Statistical Filtering on 3D Cloud Data Points on the CPU-GPU Platform

Statistical Filtering on 3D Cloud Data Points on the CPU-GPU Platform

Normi Abdul Hadi[1], Suhaila Abd Halim[2] and Norma Alias[3]

[1,2] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia,
[3] Ibnu Sina Sina Institute for Scientific and Industrial Research, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

normi@fskm.uitm.edu.my

Abstract. Recent advancement in scanning technologies has allowed an object to be represented in the 3D point cloud, which is an effective way to represent the overall view of the data and can be used for many purposes, such in manufacturing and visualization. However, the challenges in handling point cloud data are the noise and massive amount of data. Therefore, this study carries out a denoising process to remove the noise and reduce the size of data using statistical filtering. The process starts with neighboring points calculation using kNN. Then, the points are filtered using the statistical filtering method. This paper used 3D points of Armadillo and Stanford bunny retrieved from Point Clean Net database. To accelerate the performance of the distance calculation in kNN the process is executed on the CPU-GPU algorithm. The results show that the statistical filter has removed an amount of noise and preserved the features of the data. For the developed CPU-GPU platform, it is shown that the efficiency has accelerated the distance calculation process more than 700×.

1. Introduction

The 3D point cloud, a powerful representation of objects contributes to many research fields such as object recognition, human identification and virtual reality [1–2]. Point cloud presentation has wide application ranging from scanning small objects up to modeling a city [3]. This is because it can use low-cost devices with freely available algorithms [4]. The source of point clouds can be from 3D scanner, human-machine interaction and other devices exposed to the unnecessary environment. Therefore, the point cloud is inevitably suffering from noise contamination and outliers. Other factors that contribute to the noise is the restrictions of sensors, the defect of the device, the lighting or reflective nature of the studied object [1–5].

A strong method to remove the noise is the filtering method, an essential preprocessing phase. This phase is necessary to obtain accurate point clouds that are suitable for further processing such as modelling and reconstruction. The key purpose of filtering is to effectively eliminate the undesired data known as noise and preserve the important features of the object [1–5–6]. However, the challenge of each filtering method is either poor flexibility or high computational complexity [7]

Mushrooming methods have been developed in the literature for cloud points filtering. One of the existing methods is the bilateral filter, which builds a filter kernel directed by the point cloud. This method offers good feature-preserving but consumes high processing time [8]. Another method is using Mathematical morphology [9]. This method is implemented on LiDAR (Light Detection and

Ranging) data where the point is assigned to the corresponding grid in the plane coordinate. Other methods can be classified as statistical-based, neighborhood-based, projection-based, PDEs-based and signal processing based [1]. In addition, there is a hybrid filtering which combine at least two filtering method in order to get a better result. [10] employs three classical filters: low-pass, high-pass and band-pass filters to eliminate the noise in ECG signal. This study choose the statistical filtering method from [11] since it is suitable for the nature of the point cloud. This method removes sparse outliers by considering the neighbor distances (kNN) in the raw dataset.

A reason of the rapidly developed filtering method is the advancement of the scanning device. As the 3D machines are becoming more advanced, the size of the clouds is getting larger and larger [6] and the filtering process will be consuming approximately 60% to 80% of the processing time [9]. Consequently, this study took an initiative to execute the filtering process on the CPU-GPU platform with CUDA, which will be discussed in the next section. The mathematical background of the statistical filter is discussed in section 3. The method is applied on the Armadillo and Stanford bunny data and elaborated in section 4. Finally, this paper ends with conclusion.

2. The CPU-GPU Platform with CUDA

The Graphical Processing Unit (GPU) was first introduced as a graphic card for gaming purpose. Recently, the use of GPU has been expanded to assist high computing since it increases the computing performance [12]. The use of GPU is a must nowadays especially to support deep learning such as in image classification [13], character selection [14], job scheduling [15] and managing economics [16]. This technology can replace the current parallel method using multiple CPU memories such as in [17].

The use of GPU in computing has been applied in many areas, for example in modelling ice thickness [18], modelling EEG signal [19], generating 2D font [20], reconstructing 3D image of various data such as brain [21], and human head [22–23]. GPUs are composed of thousands of cores or threads. The data is transmitted to the GPU via the defined functions called kernels. These kernels run in parallel in the assigned cores. The threads are grouped in several blocks which form grids. The illustration of GPU is given in Figure 1.

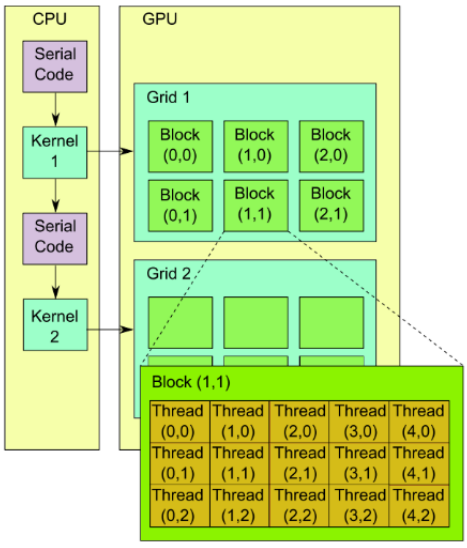


Figure 1. The GPU Illustration [24]

Based on Figure 1, the entire general process is done sequentially in CPU. However, at least one of the processes can be transferred to the GPU to be executed in parallel. This study used Compute Unified Device Architecture (CUDA) as its parallel programming platform.

3. Statistical Filtering

This study employs the statistical filtering method from [11] as shown in Figure 2.

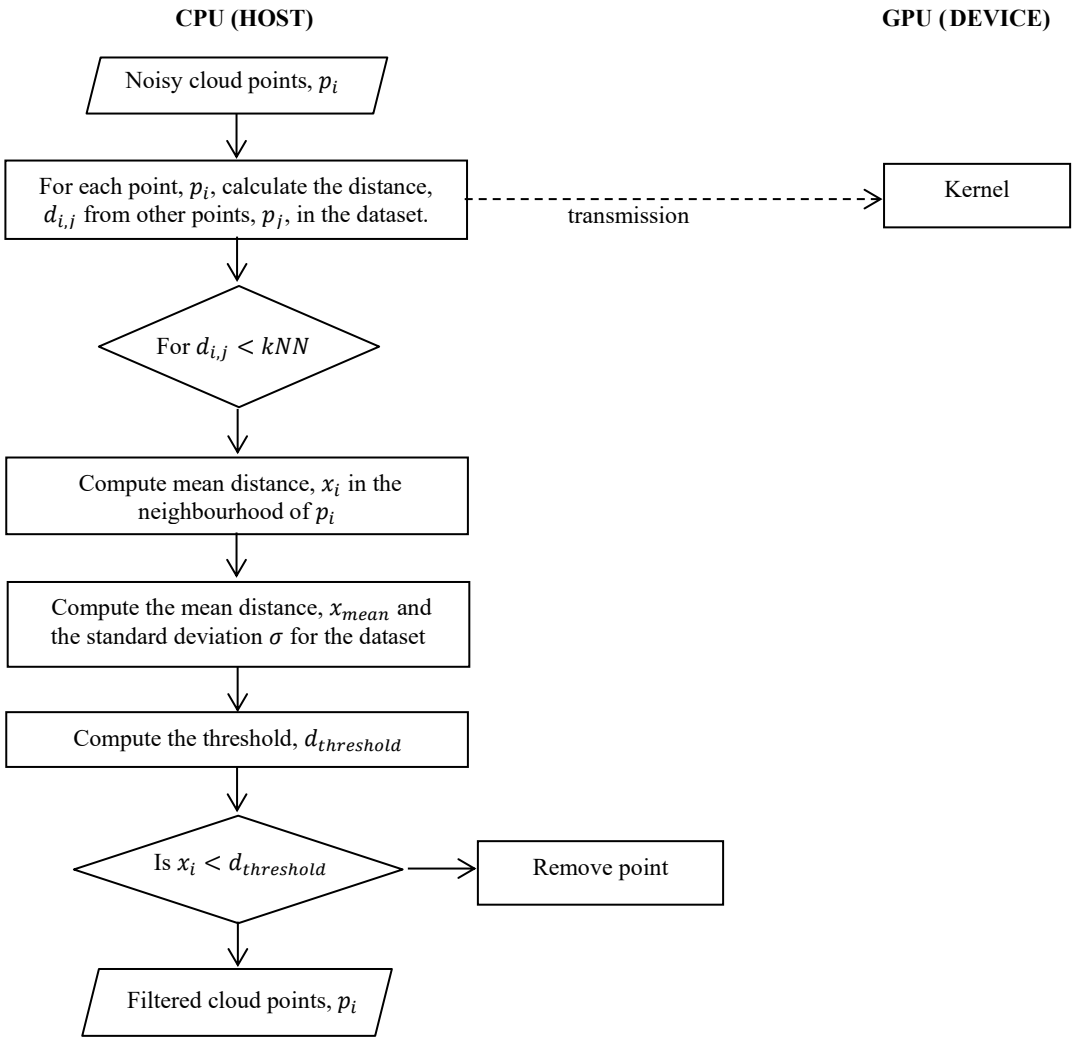


Figure 2. The statistical filtering process on the CPU GPU Platform

The original statistical filtering is shown in the CPU (HOST) column. However, since this study is considering a huge amount of data points, the distance calculation process is executed on the GPU as shown in the GPU (DEVICE) column. The transmission means the data transmission from the host to the device. The statistical filtering process starts by grouping the data points based on K-nearest neighbor (kNN) or neighbors within radius method [8]. In this study, three different values of k are tested which is 1,3 and 5.

In the next step, the mean distances for each neighborhood is calculated as $x_1, x_2, x_3 \ldots x_n$ where,

$$x_i = \frac{\sum_{j=1}^{k} d_j}{k} \, , (i = 1,2,3, \dots n) \tag{1}$$

After that, the mean distance for the dataset and the standard deviation are calculated as,

$$x_{mean} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_{mean})^2} \tag{2}$$

respectively. Finally, the threshold to decide whether to remove or retain the point is calculated as,

$$d_{threshold} = x_{mean} + std_{mul} * \sigma \tag{3}$$

In the threshold, std_{mul} represents the standard deviation multiple depending on the size of the cloud points. Points that having too high average distance where $x_i \geq d_{threshold}$ are considered as outliers and will be removed.

4. Results and Discussion

This study employed a CPU-GPU platform. The CPU computation is carried out using MATLAB R2019a software on a laptop equipped with NVIDIA GPU as given in Table 1.

Table 1. Hardware Specification

CPU	
Computer	Laptop Nitro AN515-52
Processor	with Intel (R) CORE (TM) i5-8300H (2.30GHz)
RAM	4.00 GB
OS	Windows 10 64-bit
GPU	
Name	GeForce GTX 1050
Compute Capability	6.1
Max Thread Block Size	[1024 1024 64]

Before further discussion on the filtering method, the efficiency of the GPU algorithm is presented. As shown in Figure 2 before, the distance calculation process is executed on the GPU. This is because, n data points require $n\left(\frac{n-1}{2}\right)$ number of processes for distance calculation. This study uses as many as 60000 data points which involves about 1.8 billion distance calculation process. This huge amount of data cannot be afforded by CPU-alone and need to be executed on the GPU. The comparison of processing time of CPU and GPU for a certain amount of data is presented in Figure 3.

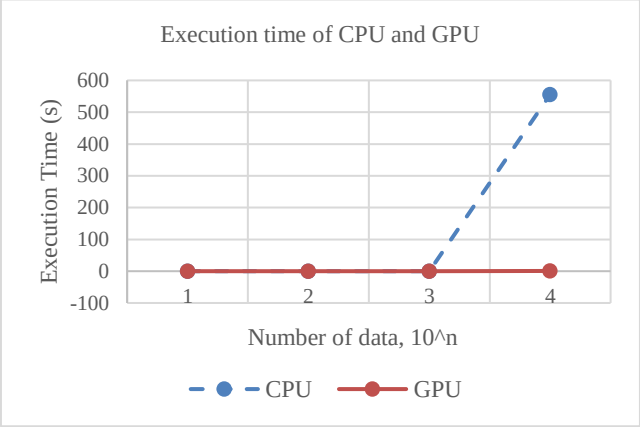
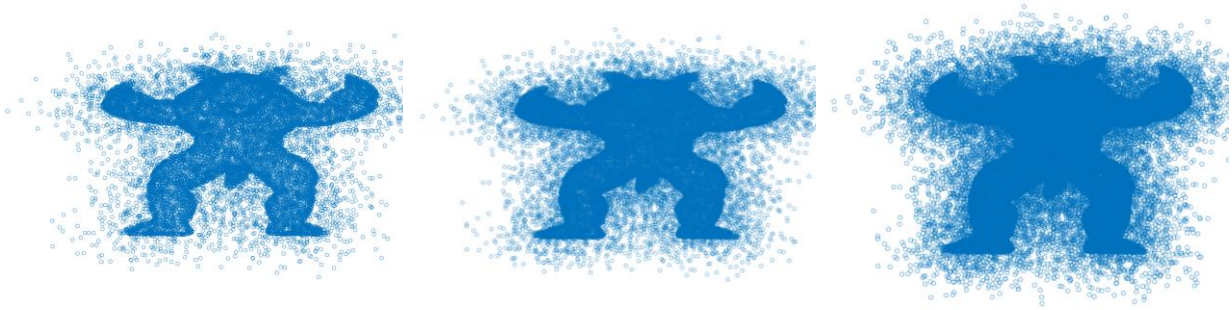


Figure 3. The execution time of CPU and GPU for different number of data points

In Figure 3, there are only 10^4 datapoints because CPU cannot compute more than that amount in this case. For 10^1 and 10^2, the GPU execution times are 1.7× higher than the CPU. This suggests that 10^1 and 10^2 number of data are not big enough to be executed on the GPU. Thus, the execution time is wasted for the data transmission instead of computation. The efficiency of GPU can be observed at 10^3 where the GPU is 6.7× faster than the CPU and accelerated to 732.9× faster for 10^4 number of data points. Therefore, the developed algorithm is suitable to be implemented for the statistical filtering.

The developed CPU-GPU algorithm is tested on armadillo and Stanford bunny dataset, two widely used dataset from Point Clean Net [25]. The datasets have been widely used in previous works and act as a benchmark of a successful reconstructed image. Both datasets are analysed based on three different data size which are 20 000, 40 000 and 60 000. This study focuses on filtering the Armadillo dataset using statistical filtering method. Three different amount of Armadillo dataset will be considered as given in Figure 4.



(a) 20000 points (b) 40000 points (c) 60000 points

Figure 4. The raw datapoints of Armadillo for 20000, 40000 and 60000 data points

From Figure 4, the shape of Armadillo can be seen as low as 20000 points. The difference with the higher data points is the intensity of the data including the noise. Higher number of data points will give a better accuracy but at the same time become noisier. For Stanford bunny, only 20000 datapoints will be considered, to confirm the performance of the filtering method. The raw datapoints of Stanford bunny is given in Figure 5.

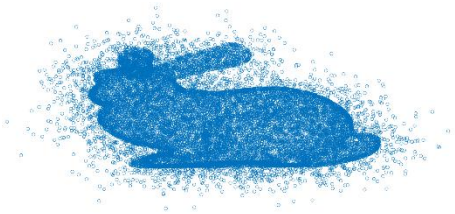


Figure 5. The raw datapoints of Stanford bunny with 20000 points

The filtering process requires two parameters to be set manually: kNN and std_{mul}. This study analyses the effect of the parameters by fixing the value of kNN and varying the value of std_{mul}. Note that, this study has tested higher values of kNN and std_{mul} and got a noisier result, and lower values have eliminated more false noise. As reference, this study set the value of kNN as,

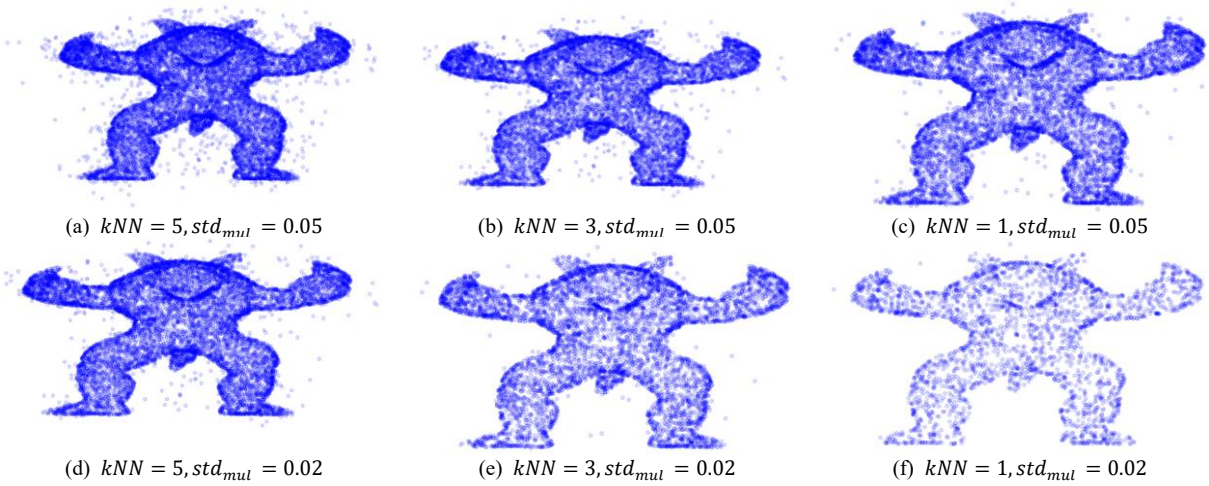
$$kNN \leq \sqrt{x_{mean}} \tag{4}$$

Based on (4), the values of kNN and std_{mul} have been narrowed to $kNN = 1,3,5$ and $std_{mul} = 0.02,0.05$ for Armadillo. The results are shown in Table 2.

Table 2. Number of data after the filtering process for Armadillo

			$kNN = 5$			
std_{mul}		**0.05**			**0.02**	
Total data points	20000	40000	60000	20000	40000	60000
Total removed points	5105 (25.5%)	10170 (25.4%)	15356 (25.6%)	7568 (37.8%)	15017 (37.5%)	22734 (37.9%)
Total unremoved noise	2166 (14.5%)	4896 (19.6%)	10984 (24.6%)	1177 (9.5%)	4896 (19.6%)	8546 (22.9%)
Threshold value	4.0992	4.1098	4.1081	2.8298	2.8375	2.8350
			$kNN = 3$			
std_{mul}		**0.05**			**0.02**	
Total data points	20000	40000	60000	20000	40000	60000
Total removed points	7345 (36.7%)	14586 (36.5%)	22092 (36.8%)	12696 (63.5%)	25288 (63.2%)	38109 (63.5%)
Total unremoved noise	1251 (9.9%)	5033 (19.8%)	8740 (23.1%)	344 (4.7%)	2567 (17.4%)	4722 (21.6%)
Threshold value	2.9153	2.9211	2.9189	1.6145	1.6173	1.6143
			$kNN = 1$			
std_{mul}		**0.05**			**0.02**	
Total data points	20000	40000	60000	20000	40000	60000
Total removed points	9507 (47.5%)	18914 (47.3%)	28527 (47.5%)	16892 (84.5%)	33769 (84.4%)	50723 (84.5%)
Total unremoved noise	730 (7%)	3883 (18.4%)	6965 (22.1%)	84 (2.7%)	1010 (16.2%)	1932 (20.8%)
Threshold value	2.2417	2.2468	2.2475	0.9230	0.9251	0.9250

From the table, the threshold values are similar for the same kNN and std_{mul} value, although the number of data points are increased. This is because the input data is unstructured. Although the number of data is increased by 20000, the positions of points are closed to the previous dataset. This situation has also affected the percentage of the removed points after the filtering process. Similarly, the percentages of removed data points are not significantly different for the same kNN and std_{mul}. Different values of kNN and std_{mul} gives the different accuracy of the produced image as shown in Figure 6.



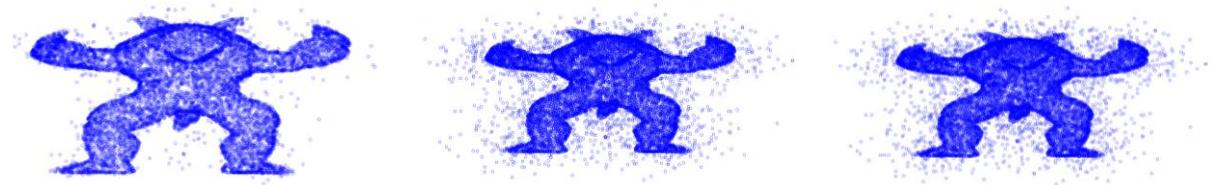
(a) $kNN = 5, std_{mul} = 0.05$ (b) $kNN = 3, std_{mul} = 0.05$ (c) $kNN = 1, std_{mul} = 0.05$

(d) $kNN = 5, std_{mul} = 0.02$ (e) $kNN = 3, std_{mul} = 0.02$ (f) $kNN = 1, std_{mul} = 0.02$

Figure 6. The filtered Armadillo dataset for 20000 points

From Figure 6, as the values of kNN and std_{mul} decreased, more noise has been removed. For instance, Figure 6(a) contains 14.5% of noise, but it is reduced to 2.7% in Figure 6(f). However, it is obvious that some of the real points (Armadillo body) are also eliminated. The eliminated real points can be called false noise. This study choses Figure 6(b) as the best filtered data where $kNN = 3$ and $std_{mul} = 0.05$. This value will be used for comparison with other amount of datapoints.

It can be observed that a part of the retained points still contains noise. The amount of this unremoved noise is increased with the increment of the number of points, but at the same time has improved the accuracy of the produced images. The comparison of the filtered data points for three different amounts of datasets of armadillo is shown in Figure 7.

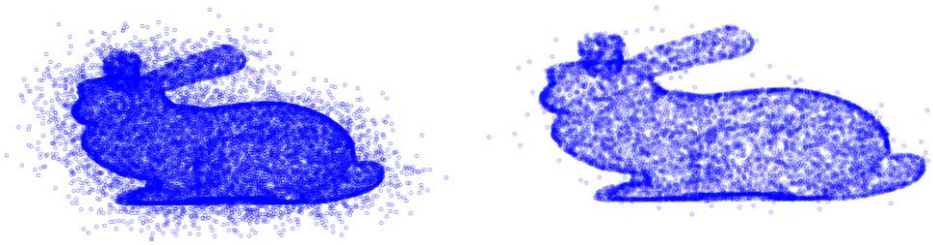


(a) 20000 points (9.9%) (b) 40000 points (19.8%) (c) 60000 points (23.1% noise)

Figure 7. The comparison of filtered data points with unremoved noise for three difference number of points

Based on Figure 7, higher number of data points will produce noisier image, but higher intensity of the real points will lead to the more accurate data interpretation.

As mentioned before, this study decides the value of kNN based on (4). Therefore, for the Stanford bunny dataset, the values of $kNN$and std_{mul} will not be the same as Armadillo as shown in Figure 8.



(a) $kNN = 0.5, std_{mul} = 0.02$ (b) $kNN = 0.1, std_{mul} = 0.02$

Figure 8. The filtered Stanford bunny dataset for 20000 points

Similarly, to Armadillo, a smaller value of kNN has reduced a large amount of noise. However, the value cannot be called as the best value since a lot of false noise has also been eliminated as shown in Figure 8(b).

5. Conclusions

Statistical filter is one of the reliable techniques for 3D cloud points filtering. However, the challenge in handling point clouds is the huge amount of the data. Thus, this paper has developed an algorithm to execute the filtering process on the CPU-GPU platform. The results show that the statistical filtering method can remove some of the noise but not totally cleanly. From the analyzed data, the largest leftover noise is 24.6% of the unremoved points. Therefore, it is suggested for future direction that, multiple-layer filtering methods to be used to ensure the accuracy of the result. In terms of the processing time, more steps will be executed on the GPU platform, and multiple GPUs will be employed to increase the efficiency of the algorithm. Despite all of the future improvement, this study has shown that statistical filtering of point cloud is suitable to implement the CPU-GPU platform. This method is a cost-effective method in terms of computation time and memory.

Acknowledgement

This research is supported by FRGS-RACER (600-IRMI/FRGS-RACER 5/3 (053/2019)), Ministry of Higher Education and Universiti Teknologi MARA Malaysia.

References

[1] Han X-F, Sheng Jin J, Jin J S, Wang M-J, Jiang W, Gao L and Xiao L 2017 A review of algorithms for filtering the 3D point cloud *Signal Process. Image Commun.* **57** 103–12
[2] El Sayed A R, El Chakik A, Alabboud H and Yassine A 2018 Efficient 3D point clouds classification for face detection using linear programming and data mining *Imaging Sci. J.* **66** 23–37
[3] Wolff K, Kim C, Zimmer H, Schroers C, Botsch M, Sorkine-Hornung O and Sorkine-Hornung A 2016 Point cloud noise and outlier removal for image-based 3D reconstruction *2016 Fourth International Conference on 3D Vision (3DV)* (IEEE) pp 118–27
[4] Almonacid J, Cintas C, Derieux C and Lewis M 2018 Point Cloud Denoising using Deep Learning *Congr. Argentino Ciencias la Inform. y Desarro. Investig. CACIDI 2018* 1–5
[5] Jia C C, Wang C J, Yang T, Fan B H and He F G 2018 A 3D Point Cloud Filtering Algorithm based on Surface Variation Factor Classification *Procedia Comput. Sci.* **154** 54–61
[6] Khatamian A and Arabnia H R 2016 Survey on 3D surface reconstruction *J. Inf. Process. Syst.* **12** 338–57
[7] Xiao Y, Wei Y, Yan C, Liu Y and Wang L 2019 A new data preprocessing method for 3D reconstruction of pavement *Int. J. Pavement Eng.* **0** 1–15
[8] Han X F, Jin J S, Wang M J and Jiang W 2018 Guided 3D point cloud filtering *Multimed.*

Tools Appl. **77** 17397–411
[9] Li S, Wang H, Ma Q and Zha X 2016 A Morphological LiDAR Points Cloud Filtering Method based on GPGPU *Proceedings ofthe 2nd International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM2016)* (SCITEPRESS –) pp 80–4
[10] Gusev M, Ristov S and Domazet E 2016 Optimizing High-Performance CUDA DSP Filter for ECG Signals *27th DAAAM International Symposium, pp.0623- 0632, B. Katalinic (Ed.), Published by DAAAM International* (Vienna, Austria: DAAAM) pp 623–0632
[11] Wang Z, Yang C, Ju Z, Li Z and Su C-Y 2017 Preprocessing and transmission for 3d point cloud data *International Conference on Intelligent Robotics and Applications* (Springer) pp 438–49
[12] Hernández-aguilar J A, Bonilla-robles J C, Díaz J C Z and Ochoa A 2019 Real-time video image processing through GPUs and CUDA and its future implementation in real problems in a Smart City *Int. J. Comb. Optim. Probl. Informatics* **10** 33–49
[13] Roslan R, Nazery N A, Jamil N and Hamzah R 2017 Color-based bird image classification using Support Vector Machine *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)* (IEEE) pp 1–5
[14] Abdullah N A S 2019 Expert System for Dota 2 Character Selection Using Rule-Based Technique *Advances in Visual Informatics: 6th International Visual Informatics Conference, IVIC 2019, Bangi, Malaysia, November 19–21, 2019, Proceedings* (Springer Nature) p 318
[15] Shuib A and Gran S S A 2018 Multi-objectives optimization model for flexible job shop scheduling problem (FJSSP) with machines' workload balancing *AIP Conference Proceedings* vol 1974 (AIP Publishing LLC) p 20106
[16] Abu Bakar N 2018 Managing economic and Islamic research in big data environment: from computer science perspective *J. Emerg. Econ. Islam. Res.* **6** 1–5
[17] Syawal M, Halim A, Hadi N A and Sulaiman H 2017 An Algorithm for Beta-Spline Surface Reconstruction from Multi Slice CT Scan Images using MATLAB pmode *IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)* (IEEE) pp 1–6
[18] Saidi M M and Alias N 2016 High Speed Computing of Ice Thickness Equation for Ice Sheet Model *J. Teknol.* **78** 143–9
[19] Alias N, Mohamad Mohsin H, Nadirah Mustaffa M, Hafilah Mohd Saimi S and Reyaz R 2016 Parallel Artificial Neural Network Approaches For Detecting The Behaviour Of Eye Movement Using Cuda Software On Heterogeneous CPU-GPU Systems *J. Teknol.* **78** 77–85
[20] Abdul Hadi N 2019 Digital Khat Calligraphy Using Beta-Spline Curve On CPU- GPU Platform *Kalam*
[21] Alias N, Said N M, Nur S, Khalid H, Sin D, Ching T and Ing P T 2008 High Performance Visualization of Human Tumor Growth Software *VECPAR* pp 591–9
[22] Hadi N A and Alias N 2019 3-Dimensional Human Head Reconstruction using Cubic spline surface on CPU-GPU Platform *International Conference Proceedings Series by ACM*
[23] Hadi N A 2018 Big Data Simulation for Surface Reconstruction on CPU-GPU Platform *J. Phys. Conf. Ser.*
[24] Bartezzaghi A, Cremonesi M, Parolini N and Perego U 2015 An explicit dynamics GPU structural solver for thin shell finite elements
[25] Ovsjanikov M PointCleanNet_data *Lab. d'informatique l'École Polytech.*
[26] Selwyn N and Jandric P 2020 Postdigital Living In The Age Of Covid *Postdigital Sci. Educ.*