E-Zone: A Faster Neighbor Point Query Algorithm for Matching Spacial Objects

Xiaobin Ma[1], Zhihui Du[1](✉), Yankui Sun[1], Yuan Bai[2], Suping Wu[2], Andrei Tchernykh[3], Yang Xu[4], Chao Wu[4], and Jianyan Wei[4]

[1] Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing, China
duzh@tsinghua.edu.cn
[2] College of Information Engineering, Ningxia University, Yinchuan, China
[3] CICESE Research Center, Ensenada, Mexico
[4] National Astronomical Observatories, Chinese Academy of Sciences, Beijing, China

Abstract. Latest astronomy projects observe the spacial objects with astronomical cameras generating images continuously. To identify transient objects, the position of these objects on the images need to be compared against a reference table on the same portion of the sky, which is a complex search task called cross match. We designed Euclidean-Zone (E-Zone), a method for faster neighbor point queries which allows efficient cross match between spatial catalogs. In this paper, we implemented E-Zone algorithm utilizing euclidean distance between celestial objects with pixel coordinates to avoid the complex mathematical functions in equatorial coordinate system. Meanwhile, we surveyed on the parameters of our model and other system factors to find optimal configures of this algorithm. In addition to the sequential algorithm, we modified the serial program and implemented an OpenMP parallelized version. For serial version, the results of our algorithm achieved a speedup of 2.07 times over using equatorial coordinate system. Also, we achieved 19 ms for sequencial queries and 5 ms for parallel queries for 200,000 objects on a single CPU processor over a 230,520 synthetic reference database.

Keywords: Cross match · Zone · Parallel · OpenMP

1 Introduction

Ground-based Wide Angle Cameras (GWAC) [7] in China is a telescope build for observing the sky continuously, which producing raw images every dozens of seconds. Millions of entries of celestial property catalogs will be extracted from these images. Spacial objects in the catalog flow need to be matched against reference data to identify them, subsequently observing astronomy incidents like Gamma Ray Bursts (GRBs) [6]. Cross match [11], is the procedure to identify an object by querying the distance among its neighbors in the image and find the nearest stars within a distance threshold. Hence, the speed of cross match and its accuracy are crucial to proceed scientific analysis of the data flow and ensure timely alerts of astronomical incidents. This procedure has to be fast enough over millions of objects on the celestial sphere. Accelerating this critical procedure, consequently, is of momentous significance.

Dozens of algorithms together with data structures, like *Hierarchical Equal-Area iso-Latitude Pixelisation (HEALPix)* [9] and *Hierarchical Triangular Mesh (HTM)* [14], have been put forward to meet different needs of a query on 2-dimensional datasets. They speed up the search procedure by decomposing and combining the search area. These methods work well on the celestial sphere coordinate system but strict distance test between these reference objects is inavoidable. So, we try to use plane coordinate system to simplify the distance calculation and speed up the search. In this paper, improvements are made based on this idea.

The major contributions of our paper are as follows: Firstly, we designed E-Zone algorithm, a memory-based neighbor points query algorithm using euclidean distance to speed up the calculation of distance between objects, which has gotten rid of the SQL database dependance. Secondly, we tested different parameters to find optimal configurations of the algorithm. Thirdly, we modified the serial algorithm with OpenMP and reached another 4 times speed up than the serial version. In this paper, we introduced the background of astronomical cross match and discussed the related works in Sect. 2. Section 3 presents the sequential E-Zone algorithm and its performance analysis. Section 4 evaluates different configurations and OpenMP to accelerate the search by lunch multiple queries. Finally, Sect. 5 concludes this paper.

2 Background and Related Work

A dominating area query algorithm called *Zone* proposed by Gray et al. [8]. It maps the celestial sphere into quantities of small horizontal zones, and reduces the search space by apportioning the search task only to these possible zones. Each zone is a declination stripe of the sphere with certain $zoneHeight$: $zoneNumber = \boldsymbol{floor}((dec + 90)/zoneHeight)$.

For given two celestial objects O_1, O_2 in spherical coordinates (ra_1, dec_1) and (ra_2, dec_2), the central angle(distance in equatorial coordinate system) between these two objects is:

$$\begin{aligned} Distance(O_1, O_2) = arccos(sin(dec_1)sin(dec_2) \\ + cos(dec_1)cos(dec_2)cos(|ra_1 - ra_2|)) \end{aligned} \tag{1}$$

But at least six trigonometric functions need to be called for calculating the distance between two celestial objects to get the distance between them. It is an nonnegligible subroutine that may consume considerable time when compared with few multiplications and additions (euclidean distance).

Hierarchical Triangular Mesh (HTM) [14] is a multilevel, recursive decomposition of the celestial sphere. Like all other algorithms accelerating the query, it tries to minimize the search space by decomposition and combination of predefined region units. By decomposing repeatedly, the celestial sphere finally will be divided into billions of little spherical triangular. The search algorithm returns a list of *HTM* triangles that cover that region, which is the minimum superset of all the query point's neighbors. Wang et al. [15] utilized both CPU and GPU to implement a similar zone divided cross match algorithm called *gridMatch*. They use a grid file to store and compute the hierarchical data structure used to store the celestial objects. Many other multicore parallel cross match algorithms for catalogs are proposed to meet different needs in various scenarios [10,13,16]. Some researchers use statistical method to improve the completeness and reliability of cross matching [12]. Multi-GPU featured astronomic catalog cross match system is proposed for workstation and cluster [5].

3 E-Zone Algorithm with Pixel Euclidean Distance

The raw images captured by GWAC are encoded with *FITS* (Flexible Image Transport System) [2], the most common digital file format designed for storage, transmission and processing astronomical images. The right ascension (*Ra*) [4] and declination (*Dec*) [1] which describe the position of the celestial object in the equatorial coordinate, were generated through this process. Thus, we can assume that there exits another projection α converting the position of the objects of the raw image $(Raw\_spher_x, Raw\_spher_y)$ into a plane coordinate (P_pixel_x, P_pixel_y), which means that the projected coordinate is certificated to calculate euclidean distance between two objects:

$$(Euclidean\_pixel_x, Euclidean\_pixel_y) = \alpha(Raw\_spher_x, Raw\_spher_y) \quad (2)$$

Equation 2 is a coordinate mapping between raw image pixel coordinate and the plane-coordinate system, where the distance can be calculated as ordinary euclidean distance. So, it is easy to express the distance between $O_1(x_1, y_1)$ and $O_2(x_2, y_2)$ using euclidean distance to avoid complex trigonometric functions: $Distance_{euc}(O_1, O_2) = \sqrt{(O_1.x - O_2.x)^2 + (O_1.y - O_2.y)^2}$.

3.1 Decompose the Celestial Sphere

Despite the approach, the calculation of the distance between two points and the definition of `zone` are different when using pixel coordinate. All the algorithms and definitions using (*ra*, *dec*) now need to be adapted to the new coordinate system. Similarly, we decompose the vision into horizontal strips according to its E_PIX_y. The zoneID using pixel coordinate (E_PIX_x, E_PIX_y) will be:

$$zoneID = floor(\frac{E\_PIX_y}{zoneHeight}) \quad (3)$$

We only need to consider the objects between the *maxZoneID* and *minZoneID*, when querying the objects within the circle of radius

R: $maxZoneID = floor(E\_PIX_y + R/zoneHeight)$, $minZoneID = floor(E\_PIX_y - R/zoneHeight)$.

The difference in the searching zones lies only in the unit, pixel, because the way E-Zone decompose the sphere is the same as in the original zone algorithm.

3.2 Neighbors with Pixel Distance

Without SQL database, we need to manage the data manually. The reference table needs to be preprocessed to implement the search algorithm. The reference object table is almost static. It hardly changes in a long period. Thus, this table seldom insert and delete while queries and indexing are the most frequent operations on this data structure. As a result, we do not need to be efficient at deleting or inserting elements in a B-tree. Therefore, we can implement simple arrays to meet our needs.

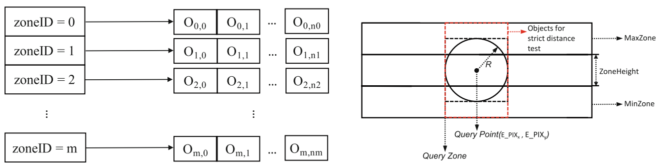
zoneID = 0
zoneID = 1
zoneID = 2
zoneID = m
$O_{0,0}$ $O_{0,1}$ ... $O_{0,n0}$
$O_{1,0}$ $O_{1,1}$ ... $O_{1,n1}$
$O_{2,0}$ $O_{2,1}$ ... $O_{2,n2}$
$O_{m,0}$ $O_{m,1}$ ... $O_{m,nm}$
Objects for strict distance test
MaxZone
R
ZoneHeight
MinZone
Query Point(E_PIX_x , E_PIX_y)
Query Zone

Fig. 1. The data structure and the procedure to determine the minimal strict distance test area. (Color figure online)

According to the structure shown in Fig. 1, we use a bunch of *zoneObjectList* (the arrays on the right) that is stored by E_PIX_y containing all the objects with the same *zoneID*. These arrays are indexed by another pointer array *zoneList* (the vertical array on the left), which designed for indexing zone vectors with its corresponding *zoneID* as a bucket.

We build the search tree as follows:

Step 1: Calculate all *zoneID* with formula (3) and find the object list's pointer by $zoneList[zoneID]$.

Step 2: Insert the reference objects into the object vector $*zoneList[zoneID]$ respectively.

Step 3: Sort all object vectors respectively by E_PIX_x.

The right subfigure of Fig. 1 shows, what we need to do to find the objects within a search threshold we have to find the zones between two bounds on the vertical axis. Subsequently, linear search task can find out which part of the plane need to conduct strict distance test (red dotted zone). Our optimization is mainly focused on these metrics. Any linear search algorithm can be applied to subsequent search procedure (eg. Binary search tree).

4 Experiment

In this section, we evaluate the performance of our E-Zone algorithm using pixel distance. First, we use an algorithm that is only differentiated by the distance formula, to compare and verify our speed up by euclidean distance measurement Then we feed the algorithm with different sizes of reference data to compare its time and space complexity with our analytic result. Finally, we implemented a simple parallel query program with OpenMP programming model [3] and evaluate its effectiveness.

4.1 Superiority in Using Euclidean Distance

According to the analysis in Sect. 3, angle calculation with trigonometric function is a cumbersome procedure that will slow down this program deeply. These two methods are implemented without parallelization.

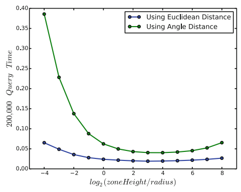

Fig. 2. Time consumption using euclidean distance and angle distance. (Serial Version)

As Fig. 2 shows, the logarithm of the ratio between $zoneHeight$ and $Radius$, which varies along with the $zoneHeight$ with a geometric progression from 2^{-3} to 2^8. The evaluation index is the average query time of 200,000 among 100 runs, excluding the time build the reference table and I/O. The minimum speed-up ratio is 2.069 when $zoneHeight/Radius = 16$. And if $zoneHeight/Radius = \frac{1}{16}$, the performance difference between two methods is the most prominent. The ratio is 5.928, but the absolute speed of this algorithm is not the optimal at this circumstance.

4.2 Performance with Parallel Queries

The catalogs obtained from raw images contains millions of objects. There is no data dependence between two queries, so higher performance could be achieved by launching these queries in parallel. We applied the simple and effective parallel model OpenMP to the outer for loop which invokes the search function. Great performance acceleration is achieved with OpenMP.

The left subplot of Fig. 3 shows the performance of the program written with and without OpenMP parallelization. With the growth of query size, the speed-up ratio becomes better, but becomes worse when $zoneHeight/R > \frac{1}{2^4}$. As the left subfigure of Fig. 3 shows, the ratio is higher on bigger datasets and that OpenMP works more efficiently on big zones, which agreed with that switch between threads for task dispatch consumes lots of time.

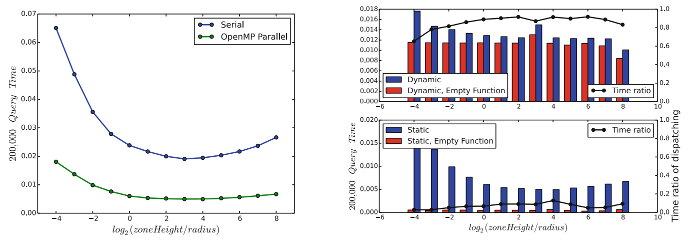

Fig. 3. Parallel performance with different environment and input parameters.

After we distribute these tasks with OpenMP on eight cores, the best average query time is reduced to around 0.0498 s for 200,000 queries over a 230,000 reference object table (Average time over 100 runs). After we adopt static dispatch strategy, we achieved a 2x speed-up over *dynamic*. We replaced the *queryPointEZone* function (seen as the code snippet above) with an empty function, which returns a constant value directly without any other statement, meanwhile, all other settings are the same. The result is shown in the right subfigure of Fig. 3. Time finishing 200,000 queries has reduced to around 5 ms. Hence, it's feasible to save time with parallel queries, and it achieved satisfying efficiency.

5 Conclusion

We find that (1) Using euclidean distance for constraint of neighbor points queries is effective in accelerating the cross match procedure of celestial queries. The formula used to compute the angle between two celestial objects in spherical coordinate is time-consuming that will slow the procedure down for over 2x. (2) Our serial cross match method can finish searching 200,000 queries over 230,520 reference objects within 0.019 s which reached 1.37x speed up over *gridMatch* algorithm. (3) The OpenMP parallelized version can finish this task within 5 ms, which achieved another 1.55x speed up on 8 cores with 16 threads. The CPU utilization is over 80%, when conducting the zone search procedure. Thus our metric is more efficient than traditional celestial angle-based metrics. For future work, we consider that it is feasible to combine the simplified coordinate system and distance measurement with GPU to achieve better performance. Moreover, a CPU/GPU hybrid algorithm has good potentials for better performance.

Acknowledgment. This research is supported in part by the National Key Research and Development Program of China (No. 2016YFB1000602 and 2017YFB0701501), the

Key Laboratory of Space Astronomy and Technology, National Astronomical Observatories, Chinese Academy of Sciences, MOE research center for online education foundation (No. 2016ZD302), and National Natural Science Foundation of China (Nos. 61440057, 61272087, 61363019).

References

1. Declination. https://en.wikipedia.org/wiki/Declination
2. Flexible image transport system. https://fits.gsfc.nasa.gov
3. openmp. http://openmp.org/wp/
4. Right ascension. https://en.wikipedia.org/wiki/Right_ascension
5. Budavari, T., Lee, M.A.: Xmatch: GPU enhanced astronomic catalog cross-matching. Astrophysics Source Code Library (2013)
6. Fishman, G.J., Meegan, C.A.: Gamma-ray bursts. Astron. Astrophys. **33**(33), 415–458 (2003)
7. Godet, O., Paul, J., Wei, J.Y., Zhang, S.-N., Atteia, J.-L., Basa, S., Barret, D., Claret, A., Cordier, B., Cuby, J.-G., et al.: The Chinese-French SVOM mission: studying the brightest astronomical explosions. In: Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray, vol. 8443, p. 84431O. International Society for Optics and Photonics (2012)
8. Gray, J., Szalay, A.S., Thakar, A.R., Fekete, G., O'Mullane, W., Nietosantisteban, M.A., Heber, G., Rots, A.H.: There goes the neighborhood: relational algebra for spatial data search. Computer Science (2004)
9. Górski, K.M., Hivon, E.: Healpix: hierarchical equal area isolatitude pixelization of a sphere. Astrophysics Source Code Library (2011)
10. Jia, X., Luo, Q.: Multi-assignment single joins for parallel cross-match of astronomic catalogs on heterogeneous clusters. In: International Conference on Scientific and Statistical Database Management, p. 12 (2016)
11. Nietosantisteban, M.A., Thakar, A.R., Szalay, A.S.: Cross-matching very large datasets. Santisteban (2008)
12. Pineau, F.X., Derriere, S., Motch, C., Carrera, F.J., Genova, F., Michel, L., Mingo, B., Mints, A., Gómezmorán, A.N., Rosen, S.R.: Probabilistic multi-catalogue positional cross-match. Astron. Astrophys. **597**, A89 (2016)
13. Riccio, G., Brescia, M., Cavuoti, S., Mercurio, A., Di Giorgio, A.M., Molinari, S.: C3, a command-line catalogue cross-match tool for large astrophysical catalogues. Publications Astron. Soc. Pac. **129**(972), 024005 (2016)
14. Szalay, A.S., Gray, J., Fekete, G., Kunszt, P.Z., Kukol, P., Thakar, A.: Indexing the sphere with the hierarchical triangular mesh. Microsoft Research (2007)
15. Wang, S., Zhao, Y., Luo, Q., Wu, C., Yang, X.: Accelerating in-memory cross match of astronomical catalogs. In: IEEE International Conference on E-Science, pp. 326–333 (2013)
16. Zhao, Q., Sun, J., Yu, C., Cui, C., Lv, L., Xiao, J.: A paralleled large-scale astronomical cross-matching function. In: Hua, A., Chang, S.-L. (eds.) ICA3PP 2009. LNCS, vol. 5574, pp. 604–614. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03095-6_57