Interactive VPL-based global illumination on the GPU using adaptive fuzzy clustering

Arnau Colom [1], Ricardo Marques [2,3], Luís Paulo Santos [4,5]

[1]Interactive Technologies Group (GTI), Pompeu Fabra University, Barcelona, Spain
[2]Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Barcelona, Spain
[3] Computer Vision Center, Cerdanyola (Barcelona), Spain
[4] Departamento de Informática, Universidade do Minho, Braga, Portuga
[5] HASLab, INESC TEC, Portugal

Abstract

Physically-based synthesis of high quality imagery results in a significant workload, which makes interactive rendering a very challenging task. Our approach to achieve such interactive frame rates while accurately simulating global illumination phenomena entails developing a Virtual Point Lights (VPL) ray tracer that runs entirely in the GPU. Our performance guarantees arise from clustering both shading points and VPLs and computing visibility only among clusters' representatives. Previous approaches to the same problem resort to K-means clustering, which requires the user to specify the number of clusters; a rather unintuitive requirement. We propose an innovative massively parallel, GPU-efficient, Quality-Threshold clustering algorithm, which requires the user to specify a quality parameter. The algorithm dynamically adjusts the number of clusters depending both on the specified quality threshold and on camera-geometry conditions during execution.

CCS Concepts

• ***Keyword*** $\rightarrow$ *Rendering; Ray Tracing; Global Illumination; Real-time Rendering; Parallel Programming;*

1. Introduction

Virtual Point Lights (VPL) based rendering solutions have been accelerated by leveraging local smoothness [OP11]. In general, these approaches cluster VPLs and/or Shading Points (SP) using some similarity metric. Visibility evaluation between SP and VPLs is then performed using each cluster representative point, rather than the original data. This results in a drastic reduction in the number of computations and, consequently, in rendering time. Several works have been published that aim to exploit local smoothness in indirect lighting to reduce computational complexity and rendering time while handling accurate global illumination light transport phenomena and interactive frame rates.

Usually the K-means clustering algorithm is used in this context and the user is forced to define upfront (i.e., before rendering) a fixed number of clusters. Selecting a suitable number of clusters is unintuitive since it depends on the desired expected quality, the geometry and illumination features of the scene, as well as the camera position and orientation. In this work, we resort to Quality-Threshold Clustering [HKY99], or QT-clustering for short, which does not require the number of used clusters to be specified. Instead, the user only supply a quality threshold which acts as a minimum quality guarantee and the number of clusters is automatically adjusted by the algorithm.

2. Related Work

Several works have been published aiming at achieving interactive VPL rendering solutions. Daqi et al. [TMKS20] propose a hierarchy method to reduce the number of evaluations needed to estimate the rendering integral. However, their method rely on a hybrid rasterization/ray tracing approach. Wang et al. [WWZ*09] presents a K-means clustering-based solution. SPs are clustered in the context of a caching/interpolation method similar to irradiance caching but applied to the final gathering stage of photon mapping. Final gathering is evaluated by hemispherical sampling at the centroids, and irradiance is interpolated for the remaining points. In [CMS22] SPs and VPLs are clustered using K-means, and have only 6 geometric attributes (3D positions and normals). The clustering runs entirely on the GPU and no data transfers between the CPU and the GPU are required since VPLs and SPs are generated and stored in the GPU. Visibility is then evaluated using each cluster representative.

3. Quality-Threshold Clustering

In QT-clustering the objective is to divide a dataset into multiple clusters. It takes as input the dataset D, the distance threshold τ, and the metric to use to quantify the distance between two data points. The quality of the clustering is reciprocal of the distance threshold τ. Given D, QT-clustering works by iteratively creating new clus-

DOI: 10.2312/egp.20241043

ters until no data-point remains to be clustered. At each iteration, all potential new clusters are evaluated taking into consideration the distance threshold, and the cluster with the largest number of elements is created. The data-points of the new cluster are marked as clustered, and the process is repeated until all the data is clustered. However, the canonical QT-clustering algorithm relies on a sequential process which makes it a challenging task to parallalize [DMV12]. We propose an efficient full-fledged GPU algorithm that allows for more than one cluster to be created at each iteration. To this end, we resort to coherent acceleration structures and efficient use of *CUDA* capabilities to maximize parallelism while minimizing overheads associated with shared data updates.

4. Results

To test the benefits of QT-clustering we set up one scene (Conference Room), rendered from two camera perspectives: a wide view where most of the scene objects are visible; and a close-up view. In both conditions, the input of both algorithms is fixed. For K-means the number of clusters is set to $K \approx 8000$ clusters and the Distance Threshold for QT-clustering $\tau = 0.016$. The number of clusters for K-means clustering has been based on the best trade-off configuration of the Conference Room in [CMS22]. The Distance Threshold parameter for QT-clustering has been determined so that the output matches the same number of clusters as K-means in the wide view configuration.

In Fig. 1 and Fig. 2 the data to be clustered is rather heterogeneous and the result of both clustering methods is relatively similar. The output for the QT-clustering algorithm results in a total number of 8722 clusters, a similar number as K-means. On the contrary, in Figs. 3 and 4, the points to be clustered exhibit similar geometry properties, since they are closer in the 3D space. The results show that QT-clustering is able to leverage this data coherence by creating bigger clusters where points share similar geometric attributes. On the contrary, in K-means the total number of centroids is defined a priori and the total number of clusters will remain constant during the application.

It is shown that the QT-clustering algorithm takes more computational time. However, in [CMS22], it's noted that the clustering step minimally impacts the overall rendering time. The more significant contributors are the estimation of visibility and the reconstruction of the image which are dependent on the number of clusters.

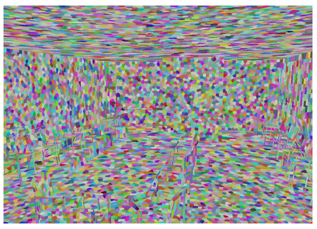
Figure 1: *Clustering from a wide-view camera setting using K-means (3ms).*

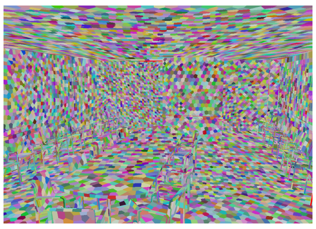
Figure 2: *Clustering from a wide-view camera setting using QT-clustering (7ms). Creates 8722 clusters.*

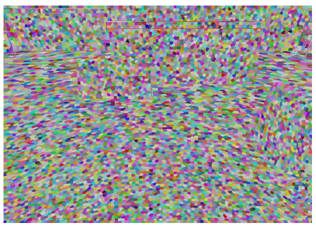
Figure 3: *Clustering from a close-view camera setting using K-means (19.9ms).*

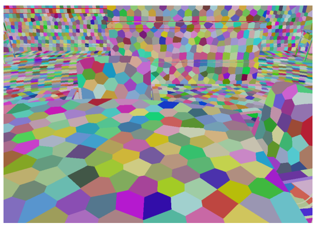
Figure 4: *Clustering from a close-view camera setting using QT-clustering (25.2ms). Creates 1669 clusters.*

5. Conclusions and Future Work

We propose a new GPU implementation for the QT-clustering algorithm that could provide faster rendering times while allowing a more intuitive setup for the user. The current results of this work in progress encourage pushing forward the research by evaluating the complete rendering time and image quality with previous works. Moreover, the flexibility that QT-clustering provides motivates the exploration of richer quality metrics that could result in better-distributed clusters.

6. Acknowledgements

This work was partially funded by the Research Consolidation Grant A-BMC (AEI-MICINN, CNS2022- 135480). Ricardo Marques acknowledges the support of the Serra Húnter Programme.

References

[CMS22] Colom A., Marques R., Santos L. P.: Interactive VPL-based global illumination on the GPU using fuzzy clustering. *Computers & Graphics 108* (2022), 74–85. URL: https://linkinghub.elsevier.com/retrieve/pii/S0097849322001753, doi:10.1016/j.cag.2022.09.008. 1, 2

[DMV12] Danalis A., McCurdy C., Vetter J. S.: Efficient quality threshold clustering for parallel architectures. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium* (2012), pp. 1068–1079. ISSN: 1530-2075. doi:10.1109/IPDPS.2012.99. 2

[HKY99] Heyer L. J., Kruglyak S., Yooseph S.: Exploring expression data: identification and analysis of coexpressed genes. *Genome research 9*, 11 (1999), 1106–1115. 1

[OP11] Ou J., Pellacini F.: LightSlice: matrix slice sampling for the many-lights problem. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011), ACM, pp. 1–8. URL: https://dl.acm.org/doi/10.1145/2024156.2024213, doi:10.1145/2024156.2024213. 1

[TMKS20] Tatzgern W., Mayr B., Kerbl B., Steinberger M.: Stochastic substitute trees for real-time global illumination. In *Symposium on Interactive 3D Graphics and Games* (2020), ACM, pp. 1–9. URL: https://dl.acm.org/doi/10.1145/3384382.3384521, doi:10.1145/3384382.3384521. 1

[WWZ*09] Wang R., Wang R., Zhou K., Pan M., Bao H.: An efficient GPU-based approach for interactive global illumination. *ACM Transactions on Graphics 28*, 3 (2009), 1–8. URL: https://dl.acm.org/doi/10.1145/1531326.1531397, doi:10.1145/1531326.1531397. 1