

GPUMLib: Deep Learning SOM Library for Surface Reconstruction

Wai Pai Lee^{1,2}, Shafaatunnur Hasan^{1,2}, Siti Mariyam Shamsuddin^{1,2},
Noel Lopes³

¹ UTM Big Data Centre & GPU Research Centre
Universiti Teknologi Malaysia (UTM), 81310 Skudai Johor Bahru Malaysia

² Faculty of Computing,
Universiti Teknologi Malaysia (UTM), 81310 Skudai Johor Bahru Malaysia
E-mail: shafaatunnur@utm.my, mariyam@utm.my

^{1,3}UDI, Polytechnic Institute of Guarda, Portugal
CISUC, University of Coimbra, Portugal
Email: noel@ipg.pt

Abstract

The evolution of 3D scanning devices and innovation in computer processing power and storage capacity has sparked the revolution of producing big point-cloud datasets. This phenomenon has becoming an integral part of the sophisticated building design process especially in the era of 4th Industrial Revolution. The big point-cloud datasets have caused complexity in handling surface reconstruction and visualization since existing algorithms are not so readily available. In this context, the surface reconstruction intelligent algorithms need to be revolutionized to deal with big point-cloud datasets in tandem with the advancement of hardware processing power and storage capacity. In this study, we propose GPUMLib – deep learning library for self-organizing map (SOM-DLLib) to solve problems involving big point-cloud datasets from 3D scanning devices. The SOM-DLLib consists of multiple layers for reducing and optimizing those big point cloud datasets. The findings show the final objects are successfully reconstructed with optimized neighborhood representation and the performance becomes better as the size of point clouds increases.

Keywords: Surface reconstruction, self-organizing map, deep learning, parallel computing, point clouds

1 Introduction

Big data is one of the topical issues nowadays since digital information has grown about nine times in year 2011 compared with the digital information in year 2006. This implies the total amount of data will reach up to 25 trillion gigabytes. The revolution of data brings a lot of opportunities for the various sectors especially to industries and government sectors. They have to adapt the changes gradually by transforming or renovating the traditional solutions to fit with the big data requirement [1].

Although the growth of point cloud data is not as faster as the normal digital data, it is worth to ponder due to the evolution of 3D scanning devices and technology. Furthermore, the number of applications dealing with 3D data processing has increased rapidly because of the maturity of 3D scanning devices [2]. Researchers start proposing new methods or inventions for solving the performance issues when processing large scale 3D point cloud data. Many intelligent algorithms have been conducted dealing with large scale 3D point cloud data such as Artificial Neural Network (ANN), Genetic algorithm, simulated annealing, particle swarm optimization and differential evolution [3,4].

Machine learning (ML) as one of the intelligent algorithms has widely being used in classifying and optimizing large scale datasets. However, dealing with big point-cloud datasets needs advance ML algorithms for better classification and optimization. Deep learning (DL) as one of the advance algorithms is a subset of machine learning, and currently, it has become popular for big data processing. The massive amounts analysis and learning of unsupervised data in DL has made it suitable in processing Big Data [5]. In previous studies, researchers use GPU to achieve parallel processing for improving ML and DL algorithm performance [1] [6] [7] [19]. Unlike CPU, GPU focuses on the throughput instead of latency, and allows recursive processes. DL is normally used for doing business analysis, decisions and predictions [5] [8].

In this paper, SOM has been chosen as an intelligent algorithm due to the relationship among nodes in the mapping grid [9]. For better big point cloud datasets processing, DL approach is embedded in multiple SOM layers. GPU platform is developed for parallelism GPUDLLib is used as the base application interface for GPU platform. In this research, our focus is on the surface reconstruction and representation to improve for 3D big point-cloud 3D. However, the proposed GPUDLLib is not limited to 3D objects only but also to the N -dimensional representation and construction respectively.

The organization of the rest of this paper as follows: Section 2 discusses the related works on the point cloud surface reconstruction, deep learning, SOM, and parallel computing. Section 3 describes the proposed SOM-DLLib: Deep Learning Library for surface reconstruction. Section 4 provides the results and analysis, and finally Section 6 provides discussions and conclusion of the study.

2 Related Work

In the early studies, soft computing techniques were proposed by many researchers as an approach for doing point cloud surface reconstruction. The strength of soft computing techniques in optimization makes it possible to solve point fitting problems [3]. There are many machine learning techniques for surface reconstruction such as Artificial Neural Network (ANN), genetic algorithms, ART and others. SOM as one of ANN models has always been the common tool for reconstructing the point-cloud datasets. This is due to the nature of SOM architecture which has topological neighborhood function and relationship between every vectors [9]. Ademb Junior *et al.* proposed processing surface reconstruction by using SOM algorithm with some vertices connection operation. The vertices relationship in a standard SOM is insufficient to reconstruct a model with concave structure [10]. Therefore, additional connection operation for selection is proposed.

In year 2010, [11] proposed growing SOM for surface reconstruction process. The algorithm used the concept of triangular faces in learning algorithm which appeared in growing neural gas (GNG). This concept allows connection selection and makes output mesh more accurate. In 2008, Forkan and Shamsuddin proposed kohonen swarm optimization for unstructured data in surface reconstruction [18]. Their work introduced a new method for surface reconstruction based on the hybridization of Kohonen Network and Particle Swarm Optimization (PSO). Kohonen network learns the sample data through mapping grid that can grow. The learned and well represented data became the input for surface fitting procedure, while PSO was used to probe the optimum fitting points on the surfaces.

Besides changing the algorithm concepts, some researchers change the shape or structure of the map in SOM. SOM is usually represented by a 2D map but 2D map alone is not promising for reconstructing close surface shape [10] [11]. Therefore, [12] suggested a new shape structure of the map which is a cube shape to fix lack of sides' relationship in a standard SOM for a closed mesh. Due to succeed of SOM algorithm in surface reconstruction, researchers start to use other algorithm that inspired by SOM algorithm which is Neural Gas. In year 2014, [14] used GNG for surface reconstruction [13] [14] to fix the close-gaps and holes produced in 3D surface reconstruction process using SOM. This can save the cost of post-processing procedure for holes filling.

On the other hand, researchers also concern about the time performance of surface reconstruction. In year 2015, [15] implemented parallel computing into surface reconstruction using GPU using Marching Cubes algorithm. In the same period, [16] used GPU to accelerate SOM algorithm for high dimensional data [16]. In fact, integration of SOM algorithm with GPU parallel computing was also proven by Hasan S. *et al.* [6] [19].

Based on these studies, there are still lacking of solving large scale surface reconstruction using ML with GPU. Most of the studies focus on improving the accuracy of output mesh instead of reconstruction speed. DL is proven to be suitable for large scale point cloud datasets. Therefore, a new approach that focuses on performance is proposed for surface reconstruction which integrates the ML, parallel computing and deep learning concepts. Our proposed method involves SOM with multiple layers under GPU environment.

3 The Proposed SOM-DLLib – Deep Learning Library for Surface Reconstruction

The proposed SOM Deep Learning library (SOMDLLib) is one of the components in GPUMLib framework proposed by [6] [17] [19]. GPUMLib is an open source GPU based Machine Learning Library. It was developed using NVidia CUDA C++ provides range of machine learning tools for researchers and practitioners who wish to take advantage of the extreme parallelism offered by the GPU on compute-intensive tasks. A number of machine learning algorithms have been implemented on the GPUMLib, notably of which are Back propagation NN, MBP, SOM and SVM amongst others (see Figure 1). Its source code and other relevant details can be obtained at <http://gpumlib.sourceforge.net/>. The development of our proposed SOMDLLib is based on the framework of GPUMLib. There are four components in SOMDLLib: point cloud data pre-processing, SOM-DLLib multilayer SOM architecture, SOM-DLLib multilayer design and process and SOM-DLLib post-processing for surface reconstruction.

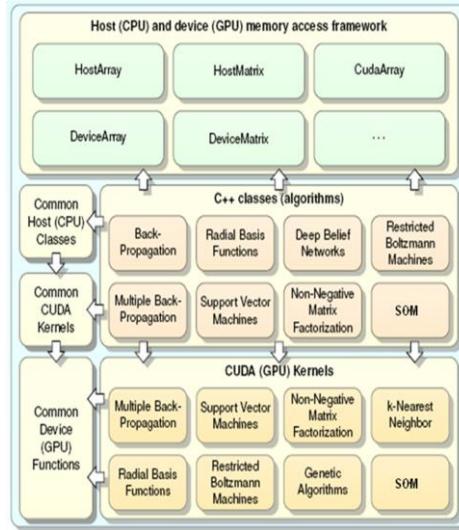


Fig. 1: GPUMLib Framework

3.1 Point Cloud Data Pre-processing

In this study, ply extension file is used for point cloud data based on bunny image (Figure 2). These data will undergo pre-processing process to meet the requirement of SOM GPUDLLib for data inputting. The raw data will be read until it meets the element of vertex syntax. The vertex number is recorded to avoid mismatch input reading of faces data. Data of X, Y and Z for each point is recorded and a unique label is given accordingly (Figure 3).

```

1 ply
2 format ascii 1.0
3 obj_info is_cyberware_data 1
4 obj_info is_mesh 0
5 obj_info is_warped 0
6 obj_info is_interlaced 1
7 obj_info num_cols 512
8 obj_info num_rows 400
9 obj_info echo_rgb_offset_x 0.013000
10 obj_info echo_rgb_offset_y 0.153600
11 obj_info echo_rgb_offset_z 0.172000
12 obj_info echo_rgb_frontfocus 0.930000
13 obj_info echo_rgb_backfocus 0.012660
14 obj_info echo_rgb_pixelsize 0.000010
15 obj_info echo_rgb_centerpixel 232
16 obj_info echo_frames 512
17 obj_info echo_lgincr 0.000500
18 element vertex 40256
19 property float x
20 property float y
21 property float z
22 element range_grid 204800
23 property list uchar int vertex_indices
24 end_header
25 -0.06325 0.0359793 0.0420873
26 -0.06275 0.0360343 0.0425949

```

Fig. 2: Bunny point cloud datasets from Stanford 3D scanning repository

```

1 514.0399 514.0237 0.0000 1
2 534.4382 615.2095 8.0630 2
3 554.8367 611.1620 8.0630 3
4 575.2350 599.0198 8.0630 4
5 591.5538 582.8300 8.0630 5
6 603.7928 566.6403 8.0630 6
7 616.0319 546.4031 8.0630 7
8 620.1116 522.1186 8.0630 8
9 620.1116 501.8814 8.0630 9
10 616.0319 477.5968 8.0630 10
11 603.7928 457.3597 8.0630 11
12 591.5538 441.1700 8.0630 12
13 575.2350 424.9802 8.0630 13
14 554.8367 412.8380 8.0630 14
15 534.4382 408.7905 8.0630 15
16 514.0399 404.7431 8.0630 16
17 489.5618 408.7905 8.0630 17
18 469.1634 412.6380 8.0630 18
19 448.7650 424.9802 8.0630 19
20 432.4462 441.1700 8.0630 20
21 420.2072 457.3597 8.0630 21
22 407.9681 477.5968 8.0630 22
23 403.8885 501.8814 8.0630 23
24 403.8885 522.1186 8.0630 24

```

Fig. 3: Labeled Point Cloud Data for Bunny Object

3.2 SOM-DLLib for Multilayer Architecture

In standard SOM, the connectivity among the neurons in the lattice structure produces holes between points, thus it is suitable for open surfaces. To deal with closed surfaces, SOM-DLLib multilayer architecture is proposed for closed connectivity as illustrated in Figure 4.

The procedures are done by slicing the standard SOM into half-shape to form multiple layers. The Z-axis distance is omitted in the computation of neighborhood distance when the updating the weighted neurons to avoid surface points discontinuity due the layers depth. In this scenario, the distance determining the winning node is computed using 2D calculation from four directions as in Figure 4.

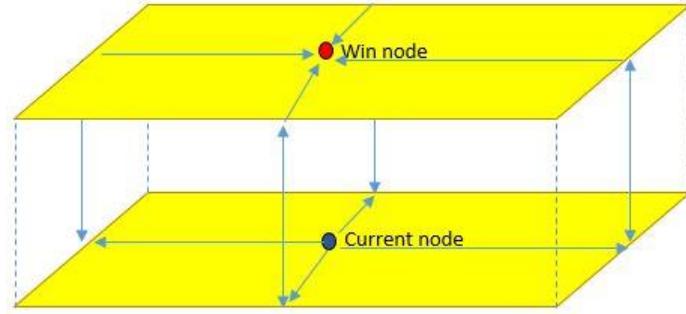


Fig. 4: SOM-DLLib Multilayer Architecture

As the layers increase, the complexity computations arise, and the processing power should be increased as well. Thus, we implement GPU-based parallel computing using CUDA programming to update the weights and distance of the winning node. Reduction techniques are implemented to obtain the smallest distance for the winning node. For weight updating process, each thread is given several nodes for calculating the distance between the winning node and the current node. Figure 5 shows the segment of CUDA coding for computing the distance of SOM-DLLib.

```

if (dz != 0) {
    // 4 direction distance
    leftDist = (tempwinY - tempY) * (tempwinY - tempY); // left
    rightDist = leftDist; // right
    topDist = ((tempY + tempwinY + 1) * (tempY + tempwinY + 1)); // top
    botDist = (((mapy / 2) - tempY) + ((mapy / 2) - tempwinY - 1)) *
        (((mapy / 2) - tempY) + ((mapy / 2) - tempwinY - 1)); // bottom
}
else {
    dy = winy - y;
}

for (int x = threadIdx.y; x < mapx; x += blockDim.y) {
    cudafloat dx = 0;

    if (dz != 0) {
        // 4 direction distance
        leftDist += ((x + winx + 1) * (x + winx + 1)); // left
        rightDist += (((mapx - x) + (mapx - winx - 1)) *
            ((mapx - x) + (mapx - winx - 1))); // right
        topDist += ((winx - x) * (winx - x)); // top
        botDist += ((winx - x) * (winx - x)); // bottom
    }
    else {
        dx = winx - x;
    }

    if (dz) {
        distance = leftDist < rightDist ? leftDist : rightDist;
        distance = distance < topDist ? distance : topDist;
        distance = distance < botDist ? distance : botDist;
    }
    else {
        distance = dx * dx + dy * dy;
    }
    cudafloat influence = exp(-distance /
        (2 * neighbourhoodRadiusSquare));
    if (distance < neighbourhoodRadiusSquare) {
        for(int f = threadIdx.x; f < features; f += blockDim.x) {
            int idx = (y * mapx + x) * features + f;
            weights[idx] += learningRate * influence *
                (inputData[vector * features + f] - weights[idx]);
        }
    }
}

```

Fig. 5: Code Segment for Distance Calculation

3.3 SOM-DLLib Multilayer Design and Process

The proposed SOM-DLLib multilayer design and process for illustrating the deep learning SOM concepts and mechanism is illustrated in Figure 6.

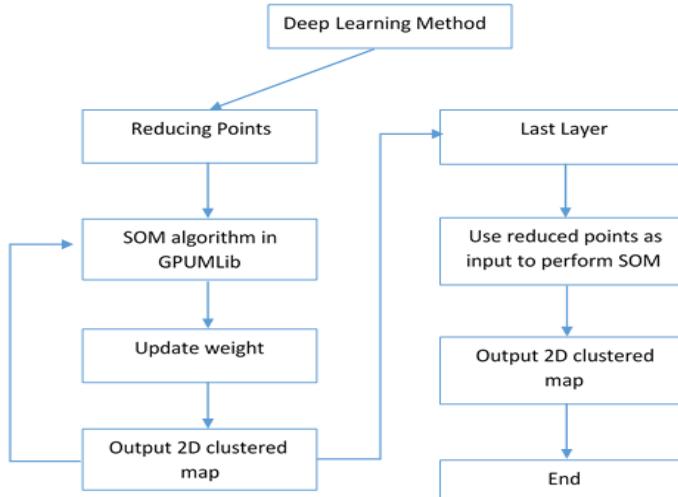


Fig. 6: SOM-DLLib: Deep Learning Design and Process

Two parts are involved in designing and developing the SOM-DLLib: 1) point reduction and 2) point optimization for surface reconstruction in the final layer. For the first part – point reduction involves the selection of significant points for reconstructing the surfaces to minimize the cost. As the iteration for the first few layers are quite small, multiple layers are added for getting better solutions. In this scenario, the first layer reduces the points without considering neurons connectivity, while the second layer, third layer and up to the n^{th} layer are responsible for improving the accuracy of the designated layer output accordingly. However, the output of the mesh points of final mapping and shrinking lattice will be based on the final weights from the final layer.

Figure 7 shows the SOM-DLLib multilayer architecture and each layer is considered as a complete process. In other words, the output from the first layer is sent to the second layer. These outputs will be the input for the second layer processes. As the mapping depth increases, the mapping size is reducing to eliminate noise in the previous layers.

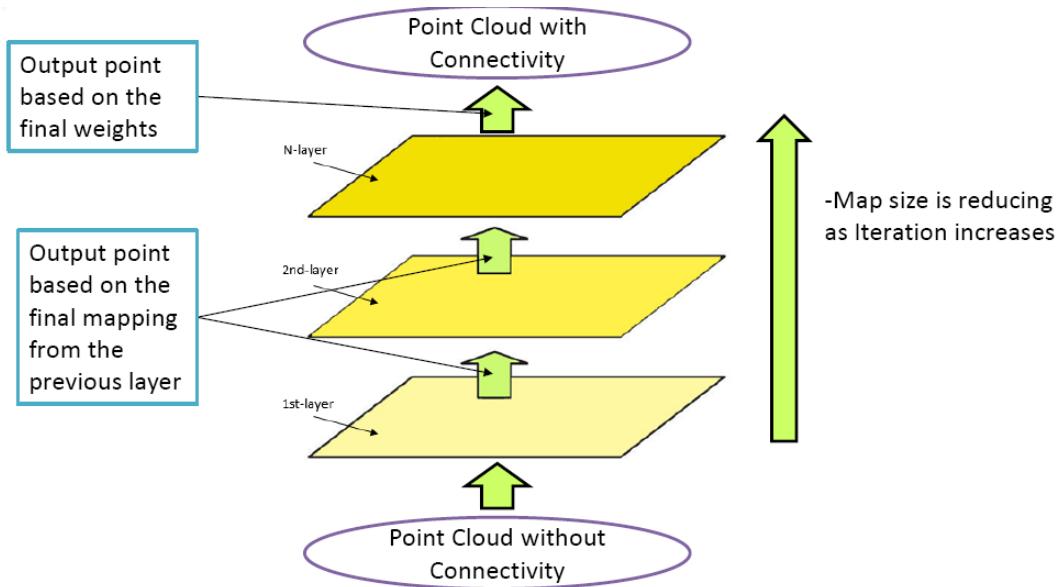


Fig. 7: SOM-DLLib Multilayer architecture

3.4 SOM-DLLib Post-processing for Surface Reconstruction

SOM-DLLib post-processing for converting the output into a 3D mesh file is done by adopting ply format file to represent 3D mesh data.. Figure 8 illustrate nodes connectivity in SOM-DLLib. After generating triangle surface data, the final weights from the last layer of SOM-DLLib are used as the new points for the

mesh. The points and triangle surface data are written into a ply extension file and the output is visualized using Point Cloud Library (PCL) viewer.

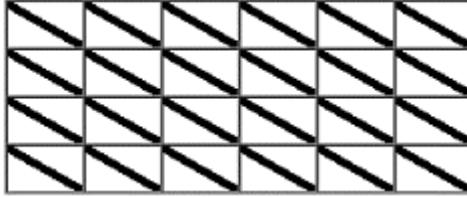


Fig. 8: Nodes connectives in SOM-DLLib

4 Results and Analysis

This section presents the results and analysis of the study together with the discussions. The analysis are given into two sub-sections: the analysis of mesh output and the performance comparison of processing powers in terms of CPU and GPU.

4.1 SOM-DDLib Analysis on the Mesh Output

This section discusses the comparison of output mesh using standard SOM and the proposed SOM-DLLib, follows by the performance evaluation using single layer and deep learning process of multilayer. For the output mesh, there are holes and gaps between the vertices in the output mesh using standard SOM as shown in Figure 9. This is due to the limitation of the neighboring connectivity of 2D SOM as the nodes which lie on the side that only have three direct neighborhoods instead of four, thus causes discontinuity to the final output mesh (see Figure 10).

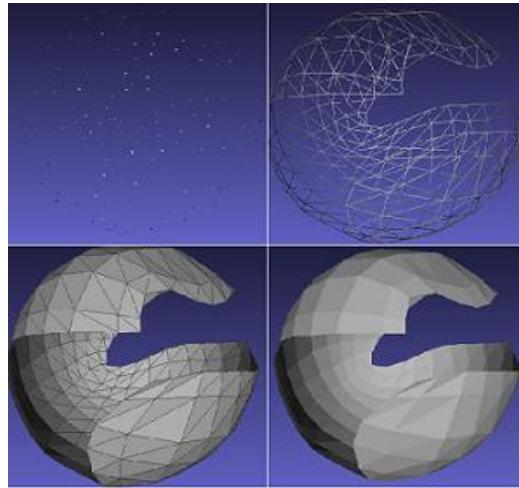


Fig. 9: Sphere Output based on standard SOM

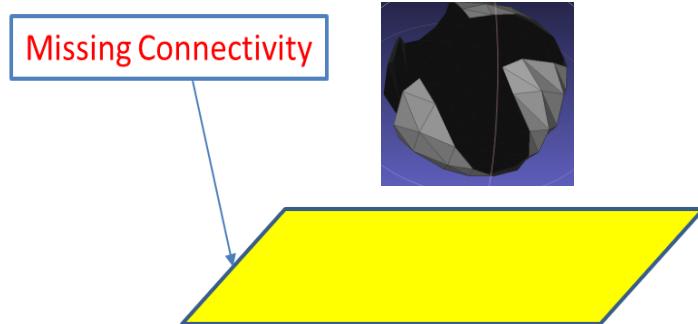


Fig. 10: Limitation of standard SOM

Figure 11 shows the output mesh of the proposed SOM-DDLib for sphere object in which the gaps and holes have been filled accordingly. This is due to the existence of four neighborhoods for every node in the proposed algorithm. Figure 12 illustrates the concept of SOM-DDLib algorithm. It looks like a circular shape, thus closed surface mesh is able to be reconstructed accurately.

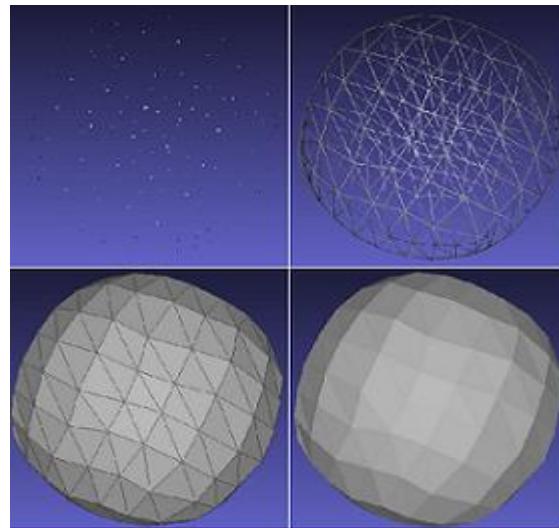


Fig. 11: SOM-DDLib for Sphere

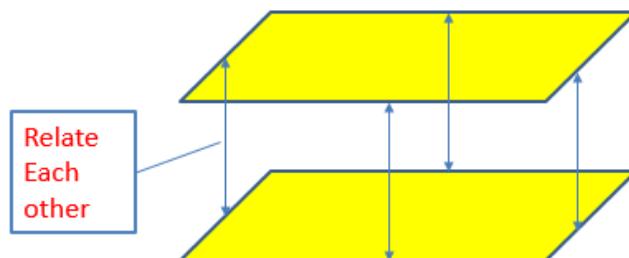


Fig. 12: Neighboring Edges Connectivity in SOM-DLLib

However, the proposed SOM-DDLib has limitations dealing with concave structure in which the nodes unable to identify the neighborhood connectivity correctly. This lead to the missing connectivities between vertices and the output mesh as illustrated in Figure 13, where the bunny's ear was not reconstructed accurately.

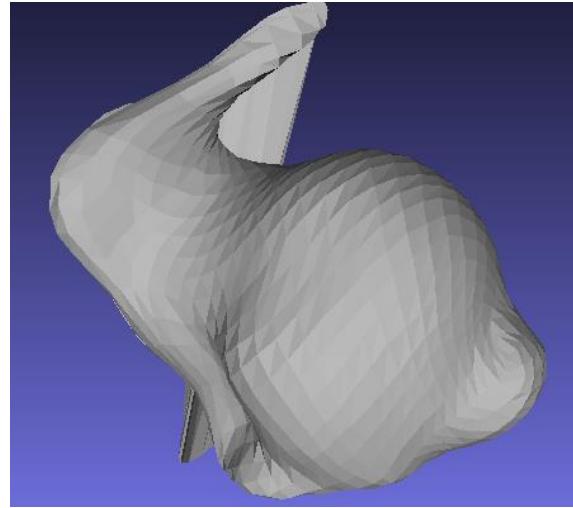


Fig. 13: Bunny Ear Limitation

4.1.1 SOM-DDLib Analysis: Single Layer and Multiple Layer

This section focuses on the output mesh of using single layer and multiple layers in SOM-DDLib. Figure 14 shows the mesh output using single layer approach and Figure 15 shows the mesh output using multiple layers. From the results, both approaches give similar mesh output result, but for the last object (eagle object), the SOM-DLLib with multiple layers have better output. This is due to the capability of SOM-DLLib in selecting the significant points for reconstructing the objects. Therefore, the connectivity can be differentiated significantly compared to the single layer approach.

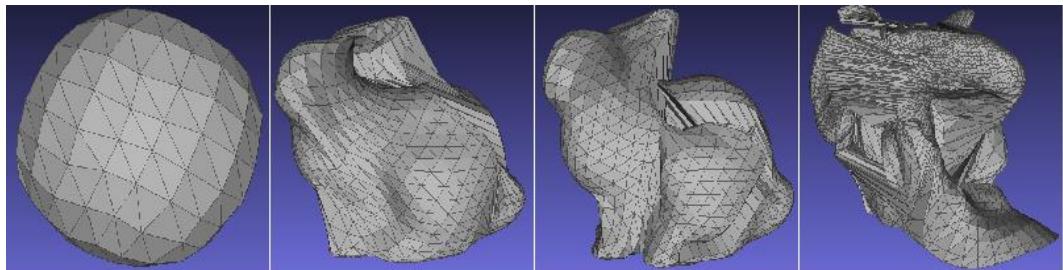


Fig. 14: Single Layer Surface Reconstruction

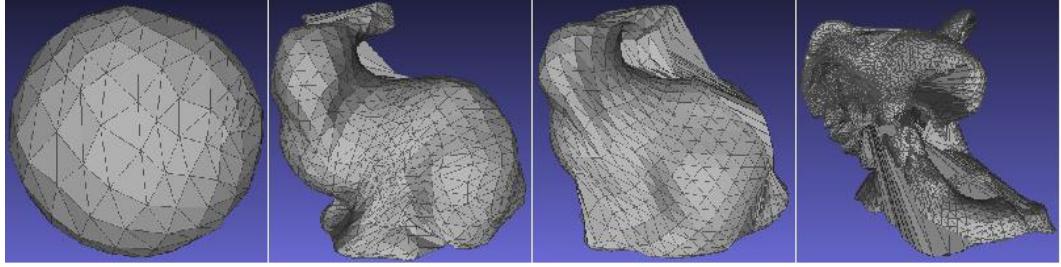


Fig. 15: Multiple Layer Surface Reconstruction

4.2 Performance Analysis of the proposed SOM-DDLib

This section describes the analysis and discussion in term of processing power of CPU and GPU for both single layer approach and the proposed SOM-DDLib multiple layers.

4.2.1 Performance Analysis using Single Layer and SOM-DDLib of Multiple Layers

Table 1 shows the performance of single layer SOM and Table 2 shows the performance of SOM-DDLib. The point cloud samples are ordered according to the number of points. From Table 1 and Table 2, it is found that SOM-DDLib is faster compare to the single layer SOM in most of the cases. When the data is small like sphere point cloud sample, point reduction process of the proposed SOM-DDLib is insignificant compare to the single layer SOM, which is much better in dealing with small datasets.

Table 1: Performance of Single Layer SOM

| Single Layer SOM | | | | |
|------------------|--------|---------|--------|--------|
| Model | Sphere | Bunny1 | Bunny2 | Eagle |
| Points | 422 | 8171 | 35947 | 796825 |
| Iteration | 1000 | 1000 | 1000 | 500 |
| MapX | 10 | 25 | 25 | 100 |
| MapY | 20 | 40 | 50 | 200 |
| Time (s) | 4.84 | 124.528 | 619.12 | 52560 |

For big point cloud datasets, the proposed SOM-DDLib is faster than single layer SOM and the difference becomes significant as the number of points increases. For instance, SOM-DDLib generates 2X times faster than single layer SOM approach for object Bunny1 and 12 times faster for eagle object. In SOM-DDLib, the initial iteration number for reduction of point cloud data is only 25 times compared to 500 times in single layer SOM. After the reduction, the input points of second layer in SOM-DDLib is 20000 (100 x 200) which contributes about

2.5% reduction from the original points. Thus, the performance of SOM-DLLib is better as the number of iterations increases in reconstructing the surfaces accordingly.

Table 2: Performance of Multiple Layer of SOM-DDLib

| Multilayer SOM | | | | |
|----------------|--------|--------|--------|--------|
| Model | Sphere | Bunny1 | Bunny2 | Eagle |
| Points | 422 | 8171 | 35947 | 796825 |
| Iteration | 1575 | 1575 | 1575 | 1575 |
| MapX | 20 | 40 | 40 | 100 |
| MapY | 40 | 60 | 60 | 200 |
| Time (s) | 10.08 | 60.192 | 75.21 | 4392 |

4.2.2 Performance Analysis between using CPU and GPU for SOM-DLLib

Table 3 shows the performance of SOM-DLLib using CPU only and Table 4 shows the performance of the proposed SOM-DLLib using GPU. From the results, GPU approach is faster than CPU in most of the cases but not for sphere object. Since GPU processing involves map computation and node updating, the map size directly affects the solutions. Sphere sample model has a map size resulting to 800 nodes (20x40) which is considered as very small. In this experiment, the GPU implementation consists of 1024 threads per block. Thus, 800 nodes cannot fully utilize the GPU processing power, hence slower the computation. Although, number of points cloud does not affect the performance, bigger map size is required to dig deeper information in the big point cloud datasets. Therefore, it is proven that SOM-DLLib, in-house tool development using cuda programming is better for big point cloud datasets for surface reconstruction.

Table 3: Performance of SOM-DDLib using CPU

| CPU | | | | |
|-----------|--------|--------|--------|--------|
| Model | Sphere | Bunny1 | Bunny2 | Eagle |
| Points | 422 | 8171 | 35947 | 796825 |
| Iteration | 1575 | 1575 | 1575 | 1575 |
| MapX | 20 | 40 | 40 | 100 |
| MapY | 40 | 60 | 60 | 200 |
| Time (s) | 4.30 | 64.70 | 87.32 | 9540 |

Table 4: Performance of SOM-DLLib using GPU

| GPU | | | | |
|-----------|--------|--------|--------|--------|
| Model | Sphere | Bunny1 | Bunny2 | Eagle |
| Points | 422 | 8171 | 35947 | 796825 |
| Iteration | 1575 | 1575 | 1575 | 1575 |
| MapX | 20 | 40 | 40 | 100 |
| MapY | 40 | 60 | 60 | 200 |
| Time (s) | 10.08 | 60.192 | 75.21 | 4392 |

5 Conclusion

This study aims to solve complex surface reconstruction problem by optimizing and reducing the big point cloud datasets. The integration of deep SOM learning concepts under GPU environment using CUDA programming is proposed to develop in-house deep learning library – SOM-DLLib for surface reconstruction. From the findings, it shows that the proposed SOM-DLLib gives significant performance dealing with big point cloud datasets. The performance differences become significant for the proposed SOM-DLLib as the input of big point cloud datasets increases. Though, it is understood that GPU will always give faster solutions, but the main focus of this research is on the in-house development of SOM deep learning algorithm using CUDA under GPU environment. Thus, it is worth to mention that the processing power performance is better when the mapping size is bigger using GPU, while for CPU, the processing is better when the mapping size is smaller.

ACKNOWLEDGEMENTS

The authors would like to thank Ministry of Higher Education (MOHE) and Universiti Teknologi Malaysia (UTM) for the funding, and NVIDIA Corporation for the hardware support in executing our R & D. This work is partially supported by the Fundamental Research Grant Scheme (FRGS: 4F786 & 4F802).

References

- [1] Chen, X. W. and Lin, X. (2014). Big Data Deep Learning: Challenges and Perspectives. *IEEE Access*. 2, 514–525.
- [2] Morell, V., Orts, S., Cazorla, M. and Garcia-Rodriguez, J. (2014). Geometric 3D point cloud compression. *Pattern Recognition Letters*. 50, 55–62.

- [3] Lim, S. P. and Haron, H. (2012). Surface reconstruction techniques: a review. *Artificial Intelligence Review*, 42(1), 59–78.
- [4] Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *TOG*, 32(3), 1–13.
- [5] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1), 1.
- [6] Hasan, S., Shamsuddin, S. M. and Lopes, N. (2014). Machine learning big data framework and analytics for big data problems. *Int. J. Advance Soft Compu. Appl.* 6(2).
- [7] Xiao, Y., Feng, R. B., Han, Z. F., & Leung, C. S. (2015). GPU Accelerated Self-Organizing Map for High Dimensional Data. *Neural Processing Letters*, 41(3), 341-355.
- [8] Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2015). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865-873.
- [9] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9), 1464-1480.
- [10] JUNIOR, A. D. M. B., NETO, A. A. D. D., & DE MELO, J. D. A self-organized neural network for 3D surface reconstruction.
- [11] Rego, R. L. M. E., Araújo, A. F. R., & de Lima Neto, F. B. (2010). Growing self-reconstruction maps. *IEEE transactions on neural networks*, 21(2), 211-223.
- [12] Lim, S. P., & Haron, H. (2013). Cube kohonen self-organizing map (CKSOM) model with new equations in organizing unstructured data. *IEEE transactions on neural networks and learning systems*, 24(9), 1414-1424.
- [13] Orts-Escalano, S., Garcia-Rodriguez, J., Moreli, V., Cazorla, M., & Garcia-Chamizo, J. M. (2014, July). 3d colour object reconstruction based on growing neural gas. In *Neural networks (IJCNN), 2014 international joint conference on* (pp. 1474-1481). IEEE.
- [14] Orts-Escalano, S., Garcia-Rodriguez, J., Morell, V., Cazorla, M., Perez, J. A. S., & Garcia-Garcia, A. (2016). 3D surface reconstruction of noisy point clouds using growing neural gas: 3D object/scene reconstruction. *Neural Processing Letters*, 43(2), 401-423.
- [15] Gao, H., Tang, J., & Wu, G. (2015, August). Parallel surface reconstruction on GPU. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service* (p. 54). ACM.

- [16] Xiao, Y., Feng, R. B., Han, Z. F., & Leung, C. S. (2015). GPU Accelerated Self-Organizing Map for High Dimensional Data. *Neural Processing Letters*, 41(3), 341-355.
- [17] Lopes, N., B. Ribeiro, and R. Quintas. *GPUMLib: a new library to combine machine learning algorithms with graphics processing units.* in *Hybrid Intelligent Systems (HIS), 2010 10th International Conference on.* 2010. IEEE.
- [18] Fadni Forkan & Siti Mariyam Shamsuddin, Kohonen – Swarm Algorithm for Unstructured Data in Surface Reconstruction, *Proceeding of IEEE Computer Society on the Fifth International Conference on Computer Graphics, Imaging and Visualization , 2008, pp: 5-11, DOI 10.1109/CGIV.2008.58.*
- [19] Hasan, S., Shamsuddin, S. M. and Lopes, N. (2015). Soft Computing Methods for big data problems. *GPU Computing & Its Applications.* Springer, pp:235-247.