3D Skeleton Extraction Method using Potential Field on OpenCL

Lu Lu
Department of Computer Science and Engineering
South China University of Technology
Guangzhou, People's Republic of China
lul@scut.edu.cn

Xuewen Wang
Department of Computer Science and Engineering
South China University of Technology
Guangzhou, People's Republic of China
xuewen.wang@foxmail.com

***Abstract*—For 3D skeleton extraction, the algorithm based on generalized potential fields, known as the outstandingly flexible and robust method, is suffering from seriously heavy computational burden. In this paper, we put forward a parallel algorithm based on OpenCL heterogeneous parallel framework, which can make full use of the great computing power provided by heterogeneous model of CPU+GPU. This algorithm focuses on computing the potential field of each interior point in parallel, with the goal of cutting down the time of potential field calculation to relieve the whole computational burden of this extraction algorithm. The proposed parallel algorithm was evaluated by using several large 3D object volumes. From the tests, we can find that the whole calculation time can be reduced up to 5 to 10 times, without affecting the extraction's accuracy.**

Keywords: parallel computation, OpenCL, heterogeneous parallel framework, 3D curve skeleton.

I. INTRODUCTION

Skeleton of 3D object is its representation in one dimension. For a more formal definition, it's the locus of centers of interior maximal circles (2D) or spheres (3D). Skeleton of 3D object is widely used in many fields such as computer animation, visual navigating, model recognition, Collision detection, Motion planning and so on. Therefore, for the last years, researchers have paid high attention to it , and a series of mature methods are found out for this problem. Some of them are based on the extension of methods for 2D objects and some are born for 3D objects.

Generally speaking, several various approaches are as following:

(1) Based on Topology thinning. Intend to constantly cut so-called simple points without changing the topology of model in order to realize the skeleton extraction. Gong proposed the parallel thinning algorithm and reduced the original cost [1].

(2) Distance field based method uses the distance of each interior volume with the boundary volume to obtain the ridge points of model. Sundar designed an approach to calculate the minimal distance and build up a model retrieval system [2].

(3) Cornea proposed the skeleton extraction algorithm based on the potential field between volumes, which applies the concept of potential field in physics to get the balance points of model and finally extracts skeleton according these key points [3].

(4) Geometric methods deal with those 3D objects described as triangular or polygonal meshes. They usually generate the Voronoi graph of the 3D object at the beginning, and then draw the medial surface according to the graph. Finally get the skeleton in 1D after the clip of medial surface [4].

(5) Based on model decomposition. Lien [5] notices that the decomposition of model without destroying its connectivity is just like so-called skeleton extraction. Therefore he puts forward the algorithm based on decomposition of approximate convex body of model. Get sag sex of each vertex trough calculating the surface bridge and recognizing surface concave ground and iteratively do the decomposition according the sag sex to create the skeleton.

These approaches are mostly implemented serially in CPU, some of which are suffering from high computing burden and consuming lots of resources. Besides, some methods are designed for parallel execution with multi-core CPU, but they don't bring out large improvements in execution consumption. For those 3D objects represented by discrete data, the algorithm based on potential field works well, but still, suffers high computational burden. Several improved extraction algorithms based on potential field are presented in [6-8], however, there still has large space to improve since this kind of algorithm has its potential parallelism. If the execution time decreases to be acceptable, it will become a better choice than others for skeleton extraction, along with its perfect extraction result. Parallel computing with GPU [9] makes it possible.

GPU is born as a fixed-function special-purpose processor and has turned into a full-fledged parallel programmable processor with additional fixed-function special-purpose functionality, along with its rapid development in computing. GPU has the features such as flow processing, high dense parallel computing, programmable line etc. In addition, its ability of floating-point calculations is more powerful than that of CPU. As a result, more and more people pay attention to general-purpose GPU. General-purpose GPU [10] applies the heterogeneous model of CPU+GPU, where CPU is responsible for complex logic processing such as branch judgment and affairs management, those not suitable for parallel computing, and GPU is in charge of handling intensive large-scale data parallel computing. This

computing model, making full use of GPU's powerful processing ability and high storage bandwidth to make up for the shortage of CPU performance, has great advantage in achieving high performance by exploring potential computing ability of computer and controlling the cost. Researchers have done some studies on GPU such as Local Search [11], Large Graph algorithm [12], simulate the Mush-room Cloud [13], bringing out perfect improvements

This paper centers on two important points, one is making full use of GPU's ability in high-dense data parallel computing. The other one is digging out its potential market and discussing how to map applications from CPU to GPU. So we put forward a parallel method for potential field algorithm implemented based on OpenCL heterogeneous framework [14-15]. It tries to solve the problem about consuming too much computing resources along with enormous calculation load by making the most of CPU+GPU heterogeneous model. In this model, CPU cares about logic control and GPU provides its powerful parallel computing ability. This type of computing model overcomes the limit of CPU cores for multi-thread, since GPU has hundreds of cores and is able to meet the need of large scale of data parallel. 3D skeleton extraction for large objects requires great computing power and costs a lot, this paper wants to cut down the computing burden through a parallel algorithm implemented using OpenCL framework. It's a new try to combine parallel computing with CPU+GPU heterogeneous model.

II. SKELETON EXTRACTION BASED ON POTENTIAL FIELD

In physics, we all know that an electrical charge generates a potential field, and whenever another charged object approaches, there will be a force between them [7]. The force can be identified by the formula: $F=kq_1q_2/d^2$, *where* k is a constant, q_1, q_2 is the charge and d is the distance between them. When we apply the theory to application of computer graphic, k will be ignored. For expression in 3D format, it's described as:

$$F_x = \frac{x_2-x_1}{|d_{12}|^{\alpha+1}} \quad F_y = \frac{y_2-y_1}{|d_{12}|^{\alpha+1}} \quad F_z = \frac{z_2-z_1}{|d_{12}|^{\alpha+1}}. \qquad (1)$$

Where α is the force strength, x_1, x_2 are the positions where the charges are placed. In addition,

$$|d_{12}| = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}. \qquad (2)$$

The algorithm is under such assumption: there are electrical charges distributed upon the outer surface of 3D objects uniformly. And the core of this algorithm is to calculate the potential field force of each interior point, and then do the skeleton extraction on the basis of force data. For an interior point I, and a set of boundary points S, the total force vector generated by points in S is:

$$F(I) = \sum_{P\in S} F_x(I,P)\times i + \sum_{P\in S} F_y(I,P)\times j + \sum_{P\in S} F_z(I,P)\times k. \qquad (3)$$

Where i, j, k represent the unit vectors in three coordinates. Furthermore, only P that is visible to I should be calculated, and visibility can be identified in mathematics as the following:

P is visible to $I \Leftrightarrow \boldsymbol{\lambda} I+(1-\boldsymbol{\lambda})P$ is internal, for $\forall \mathbf{0} < \boldsymbol{\lambda} \leq \mathbf{1}$.

Considering visibility can reduce the count of points in set S calculated for point I, algorithms based on the above formula require high realization complexity. There're some easy methods: setting up suitable range domain for three directions [3] , setting up a fit radius with using sphere model , considering neighbors' type and then use special distance r for each type, such as $1,\sqrt{2},\sqrt{3}$ [16].

The core of PF (potential field) algorithm is:

a. boundary points of object are represented by set S $(P_1, P_2, \ldots, P_n)$,
b. interior points are in set T $(I_1, I_2, \ldots, I_m)$,
c. vectors for interior points are described as set $Force$ $(F_1, F_2, \ldots, F_m)$,
d. $Fm = \sum_{P\in S} F_x(Im,P)\times i + \sum_{P\in S} F_y(Im,P)\times j + \sum_{P\in S} F_z(Im,P)\times k$.

III. PARALLEL ALGORITHM USING OPENCL

Time consumption of calculating potential field vectors for interior points composes the main part of the whole extraction time, thus if it can be reduced effectively, the whole time will be reduced effectively. The process of serial implementation on CPU is shown in Fig.1.

```
make up set T,S;
for I in T
    for P in S
      if ( I – limit < P < I + limit )
       calculate distance between I and P;
       calculate Fpx, Fpy, Fpz;
          sum to Fx, Fy, Fz;
       end if
    end for
end for
```

Figure 1. The process of serial implementation

There are two for loops which can be made for parallel execution. At the beginning, our original idea is trying adopting the advantage of OpenCL so as to amplify the power of parallelization as much as possible, where PEs (processing elements) will execute the same kernel for each interior voxel, and the kernel is responsible for calculating the vector obtained by outer voxels. But to our

disappoint, since all the workitems for each interior voxel should sum up their personal result, this method needs support for atom operations which are not provided for double data type by GPU. As with such a reason, the final solution is about just cutting one loop down to parallel execution. Process of the final adopted parallel realization on heterogeneous model of CPU+GPU is shown in Fig. 2, as a result of the consideration that interior voxels are much more than the outer voxel.

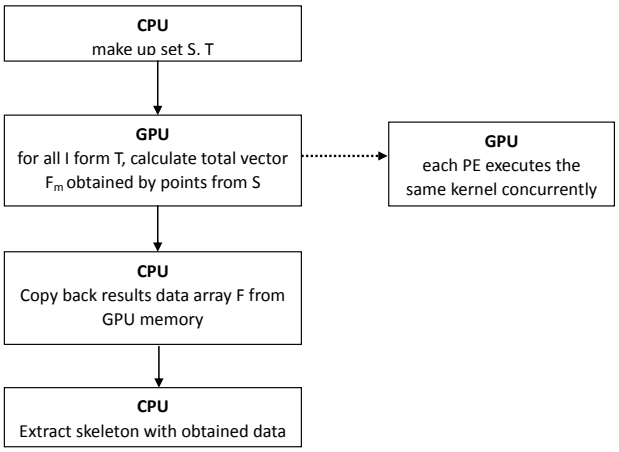

Figure 2. Parallel execution process diagram

For the programming of parallel implementation on GPU, firstly, allocate memory objects identified for interior and outer voxels. Since GPU don't provide support for allocating memory dynamically, it needs to define data objects storing information in CPU memory and then copy them to memory object defined in GPU. This time, we deal with 3D data and need to use self-define data type which is supported by GPU. Array *Interior*, *Outer*, *Force* were used for storing information about interior, outer voxels and potential field vectors, parameter limit is used to reduce the amount of outer voxels that are active in the calculation, *numOuter* stands for the size of array Outer and the power strength is identified by parameter *segma*.

Secondly, assign the potential field vector calculation of each interior voxel into a PE on GPU, where each PE will execute the same kernel responsible for obtaining total vectors. In this step, it needs to allocate a one-dimension index space with *numInterior* work-items, where *numInterior* is the count of interior voxels. What calls for special attention is the design about how the kernel works, which is shown in Fig. 3. It describes how the algorithm is redesigned to be suitable for GPU. By the way, GPU needs to support double data type by adding such code: #pragma OPENCL EXTENSION cl_khr_fp64: enable.

Kernel CalPotentialField (*Outer*, *Interior*, *Force*, *sigma*, *numOuter*, *limit*)

1: id←get_global_id(0)
2: Force[id] . xd ← 0, Force[id] . yd ← 0,
Force[id]. zd ← 0
3: **for** all voxels para in Outer **do**
4: **if** para is far away from Interior[id] with limit distance
5: $v_1 \leftarrow$ Interior[id]. x − Outer[para].x
6: $v_2 \leftarrow$ Interior[id]. y − Outer[para].y
7: $v_3 \leftarrow$ Interior[id]. z − Outer[para].z
8: temp ←sqrt($v_1 * v_1 + v_2 * v_2 + v_3 * v_3$)
9: r← $temp^{sigma+1}$
10: Force[id].xd←Force[id].xd+v_1 / r
11: Force[id].yd←Force[id].yd+v_2 / r
12: Force[id].zd←Force[id].zd+v_3/ r
13: **end if**
14: **end for**

Figure 3. Kernel code of potential field algorithm on GPU

IV. EXPERIMENT RESULTS

This paper focuses on improving the execution of skeleton extraction algorithm through adopting the new computing method: heterogeneous parallel computing with OpenCL. To validate the parallel algorithm, we compare its implementation using OpenCL with serial realization of the original algorithm on CPU. Both of them are implemented in C and C++ language, and the extracted skeletons are shown by OpenGL, rendering obtained key points.

The experiment is designed based on the open procedure provided by Cornea from Rutgers University [17]. With limit of time and technology, this pager just cares about reducing time of calculating the potential field vectors. Several tests have been made to prove the algorithm feasible and effective. The details information of the extracted skeleton is showed in Fig. 4 and Fig. 5. And the most suitable value for parameter strength is ∂=5 found from many tests, and the divergence threshold used in tests is 25%.

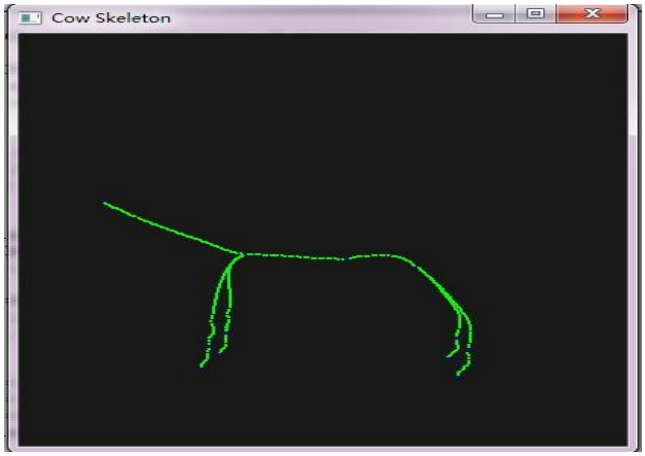

Figure 4. Core skeleton of cow

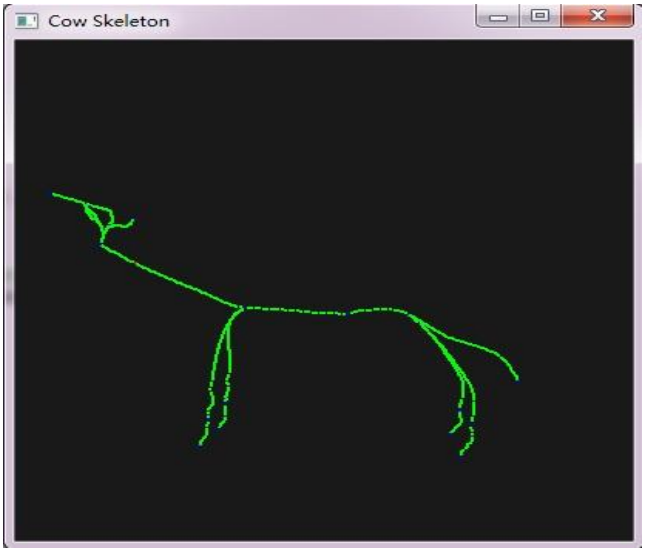

Figure 5. Skeleton of cow（divergence threshold is 25%）

As is shown in Table I and Fig. 6, while the size of 3D object is within certain range, potential field (PF) calculation composes the main part of the whole algorithm (Knight and Cow indicate this case). What's more important to be mentioned is that, the proposed method makes PF calculation time rise linearly and slowly, even though there is a highly increase in object's size. Pay attention again to Fig. 6, you can find the parallel method factually decreases the influence caused by interior points. If the size is too large, it will increase the count of boundary points along with that of interior points largely, but the effect of interior points is cut down by our algorithm, then the main time consumption will be the left processing about extracting skeleton. Just as shown by the object colon and m112, the whole time is much longer than that of PF calculation.

Table II shows the execution information on CPU and GPU. The exciting results shown in Fig. 7 prove, for large 3D objects, this proposed parallel algorithm based on OpenCL framework really reduces the computational load and improves efficiency of potential field method with a speedup factor of 5-10. At the same time, it guarantees the accuracy, not like the method applied in article [7].

TABLE I. EXECUTION TIME FOR PARALLEL IMPLEMENTATION

3D object	volume size			outer points	inter points	PF cal-time（ms）	total time (ms)
	L	M	N				
Knight	40	39	87	6444	29246	921	1622
Cow	85	31	54	6555	28357	1280	2247
m112	130	300	32	30110	144531	3089	11123
Colon	204	132	100	28023	165500	2902	16676

TABLE II. TIME FOR SERIAL AND PARALLEL IMPLEMENTATION WITH THE SPEEDUP

3D objects	serial time (ms)		parallel time(ms)		speedup	
	PF	Total	PF	Total	PF	Total
Knight	14586	16302	921	1622	15.8	10.1
Cow	15662	17457	1280	2247	12.2	7.77
Colon	29765	41434	2902	16676	10.2	2.48

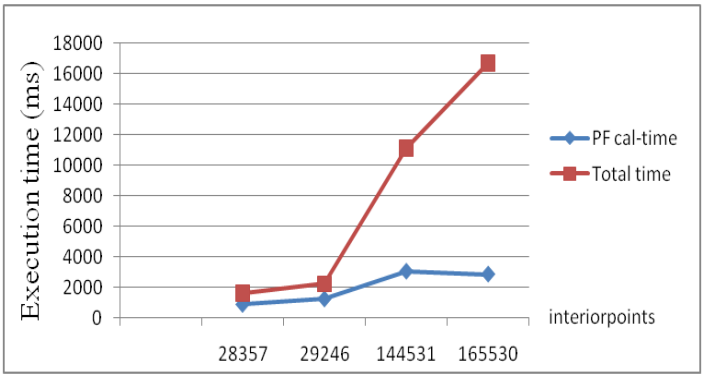

Figure 6. Execution time for PF and total calculatio

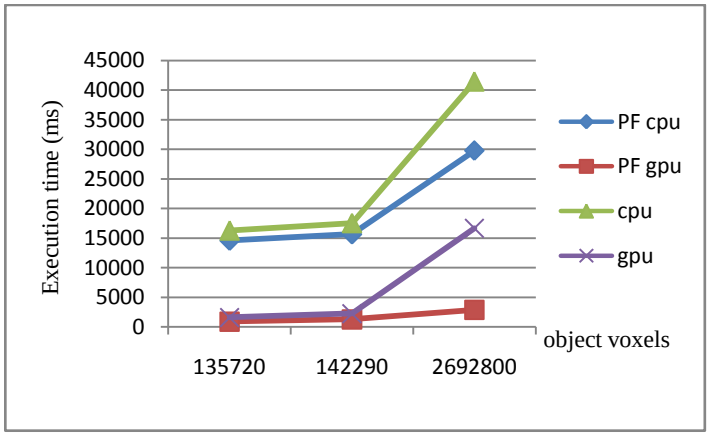

Figure 7. Execution time for CPU and GPU implementation

V. CONCLUSION

In this paper, we propose a parallel algorithm for 3D skeleton extraction using OpenCL and prove its validity and efficiency through simulation experiments. Experiments show that efficiency of the GPU parallel implementation of the skeleton extraction algorithm based on the generalized potential field, compared to that of traditional CPU serial implementation, has been greatly improved, and the speedup is basically 5 ~ 10 times.

We concern on the implement calculating internal force vectors in parallel. But the number of boundary point is also very large, if we can continue to reduce the number of boundary points participated in the calculation,

the efficiency of the whole algorithm will continue to gain more improvement. Therefore, our next step could be to take into the visibility issues, going on reducing the cost of time and resources. On the other hand, some steps can't be executed concurrently with the considering that GPU's processing ability in logic is much worse than CPU. In addition, GPU has some limits itself, such as limited video memory and register etc physical properties, besides, the number of concurrent threads is also restricted. All of these restrictions have led to the fact that the algorithm fits for 3D objects having certain restriction in its size. Considering how to use multi-GPU in parallel computing is our future work.

VI. ACKNOWLEDGMENT

This paper is supported by Guangdong Production, Education & Research Project (2011B090400139), Guangzhou Production, Education & Research Project (2011Y5-0004).

REFERENCES

[1] G. WeiXin, B. Gilles, "A Simple parallel 3D Thinning Algorithm", In ICPR, ICS Press, ed. 1990, pp. 188-190.

[2] H. Sundar, D. Silver, N. Gagvani, S. Dickinson. ,"Skeleton Based Shape Matching and Retrieval", Proc. Conf. Shape Modeling International 2003.

[3] N.D. Cornea, D. Silver, X. Yuan, R. Balasubramanian, "Computing hierarchical curve-skeletons of 3D objects", The Visual Computer 21, 2005, pp. 945–955 .

[4] T. K. Dey, "Approximating the Medial Axis from the Voronoi Diagram with a Convergence Guarantee", Algorithmica, 2001, 38, (1), pp. 179-200.

[5] L. Jyh-Ming, M. A. Nancy,"Simultaneous Shape Decomposition and Skeletonization Using Approximate Convex Decomposition", Proc. Int. Conf. ACM Solid and Physical Modeling, Cardiff, Wales, UK, Jun 2006, pp. 219-228.

[6] B. Yunna, S. Xiaodong, Zh. Hongbin, "A semantic segmentation algorithm of 3D model", Proc. Int. Conf. Networked Computing and Advanced Information Management (NCM), 2011, pp. 222-225.

[7] L´aszl´o Szil´agyi, S´andor Mikl´os Szil´agyi, David Icl˘anzan, Lehel Szab´o., "Efficient 3D Curve Skeleton Extraction from Large Object", Proc. Int. Conf., Springer-Verlag, Berlin, Heidelberg, 2011, pp. 133-140.

[8] J.Chuang, C.Tsai , M.C.Ko, "Skeletonization of three-dimensional objects using generalized potential field", IEEE transactions on pattern analysis and machine intelligence, 2000, 22, (11), pp. 1241-1251.

[9] D. O. John, H. Mike, L. David, G Simon., E. S. John, C. P.James, "GPU computing", Proc. Int. Conf. IEEE, 2008, 96, (5), pp. 879-886.

[10] http://en.wikipedia.org/wiki/GPGPU, GPGPU.

[11] Th ´e Van Luong, M. Nouredine, Talbi E.G., "GPU-Based Multi-start Local Search Algorithms", Proc. Int. Conf. Springer-Verlag, Berlin, Heidelberg, 2011, pp. 321–335.

[12] H.Pawan, P.J. Narayanan, "Accelerating Large Graph Algorithms on the GPU Using CUDA", Proc. Int. Conf. Springer-Verlag, Berlin, Heidelberg 2007, pp. 197-208

[13] C. Xingquan, L. Jinhong, S. Zhitong, "Effcient Mushroom Cloud Simulation on GPU", Proc. Int. Conf. Springer-Verlag Berlin Heidelberg 2008, pp. 695–706

[14] A. Munshi , "The OpenCL Specification Version 1.1"

[15] t R.G. Benedic, L. Howes, K.David , M. Perhaad, S. Dana, Hetero -geneous Computing with OpenCL, Published by Elsevier, 2012, 2nd .

[16] MaRi, TieRu, Wu, "Hierarchy Skeleton of 3D object Based on the generalized force field", Journal of Computer Application, 2011, 31, (1), pp. 17-18

[17] http://coewww.rutgers.edu/www2/vizlab/NicuCornea/Nicu%20D.%20Cornea%20-%20home%20page.html , Nicu Daniel Cornea homepage