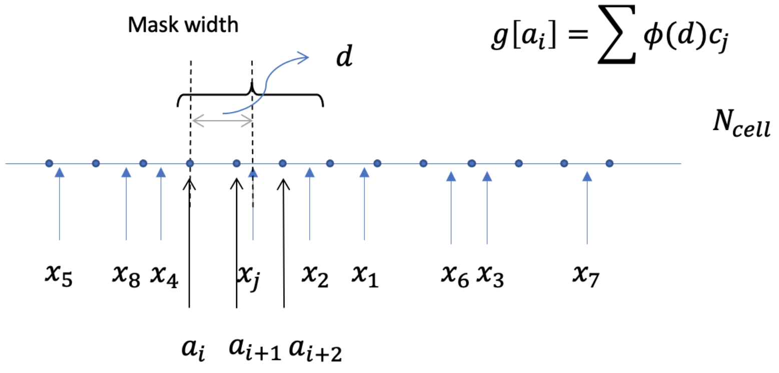
Mask width
d
$g[a_i] = \sum \phi(d)c_j$
N_{cell}
x_5 x_8 x_4 x_j x_2 x_1 x_6 x_3 x_7
a_i a_{i+1} a_{i+2}

Figure 1: Convolution example

2.3 Other operations

For w term processing, the grid size (n) is determined by the maximum and minimum of w. Arras et al. [3] gave the way to choose the number of w planes (N_w), see equation 18. *sigma* is upsampling factor. Δw is the distance between two plane. The maximum and minimum are calculated by reduction. The computation related to w term is also decomposed by input data.

$$N_w = \frac{max(w) - min(w)}{\Delta w} + \alpha \tag{18}$$

$$\Delta w = min(\frac{1}{2\sigma|n_{lm} - 1|}) \tag{19}$$

For the entire workflow, it consists of convolution, fast Fourier transform, discrete Fourier transform on w term, deconvolution and some ending work.

3 Further work

For convolution, coalesce memory access can be achieved by sorting the input data. And in the sort part, multi-pass may be tried.
For Gaussian quadrature, we could implement a GPU version for integral.

References

[1] Alex H. Barnett, Jeremy F. Magland, and Ludvig af Klinteberg. A parallel non-uniform fast fourier transform library based on an ”exponential of semicircle” kernel, 2019.

[2] Andreas Glaser, Xiangtao Liu, and V. Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM J. Sci. Comput.*, 29:1420–1438, 2007.

[3] P. Arras, M. Reinecke, R. Westermann, and T. Ensslin. Efficient wide-field radio interferometry response. *arXiv: Instrumentation and Methods for Astrophysics*, 2020.