CEIG - Spanish Computer Graphics Conference (2024)
J. Marco and G. Patow (Editors)

An efficient point selection process over a meshlet-structured point cloud

Lidia M. Ortega [†1] and J. Carlos Fernández [1] and J. Antonio Collado[1] and J. Francisco R. Feito [1]

[1]Dto Informática, Universidad de Jaén, Spain

Abstract

Visualizing and interacting increasingly large and dense point clouds imposes the need for new methods with real-time results, where most common solutions imply a disadvantage greater than their benefit. Among the recent software and hardware advances in computer graphics and visualization, it is possible to take the concept of meshlet as a clustering of nearby points in space; this nature can bring a considerable improvement in the interaction process over the classical brute-force based algorithm, similar to common three-dimensional spatial structures. This work implements the point selection process over a meshlet-structured point cloud, assessing its performance against alternative methods and validating its correctness by visualizing the selection result on a graphical interface. By exploiting the meshlet instead of building additional spatial structures, the method's execution time can be optimized, as well as the use of the system's main memory.

CCS Concepts

• ***Computing methodologies*** → ***Mesh geometry models;*** *Computer graphics;*

1. Introducción

El manejo de nubes de puntos está cada vez más presente en múltiples disciplinas, que pueden tomar provecho de sus propiedades. Áreas como la robótica hacen uso de las características tridimensionales para el reconocimiento de objetos y su posición, mientras que soluciones más enfocadas a un entorno virtual logran reconstrucciones de escenarios próximos a la realidad. Por otro lado, las técnicas de teledetección en las que se acoplan diferentes sensores a drones, como pueden ser cámaras RGB o LiDAR (Light Detection and Ranging), tienen como resultado modelos 3D con cantidades ingentes de puntos.

Algo que comparten todas estas disciplinas en cuanto al uso de nubes de puntos recae en el nivel de detalle necesario. Cada vez se requieren nubes más densas y de mayor extensión con el fin de afrontar nuevos retos. Dado este aumento en la cantidad de información de entrada, los métodos clásicos para visualización e interacción con nubes de puntos no son suficientes.

Mediante el uso de estructuras de datos y métodos destinados específicamente a información espacial, gran parte de los requisitos más exigentes quedan suficientemente simplificados [Han18]. El objetivo que persiguen estas estructuras es reducir el número de operaciones de los algoritmos clásicos. Si bien suponen una mejora directa en términos de rendimiento, su granularidad debe ser controlada, pues tienen un impacto considerable en el uso de memoria.

Gracias a los avances hardware en dispositivos dedicados, especialmente por parte de la compañía Nvidia, aparecen conceptos novedosos de mayor potencial. Este es el caso del *meshlet* [Kub18], una agrupación de primitivas con las que puede trabajar la unidad de procesamiento gráfico (GPU) directamente a nivel de sombreador (*shader*), en el denominado *mesh shader* propio del hardware más reciente por parte de Nvidia®.

En la actualidad, el *meshlet* ha sido aplicado como mejora en procesos de visualización [UKPW21, NMSS22] y su optimización [MSS24], así como en tareas de reconstrucción gracias a sus propiedades [BGKS20]. Una de las aplicaciones más exitosas de este concepto viene de la mano del sistema *Nanite*, implementado en el motor gráfico Unreal Engine [Tat09]. Aprovechando los conjuntos de triángulos, aplican el descarte de aquellas partes de la malla fuera del campo de visión, y, adicionalmente, sustituyen los conjuntos más lejanos por otros de menor calidad, formando un sistema de nivel de detalle dinámico y casi imperceptible.

2. Objetivos

El método propuesto en este trabajo busca implementar el uso de la estructura de *meshlet* más allá del proceso de visualización. Con-

† Campus Las Lagunillas A3-136, Universidad de Jaén 23071 (Jaén) (lidia@ujaen.es)

DOI: 10.2312/ceig.20241144

siderando sus propiedades espaciales, puede emplearse como alternativa a las estructuras de datos comúnmente empleadas. En concreto, quedará implementado un sistema de selección sobre una nube de puntos estructurada internamente en *meshlets*. Esta interacción quedará resuelta mediante el lanzamiento de rayos, aplicando diversas optimizaciones en los procesos de comparación.

Hacer uso de esta estructuración supone ciertas mejoras frente a los métodos actuales:

- Mejoras visuales y de rendimiento al trabajar con nubes de puntos de gran tamaño y densidad, minimizando el tiempo de ejecución requerido al generar fotogramas, mientras se mantiene un nivel de detalle próximo a los datos crudos.
- Reducción del uso de memoria principal del sistema al no requerir estructuras de datos adicionales. La misma estructura empleada en la mejora visual queda reutilizada durante los procesos de interacción espacial.

3. Metodología

Nuestro método parte de un conjunto de datos obtenido de escenarios rurales mediante imágenes aéreas tomadas con drones y múltiples sensores espectrales, así como capturas de nubes de puntos LiDAR. Aquellas regiones de las cuales no se tienen escaneos LiDAR quedan resueltas con la generación de la nube de puntos correspondiente empleando procesos *SfM* (*Structure from Motion*).

Las nubes de puntos empleadas corresponden a fincas rurales dedicadas al cultivo, predominando la dedicación al olivar. Una propiedad considerable de estos datos, además de su densidad, es su distribución sobre el plano: la construcción de los *meshlets* toma la proximidad espacial de las primitivas, por tanto, su distribución influye directamente en la calidad de la estructura.

3.1. Construcción de *meshlets*

La mayor motivación en el uso de *meshlets* es el aprovechamiento de la agrupación de primitivas cercanas espacialmente, gracias a lo cual compartirán ciertas propiedades. Además, trabajos como [SKW21] exponen la utilidad de ordenar espacialmente las primitivas de cara a optimizar la eficiencia del proceso de dibujado, resultando en mejoras considerables de tiempo.

Partiendo de estos conceptos, la generación de *meshlets* parte de una nube de puntos cualquiera, que es ordenada espacialmente antes de mandar su información a la GPU. Por cada punto, se obtiene su índice relativo a una curva de Hilbert [Sag93], capaz de rellenar el espacio con un único trazo y con profundidad variable. Tal índice queda calculado a partir del código Morton del punto con 30 bits de longitud, expandiendo los bits que representan las tres coordenadas de su posición, como muestra la Figura 1 y el mapeo correspondiente a la curva de Hilbert [noa18].

Tras este preprocesamiento, los puntos de la nube quedan ordenados tal que el resto de la división de su índice con el tamaño de *meshlet* resultan en el *meshlet* que lo contiene. El resultado de aplicar la estructura queda visualizado en la Figura 2, mostrando la comparativa entre una nube de puntos con su color RGB asignado a cada punto y un color asignado a cada uno de los *meshlets*.

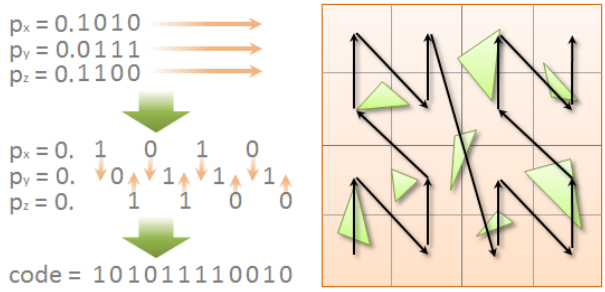

Figure 1: *Obtención del código Morton de elementos espaciales. Extraído de [Kar12]*

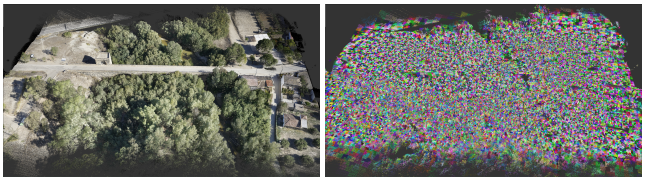

Figure 2: *Visualización de una nube de puntos en color RGB y por código de color según su meshlet.*

3.2. Interacción a nivel de *meshlet*

La interacción con objetos o entidades distribuidas en el espacio tridimensional queda resuelta mediante la comprobación de la trayectoria de un rayo sobre el espacio, tomando la posición y dirección según el punto de interés que seleccione el usuario en pantalla.

A la hora de aplicar los *meshlets* en este proceso, es posible considerar un subconjunto de los puntos que forman la nube. Empleando la caja envolvente alineada con los ejes de coordenadas (*Axis-Aligned Bounding Box* o *AABB*) y el rayo que traza la dirección de la selección, se determina la existencia de intersección entre ambos mediante el algoritmo propuesto por Andrew Woo en [Hai13]. Los datos de entrada requeridos por el algoritmo son extraídos del propio *meshlet*, tomando esa AABB que envuelve a todos los puntos, y la posición de la cámara virtual junto al punto de interacción indicado por el usuario, generando el origen y vector director del rayo.

Este vector, en proyección perspectiva, está formado como la diferencia entre la posición proyectada del puntero sobre el plano cercano del polígono de visión y la posición de la cámara virtual. En el caso de una perspectiva ortográfica, basta con utilizar la posición proyectada del puntero como origen y equiparar su dirección a la dirección de visión de la cámara virtual en ese instante. La Figura 3 presenta ambos casos de forma simplificada, especificando el rayo generado sobre un punto.

4. Resultados

La implementación se ha realizado en el lenguaje C++, utilizando OpenGL como librería de gráficos por computador e ImGUI como

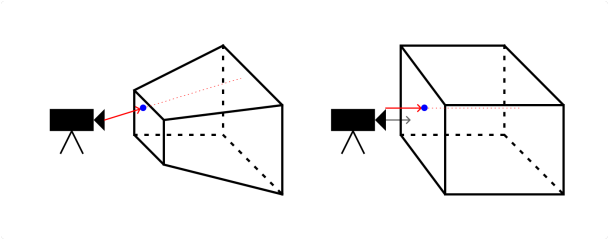

Figure 3: *Formación del rayo de selección para proyección perspectiva (izquierda) y ortográfica (derecha).*

librería destinada a la interfaz gráfica de usuario, mediante la cual queda visualizado el resultado.

La Figura 4 presenta el resultado de visualizar los *meshlets* que atraviesa el rayo de selección, mostrando la AABB de cada uno. Es posible apreciar, gracias al cúmulo de *meshlets* de menor tamaño, la trayectoria que realiza el rayo sobre la nube.

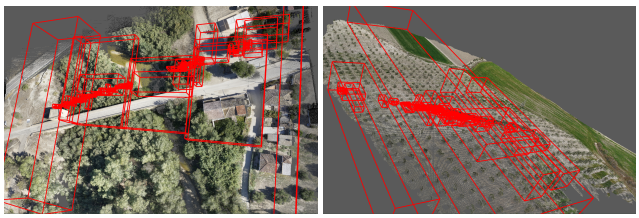

Figure 4: *Resultado de la selección de meshlets.*

A modo de comparativa, se ha realizado una selección de puntos sobre dos nubes de gran tamaño, empleando dos métodos: primero, mediante el uso de *meshlets* propuesto en este trabajo; segundo, tras la construcción de un *Octree* en tres dimensiones que englobe la totalidad de la nube de puntos. Dicho *Octree* es generado y gestionado por la librería externa *PCL* (*Point Cloud Library*) [RC11], resultando en una división jerárquica del área ocupada por la nube. La Tabla 1 presenta las características de ambas nubes de puntos tras generar las estructuras, resaltan el número y tamaño de los *meshlets*, junto a la cantidad de celdas y profundidad máxima del *Octree* asociado.

Nro. puntos	37.13 M	22.3 M
Número de *meshlets*	72 531	43 656
Tamaño de *meshlet*	512	512
Profundidad del *Octree*	8	10
N. nodos hoja del *Octree*	202 400	253 487

Table 1: *Comparativa de las propiedades de la estructura de meshlets y Octree de las nubes empleadas.*

Ambos métodos frente a ambas nubes se evalúan según tres atributos. El tiempo medio de selección representa la cantidad de tiempo necesaria desde que el usuario realiza la interacción hasta que la aplicación devuelve el resultado, dado en milisegundos. El uso de memoria principal indica la ocupación total de la aplicación en la memoria del sistema. Por último, el tiempo de construcción abarca todo el proceso de generación de cada estructura antes de poder ser utilizada. La Tabla 2 muestra los resultados medidos durante las pruebas realizadas.

Método	Meshlet		Octree	
Nro. puntos	37.13 M	22.3 M	37.13 M	22.3 M
T. selección	385 ms	236 ms	5 ms	2 ms
Memoria ppal.	1.55 GB	0.8 GB	2.54 GB	1.58 GB
T. construcción	647 ms	491 ms	3985 ms	2483 ms

Table 2: *Comparación de resultados del uso de meshlets frente al uso de Octree en tiempo de ejecución y construcción, y uso de memoria principal en dos nubes de puntos de distinta densidad.*

Comparando los resultados obtenidos, claramente el *Octree* presenta tiempos considerablemente mejores, pero con cierto impacto en el uso de memoria y tiempo de construcción. Considerando el enfoque hacia nubes de puntos cada vez más densas, el uso de estructuras de datos adicionales puede acaparar la mayoría del espacio en memoria principal y añadir un retardo importante mientras se construye la estructura. Por tanto, dependerá del problema a resolver la elección y evaluación de la estructura óptima. Con un enfoque general, los *meshlets* no suponen un alto impacto en la capacidad del hardware, y ofrecen mejoras visuales y de rendimiento, a la par que permiten su uso en procesos adicionales fuera del *pipeline* básico de renderizado.

Indagando en la comparativa anterior, cabe destacar los requisitos en memoria principal de los dos métodos. Según muestra la Figura 5, la ocupación de memoria atribuida al *Octree* se incrementa siguiendo una pendiente mucho más aguda frente a la estructura de *meshlets*. También resalta a favor de los *meshlets* la gran diferencia en tiempo de construcción, ya que éstos últimos se forman simplmente mediante agrupación de primitivas, a la par que mantienen un tiempo de selección competente. Ante esto, la opción de utilizar únicamente *meshlets* durante la interacción resulta óptima en casos con nubes mucho mayores en que las limitaciones sean más estrictas, a cambio de cierta pérdida de eficiencia en el lanzado de rayos para la selección.

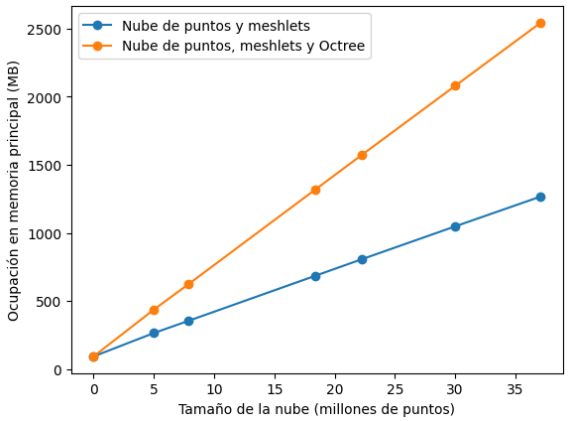

Figure 5: *Gráfica comparativa del uso de memoria principal antes y después de generar el Octree sobre la nube.*

5. Conclusiones

Frente al incesable aumento en tamaño y densidad de los datos de entrada para aplicaciones gráficas, aparecen soluciones y métodos con los que aliviar la carga de cómputo necesaria durante la visualización e interacción. El *meshlet* o clúster de primitivas espaciales ha demostrado una utilidad a tener en cuenta durante esos procesos, permitiendo la expansión hacia una visualización de mayor calidad y mejor rendimiento.

Este trabajo presenta una aplicación del *meshlet* más allá del proceso de visualización, incorporando sus características a la interacción con una nube de puntos. Partiendo de la caja envolvente alineada con los ejes (*AABB*) de cada *meshlet*, optimizamos el proceso de búsqueda del punto más cercano evaluando únicamente aquellos *meshlets* cuya AABB se encuentre en el camino del rayo de búsqueda arrojado.

Al comparar su rendimiento frente al uso de un *Octree*, encontramos diferencias en tiempo de ejecución que favorecen al *Octree*, pero con mejoras considerables en ocupación de memoria principal y tiempo de construcción por parte de los *meshlets*. Si bien no se han percibido mejoras en el tiempo de ejecución a favor de los *meshlets*, la reducción del uso de memoria principal puede suponer un gran aporte al enfocar este método sobre dispositivos con recursos limitados o al trabajar en escenas de tamaños mucho mayores. También facilita el tratamiento de nubes de puntos con información adicional sobre datos espectrales y su posterior gestión y análisis, habilitando un mayor rango de valores activos simultáneamente en la memoria principal del sistema, a la par que mantiene un tiempo de respuesta aceptable durante la selección.

En trabajos posteriores, queda planteada la posibilidad de extender aún más el uso del *meshlet* en otros procesos, hasta el punto de aprovechar esta estructura directamente en la unidad de procesamiento gráfico (GPU) fuera del renderizado, similar a lo que aplica [SGG*22]. Adicionalmente, las cualidades de proximidad espacial propias del *meshlet* pueden aportar mejoras durante la fusión de datos espectrales procedentes de fuentes externas a la nube de puntos, simplificando el proceso al tratar con agrupaciones de puntos próximos en el espacio que, generalmente, mostrarán un comportamiento similar. De este modo se puede reducir la escala de las nubes de puntos asociando a cada meshlet los valores medios de los valores espectrales o índices de vegetación.

Por último, otro aspecto a considerar en el futuro consiste en la selección de objetos de interés de la escena, como por ejemplo árboles individuales, empleando los *meshlets* que lo componen en lugar de aplicar un proceso de segmentación exhaustivo.

6. Agradecimientos

Este trabajo ha sido parcialmente financiada a través de los proyectos de investigación PID2021-126339OB-I00 y TED2021-132120B-I00, financiados por el Ministerio de Ciencia e Innovación.

References

[BGKS20] BADKI A., GALLO O., KAUTZ J., SEN P.: Meshlet priors for 3D mesh reconstruction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 2846 – 2855. Type: Conference paper. doi:10.1109/CVPR42600.2020.00292. 1

[Hai13] HAINES E.: GraphicsGems at master · erich666/GraphicsGems, May 2013. URL: https://github.com/erich666/GraphicsGems/blob/master/gems/RayBox.c. 2

[Han18] HAN S.: Towards efficient implementation of an octree for a large 3D point cloud. *Sensors (Switzerland) 18*, 12 (2018). Type: Article. doi:10.3390/s18124398. 1

[Kar12] KARRAS T.: Thinking Parallel, Part III: Tree Construction on the GPU, Dec. 2012. URL: https://developer.nvidia.com/blog/thinking-parallel-part-iii-tree-construction-gpu/. 2

[Kub18] KUBISCH C.: Introduction to Turing Mesh Shaders, Sept. 2018. URL: https://developer.nvidia.com/blog/introduction-turing-mesh-shaders/. 1

[MSS24] MLAKAR D., STEINBERGER M., SCHMALSTIEG D.: End-to-End Compressed Meshlet Rendering. *Computer Graphics Forum* (2024). Type: Article. doi:10.1111/cgf.15002. 1

[NMSS22] NEFF T., MUELLER J., STEINBERGER M., SCHMALSTIEG D.: Meshlets and How to Shade Them: A Study on Texture-Space Shading. *Computer Graphics Forum 41*, 2 (2022), 277 – 287. Type: Article. doi:10.1111/cgf.14474. 1

[noa18] 3D Hilbert curves in even fewer instructions – threadlocalmutex.com, Nov. 2018. URL: https://threadlocalmutex.com/?p=178. 2

[RC11] RUSU R. B., COUSINS S.: 3D is here: Point Cloud Library (PCL), 2011. URL: https://ieeexplore.ieee.org/abstract/document/5980567. 3

[Sag93] SAGAN H.: A three-dimensional hilbert curve. *International Journal of Mathematical Education in Science and Technology 24*, 4 (1993), 541–545. doi:10.1080/0020739930240405. 2

[SGG*22] SHAH H., GHADAI S., GAMDHA D., SCHUSTER A., THOMAS I., GREINER N., KRISHNAMURTHY A.: GPU-Accelerated Collision Analysis of Vehicles in a Point Cloud Environment. *IEEE Computer Graphics and Applications 42*, 5 (Sept. 2022), 37–50. Conference Name: IEEE Computer Graphics and Applications. URL: https://ieeexplore.ieee.org/abstract/document/9782096, doi:10.1109/MCG.2022.3177890. 4

[SKW21] SCHÜTZ M., KERBL B., WIMMER M.: Rendering Point Clouds with Compute Shaders and Vertex Order Optimization. *Computer Graphics Forum 40*, 4 (2021), 115–126. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14345. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14345, doi:10.1111/cgf.14345. 2

[Tat09] TATARCHUK N.: Advances in real-time rendering in 3D graphics and games I. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, Aug. 2009), SIGGRAPH '09, Association for Computing Machinery, p. 1. URL: https://doi.org/10.1145/1667239.1667243, doi:10.1145/1667239.1667243. 1

[UKPW21] UNTERGUGGENBERGER J., KERBL B., PERNSTEINER J., WIMMER M.: Conservative Meshlet Bounds for Robust Culling of Skinned Meshes. *Computer Graphics Forum 40*, 7 (2021), 57 – 69. Type: Article. doi:10.1111/cgf.14401. 1