2020 – 02

CRYO-RALIB - A MODULAR LIBRARY FOR ACCELERATING ALIGNMENT IN CRYO-EM

Szu-Chi Chung*, Cheng-Yu Hung*, Huei-Lun Siao*, Hung-Yi Wu*, Wei-Hau Changy†, I-Ping Tu*

December 25, 2020

CRYO-RALIB - A MODULAR LIBRARY FOR ACCELERATING ALIGNMENT IN CRYO-EM

Szu-Chi Chung[⋆]*, Cheng-Yu Hung*[⋆]*, Huei-Lun Siao*[⋆]*, Hung-Yi Wu*[⋆]*, Wei-Hau Chang*[†]*, I-Ping Tu*[⋆]

[⋆]Institute of Statistical Science, Academia Sinica.
[†]Institute of Chemistry, Academia Sinica.

ABSTRACT

Thanks to automated cryo-EM and GPU-accelerated processing, single-particle cryo-EM has become a rapid structure determination method that permits capture of dynamical structures of molecules in solution, which has been recently demonstrated by the determination of COVID-19 spike protein in March, shortly after its breakout in late January 2020. This rapidity is critical for vaccine development in response to emerging pandemic. This explains why a 2D classification approach based on multi-reference alignment (MRA) is not as popular as the Bayesian-based approach despite that the former has advantage in differentiating subtle structural variations under low signal-to-noise ratio (SNR). This is perhaps because that MRA is a time-consuming process and a modular GPU-acceleration package for MRA is still lacking. Here, we introduced a library called *Cryo-RALib* that contains GPU-accelerated modular routines for accelerating MRA-based classification algorithms. In addition, we connect the cryo-EM image analysis with the python data science stack so as to make it easier for users to perform data analysis and visualization. Benchmarking on the Tai-Wan Computing Cloud (TWCC) container shows that our implementation can accelerate the computation by one order of magnitude. The library has been made publicly available at https://github.com/phonchi/Cryo-RAlib.

***Index Terms*—** Computational biology, cryo-EM, GPU acceleration, multiple reference alignment

1. INTRODUCTION

In contrast to X-ray crystallography, cryo-EM is a method that is amenable to the structural determination of proteins in non-crystalline state. With the instrument automation and advances in algorithms, single particle cryo-EM has become a mainstream tool to solve 3D structures of molecules at near-atomic resolution. It is noted that as the automated data collection is becoming mature, the equipment that is up-running 24/7 can generate 1000 to 4000 micrographs per day. Therefore, the GPU hardware has been invoked to accelerate the cryo-EM workflow in different stages to meet the demand for processing a large volume of data. The GPU is now used in various steps in data processing including movie alignment [1], contrast transfer function estimation [2], identification of particles within micrographs [3], Bayesian 2D classification algorithm like RELION [4] or cryoSPARC [5] and 3D refinement algorithms [4,5]. Even with GPU acceleration, the popular RELION 2D classification usually takes several days to finish the classification task. Evidently, image classification has become the bottleneck in the workflow. We and others notice that the computational complexity of the Bayesian approach is much higher than the one based on multireference alignment (MRA). The complexity is at least in the order of $O(Kt^2nL^4 \log n)$ compared with $O(KnL^3)$ of MRA, where n is the sample size and L is the pixel number for one direction of the particle image, and t is the translational shift search range [6]. Besides, to further extend to atomic resolution, one must carefully deal with heterogeneity within the image data. When such is concerned, an MRA-based method has advantage because it can better differentiate subtle structure differences under low SNR compared with the Bayesian approach [7, 8].

2. PREVIOUS WORKS

2D and 3D classification algorithms based on the MRA or Bayesian approach are standard steps in cryo-EM workflow [6]. The 2D algorithm is depicted as follows. We use 2D to simplify the illustration while the extension to 3D is straightforward. First, K initial 2D average images are randomly generated or provided by the user. Second, the particle images are compared with those K initials as references by tuning parameters of shifts and in-plane rotations. The reference that maximizes the cross-correlation (CC) is to be recorded for each image. The new 2D averages can then be updated with the aligned images. After multiple iterations of refinement on parameters, the class assignment of each image along with its shift and rotation parameters to a particular 2D class is usually unambiguous. The popular Bayesian approach like RELION [4] is slightly different in that it uses fuzzy assignments for both alignment parameters and classes instead of the best one. The 2D averages are then obtained through weighted averages over all possible orientations and classes. Although the Bayesian approach is very powerful in 3D refinement [4], the algorithms based on MRA have been shown to provide better performance in 2D classification. For instance, CL2D [9] can

reduce the influence of noise and avoid the unbalance class phenomenon, ISAC [10] is an algorithm that ensures robust classes to be output by repeated tests, while Prime [11] can escape the local minimum by adopting stochastic hill-climbing. There are many implementation strategies for the alignment step in MRA [12]. Among these implementations, re-sample to polar coordinate (RPC) has been shown to have the best performance under low SNR so that we use it in this work. It is noted that most of the works do not exploit GPU in the MRA step. However, since the calculation of similarities is independent of particles, 2D reference and orientation, it is possible to calculate them in parallel using GPU.

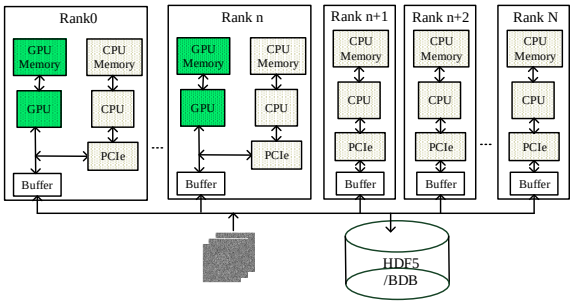

Fig. 1: The architecture is a summary of the coding framework underlying the GPU-accelerated ISAC [13]. We employ this architecture to implement MRA.

Fortunately, RPC has been GPU-accelerated in reference-free alignment (RFA) of GPU ISAC [13], whose code has been released in July 2020. We summarize the architecture of the coding framework underlying the GPU-accelerated ISAC in Fig. 1. This architecture is implemented using message passing interface (MPI) and Nvidia CUDA and the processing flow is described as follows. First, the particle images are read in parallel and immediately pre-processed by all N MPI processes. Second, every process is sent to the first n MPI processes that contain GPU resources through MPI. Third, the first n MPI processes perform the alignment algorithm. Finally, the first n MPI processes send the data back to all N MPI processes and all the processes write the metadata and the transformed images to the disk. Notice that the main task in RPC procedure is to compute the rotation cross-correlation function. The function is defined as $c(\phi) = \int_{r_1}^{r_2} \int_0^{2\pi} x(r,\theta) y(r,\theta+\phi)|r| \, d\theta \, dr$. To speed up the host-to-device memory transfer, the authors in [13] employ the texture memory to store the image as well as its metadata, and a floating array in global memory is used to store the images after polar conversion, Fast Fourier Transform (FFT), Inverse FFT (IFFT) or alignment. The results of the CC computation are stored in a table which holds the CC information of all (mirrored) images with all reference images together with all shifts. Each row in the table stores all CC information for one image.

3. THE MRA FRAMEWORK

In this work, we exploit the parallelism and architecture mentioned in Section 2. Our framework is built upon the general-purpose cryo-EM processing library called EMAN2 [14] and the MRA is based on GPU ISAC [13]. The architecture and data layout of MRA we use is from the GPU-accelerated RFA in [13]. We describe the acceleration of MRA in this section and provide a simple link between cryo-EM image processing and the python data science stack to enable rapid development of GPU-accelerated operations which is described in Section 4. The programs are integrated into a library called *Cryo-RALib* which is accessible to the cryo-EM community.

Algorithm 1: GPU Accelerated MRA

Input: Set of particle images X
Set of reference images Y
Output: Set of class averages Y'

```
repeat
    for each batch do
        Transfer X and Y in this batch from host
          memory into texture memory in device.
        do in parallel
            Convert y_i to polar coordinates and apply
              FFT to obtain y'_i.
        for each shift do
            do in parallel
                Convert x_i to polar coordinates with
                  given shift and apply FFT to obtain
                  x'_i. Compute CC between x'_i and y'_i
                  and store CC into the table.
            Apply inverse FFT to CC table.
        do in parallel
            Find the largest CC value for each image
              using reduction. Find the corresponding
              alignment parameters and apply them to
              each image.
        Compute new Y' from aligned images.
        Transfer averages and parameters to host.
until convergence;
Return Y' and parameters.
```

Our GPU implementation follows the framework of CPU implementation from EMAN2, as shown in Algorithm 1. In other words, we provide a GPU version of MRA from EMAN2. To do so, we made use of existing libraries with optimized CUDA primitives, the polar coordinate conversion, FFT, IFFT and applying alignment parameters are from GPU ISAC [13]. Our CC computation of $FFT(x)' \times FFT(y)$ in MRA is similar to the one used in RFA of [13]. The difference is that each block, which processes one particle image, now computes its CC values with all the given reference

images instead of single reference. This is done by adding another layer of parallelism for the reference images in the y-dimension of GPU grid.

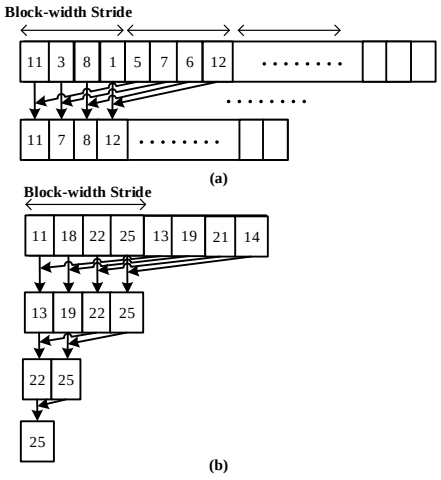

Fig. 2: The reduction strategy. (a) Initialize the shared memory. (b) General reduction on the shared memory.

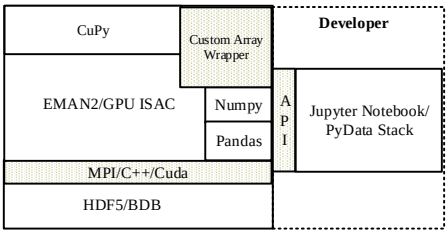

Fig. 3: The software stack of our framework. The shaded blocks represent the interfaces with other packages.

The parallel reduction scheme we employed to find the maximum value of each row in the table is shown in Fig. 2. Here, we are not deciding how many threads to launch based on the data size since the length of each row is very large and can easily exceed the maximum number of threads allowed in a block. Instead, we launch a fixed number of threads and let each thread loop through memory, computing partial reduction operations on each element to initialize the memory. Finally, the standard parallel reduction is employed to reduce the initialized memory to find the largest element and its index. It is noted that our approach also enables coalesced access of the memory within each block during the initialization which can improve the memory access speed. After finding the index for each image, the corresponding alignment parameters are calculated for each image in parallel with GPU kernel. Finally, the new class averages are calculated through CuPy [15] using our custom class depicted in next section.

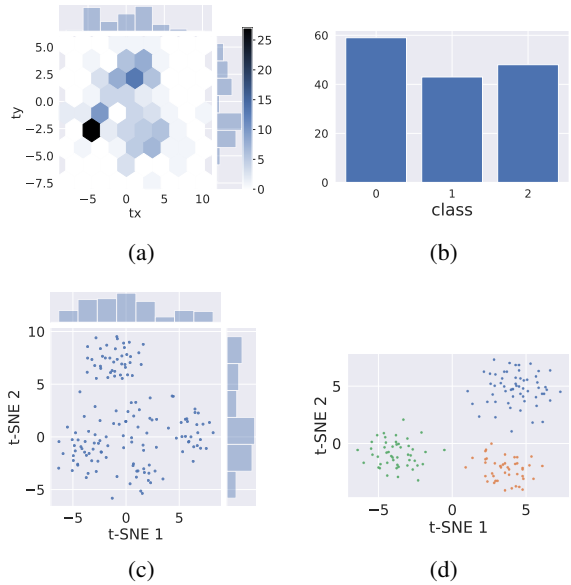

Fig. 4: Histogram of (a) x and y shift, (b) class assignments. t-SNE plot of (c) all particles before alignment, (d) after alignment with each class labeled in different colors. From (c),(d), we can perceive that the particles are better aligned after MRA as it groups nearest neighbors into the same class.

4. PROPOSED SOFTWARE STACK

The software stack of our library is shown in Fig. 3. Popular computational cryo-EM packages are built around C and C++ [4, 14]. On the other hand, the recent trend of packages designed for data analysis and data visualization is to build upon Python libraries. However, there is no convenient way to perform cryo-EM data analysis and visualization in the Python environment. In our framework, the functionality of EMAN2 and the CUDA code is made available through several well-defined interfaces. We used CuPy to implement the required interoperability layer. To compute the averages and calculate the statistics of the transformed images, we implement a custom class as the GPU array interface. This class encapsulates the *__cuda_array_interface__* which is created for interoperability between different GPU arrays in various python projects. With this class, we can easily access the GPU buffer of the transformed images and calculate them in Python to reduce the need to transfer data between host and device as much as possible. In the Python environment, we use NumPy [16] and CuPy array to represent the particle images; it can be transformed into other popular computer vision or machine learning libraries for analysis [17]. Finally, the metadata is represented as Pandas [18] dataframe for exploratory data analysis. The images and metadata are stored into either HDF5 or Berkeley DB (BDB) files. Developer or user can use the Jupyter notebook interface with our API as a pipeline for common data analysis. Several notebooks are

included in the library to show that basic image operations like rotation and shift can be easily accelerated and the data analysis/visualization can be performed on the platform.

The processing pipeline of our framework is illustrated as follows. First, the data source (cryo-EM images and metadata stores in HDF5, BDB or text file) are to be read in through the Python wrapper in parallel. Second, the user performs preprocess or exploratory data analysis in the Jupyter notebook. Third, the alignment and clustering are performed by calling the MRA script. Finally, the transformed images and metadata can be read into a notebook for further analysis or visualization. Fig. 4 shows an example of exploratory data analysis. With the plotted 1D and 2D histogram of shifts and class assignments from MRA procedure are plotted, we can then analyze the performance on particle picking or diagnose the problems like preferred orientations or the attraction by large clusters [9]. In addition, we use 2SDR [19] and t-SNE [20] to plot the 2D embedding of all particles as shown in Fig. 4 (c),(d). We can then get an idea about the alignment progress and the performance of 2D classification algorithm.

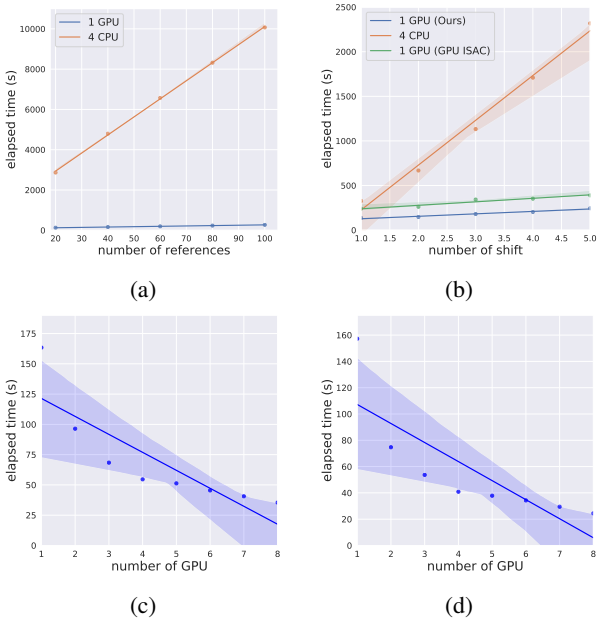

Fig. 5: (a) Computation time of MRA for a different number of references. (b) Computation time of RFA for different search ranges of shift. (c) The scaling behavior of MRA. (d) The scaling behavior of RFA.

5. THE EXPERIMENT RESULTS

Comparisons between different implementations are conducted on TaiWan Computing Cloud (TWCC). The programs were running in the TWCC c.super instance, with 4 cores of Xeon Gold 6154 processors and an Nvidia Tesla V100 GPU. The benchmark dataset of Ribosome 80s [21] from RELION is used with height and width both down-sampled to 90 pixels. We first compared the CPU implementation of MRA from EMAN2 as shown in Fig. 5(a). The search range in the x and y direction is set to 3 and the radius of particles is 36 pixels. We can perceive that the speedup is $22\times$ to $37\times$ with different reference numbers. We then compared the computation time of CPU implementation of RFA from EMAN2 as shown in Fig. 5(b). The search step is set to 1 and the speedup is $2.4\times$ to $9.4\times$ with different 2D shifts. We also compared with the RFA in GPU ISAC and the speedup is about $1.6\times$. Fig. 5(a) presents the performance of our implementation increase that is gained over multiple-GPU on the c.4xsuper instance with 8 GPUs. It can be observed that our implementation scales well from 1 to 4 GPUs and provides near linear speedup compared to single GPU. On the other hand, when the number of GPU is greater than 4 the speedup is saturated because the time is dominated by the data transfer.

6. CONCLUSION AND FUTURE WORKS

With continuing enhancement of processing algorithms, cryo-EM has become a progressively powerful and efficient technique to solve structures of molecules. Currently, the performance of the popular GPU-accelerated Bayesian approach for 2D Classification in cryo-EM still poses a bottleneck in the processing as the computation time and the classification quality are sub-optimal. As a result, MRA-based approaches represent an option as it can better reveal structural variations at the 2D level. To the best of our knowledge, until now MRA-based methods are mostly implemented with CPU-supported algorithms and thereby time-consuming. In this work, a modular GPU-accelerated alignment library termed *Cryo-RALib* is introduced. The library contains a set of alignment routines that can be used to accelerate 2D classification algorithms. We also provide several well-defined interfaces to connect the cryo-EM image analysis with the python data science stack to help users to perform analysis and visualization more easily. Our benchmarking results on TWCC show this implementation can accelerate the alignment procedure by one order of magnitude. In the future, it would be interesting to expand the library to accommodate other cryo-EM packages into this framework, and to provide a friendly environment for users and developers as well.

Acknowledgement

The authors gratefully acknowledge Ryan Jeng from Nvidia and the NCHC GPU Hackathon for the help on our research project. The authors also acknowledge the open-source projects that we built upon, especially the EMAN2 and GPU ISAC 2.3.2. Finally, the authors acknowledge the insightful suggestions from Dr. Stefan Raunser.

7. REFERENCES

[1] Shawn Q Zheng, Eugene Palovcak, Jean-Paul Armache, Kliment A Verba, Yifan Cheng, and David A Agard, "Motioncor2: anisotropic correction of beam-induced motion for improved cryo-electron microscopy," *Nature Methods*, vol. 14, no. 4, pp. 331–332, 2017.

[2] Kai Zhang, "Gctf: Real-time ctf determination and correction," *Journal of structural biology*, vol. 193, no. 1, pp. 1–12, 2016.

[3] Thorsten Wagner, Felipe Merino, Markus Stabrin, Toshio Moriya, Claudia Antoni, Amir Apelbaum, Philine Hagel, Oleg Sitsel, Tobias Raisch, Daniel Prumbaum, et al., "Sphire-cryolo is a fast and accurate fully automated particle picker for cryo-em," *Communications Biology*, vol. 2, no. 1, pp. 1–13, 2019.

[4] Sjors HW Scheres, "Relion: implementation of a Bayesian approach to cryo-em structure determination," *Journal of Structural Biology*, vol. 180, no. 3, pp. 519–530, 2012.

[5] Ali Punjani, John L Rubinstein, David J Fleet, and Marcus A Brubaker, "cryosparc: algorithms for rapid unsupervised cryo-em structure determination," *Nature Methods*, vol. 14, no. 3, pp. 290–296, 2017.

[6] Amit Singer and Fred J Sigworth, "Computational methods for single-particle electron cryomicroscopy," *Annual Review of Biomedical Data Science*, vol. 3, pp. 163–190, 2020.

[7] Jiayi Wu, Yong-Bei Ma, Charles Congdon, Bevin Brett, Shuobing Chen, Yaofang Xu, Qi Ouyang, and Youdong Mao, "Massively parallel unsupervised single-particle cryo-em data clustering via statistical manifold learning," *PLoS ONE*, vol. 12, no. 8, pp. e0182130, 2017.

[8] S.C. Chung, H.H. Lin, P.Y. Niu, S H. Huang, I.P. Tu, and W.H. Chang, "Pre-pro is a fast pre-processor for single-particle cryo-em by enhancing 2d classification," *Communications Biology*, vol. 3, no. 1, pp. 1–12, 2020.

[9] C.O. Sorzano, J.R. Bilbao-Castro, Y. Shkolnisky, M. Alcorlo, R. Melero, G. Caffarena-Fernández, M. Li, G. Xu, R. Marabini, and J.M. Carazo, "A clustering approach to multireference alignment of single-particle projections in electron microscopy," *Journal of Structural Biology*, vol. 171, no. 2, pp. 197–206, 2010.

[10] Zhengfan Yang, Jia Fang, Johnathan Chittuluru, Francisco J Asturias, and Pawel A Penczek, "Iterative stable alignment and clustering of 2d transmission electron microscope images," *Structure*, vol. 20, no. 2, pp. 237–247, 2012.

[11] Cyril F Reboul, Frederic Bonnet, Dominika Elmlund, and Hans Elmlund, "A stochastic hill climbing approach for simultaneous 2d alignment and clustering of cryogenic electron microscopy images," *Structure*, vol. 24, no. 6, pp. 988–996, 2016.

[12] Laurent Joyeux and Pawel A Penczek, "Efficiency of 2d alignment methods," *Ultramicroscopy*, vol. 92, no. 2, pp. 33–46, 2002.

[13] Fabian Schoenfeld, "GPU ISAC (2.3.2)," `http://sphire.mpg.de/wiki/doku.php?id=gpu_isac/`, 2020.

[14] Guang Tang, Liwei Peng, Philip R Baldwin, Deepinder S Mann, Wen Jiang, Ian Rees, and Steven J Ludtke, "Eman2: an extensible image processing suite for electron microscopy," *Journal of Structural Biology*, vol. 157, no. 1, pp. 38–46, 2007.

[15] ROYUD Nishino and Shohei Hido Crissman Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," *31st confernce on neural information processing systems*, p. 151, 2017.

[16] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] The pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[19] S.C. Chung, S.H. Wang, P.Y. Niu, S.Y. Huang, W.H. Chang, and I.P. Tu, "Two-stage dimension reduction for noisy high-dimensional images and application to cryo-em," *Annals of Mathematical Sciences and Applications*, vol. 5, no. 2, pp. 283–316, 2020.

[20] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[21] Wilson Wong, Xiao-chen Bai, Alan Brown, Israel S Fernandez, Eric Hanssen, Melanie Condron, Yan Hong Tan, Jake Baum, and Sjors HW Scheres, "Cryo-em structure of the plasmodium falciparum 80s ribosome bound to the anti-protozoan drug emetine," *Elife*, vol. 3, 2014.