Modelos de Remeshing

Jean Pierre Gabriel Sotomayor Cavero, Jose Leandro Machaca Soloaga,
Gonzalo Alonso Rodriguez Gutierrez
Universidad de Ingenieria y Tecnologia UTEC

***Resumen*—En este informe, se presenta la implementación y el analisis de dos modelos de remeshing basados en el cálculo de quadric errors y diagramas de voronoi, con el objetivo de simplificar mallas 3D.**

***Index Terms*—Remsehing, quadric errors, diagramas de voroni, vertices, mallas 3D, clusters, contracción de pares.**

I. Introducción

El remeshing es una técnica fundamental en el procesamiento de geometría tridimensional, cuyo objetivo principal es modificar o mejorar la estructura de una malla existente sin alterar significativamente su forma o características geométricas. Las mallas, estan compuestas por vértices, aristas y caras, se utilizan para representar superficies y volúmenes en una amplia variedad de aplicaciones como simulaciones físicas, gráficos por computadora, impresión 3D, entre otras. Es por ello que en este infrome se presentan dos modelos de remeshing que se basan en errores cuadricos y diagramas de voronoi.

II. Quadric Error Model

El modelo de Quadric Error se basa en la simplificación de polígonos a partir de la contracción iterativa de pares de vértices. Para cada vertice, se calcula un error aproximado al ser contraído utilizando matrices quadric. Esto permite que el algoritmo tenga una buena calidad, ya que las características de las mallas 3D se preservan incluso después de una gran simplificación.

II-A. Contracción de Pares

La contracción de pares de vértices consiste en reemplazar dos vértices cercanos por un solo vértice nuevo, eliminando las aristas y caras que los conectan directamente y ajustando la topología de la malla en consecuencia.

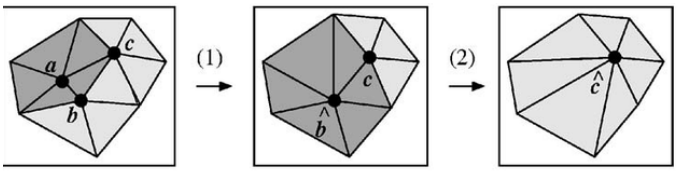

Figura 1. Contracción de Pares

Para ello, el algoritmo calcula inicialmente los costos de contracción de todas las aristas de la malla. Luego, selecciona la arista con el menor costo, extrae sus vértices y crea un nuevo vértice en una posición óptima, calculada a partir de las matrices quadric de ambos vértices.

II-B. Cálculo de Costos

Con el objetivo de calcular el costo de contracción, se deben recuperar las matrices quadric de los vértices involucrados. Estas matrices son definidas como matrices de tamaño 4×4. Una vez obtenidas las matrices quadric de ambos vértices, se suman para obtener la matriz que representa el error total que tendria el nuevo vertice respecto a todos los planos de ambos vertices originales. A partir de esta matriz combinada, se calcula la posición óptima del nuevo vértice que tiene como forma $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}^T$. Finalmente, se evalúa la forma cuadrática, lo que da como resultado el error asociado a esa posición óptima, expresado como: $\Delta(\mathbf{v}) = \mathbf{v}^T Q \mathbf{v}$

II-C. Cálculo de Posición Optima

Para calcular la posición óptima del nuevo vértice tras la contracción, se debe encontrar la posición que minimiza $\Delta(\mathbf{v})$. Para ello, se deriva $\Delta(\mathbf{v}) = \mathbf{v}^T Q \mathbf{v}$ con respecto al vector $\mathbf{v}$, es decir: $\frac{\partial(\mathbf{v}^T Q \mathbf{v})}{\partial \mathbf{v}}$. Igualando la derivada a cero para encontrar el mínimo, se obtiene el sistema lineal: $Q\mathbf{v} = 0$. Sin embargo, este sistema tiene infinitas soluciones, por lo que se impone la restricción de que el componente homogéneo $w = 1$. Esto permite obtener una solución única para la posición óptima del nuevo vértice. El sistema final tendra la siguiente estructura:

$$\begin{bmatrix} q_{11} & a_{12} & a_{13} & a_{14} \\ q_{12} & a_{22} & a_{23} & a_{24} \\ q_{13} & a_{23} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} * v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

En el algoritmo, se toma la matriz que representa el error total y se resuelve el sistema para obtener la posición óptima del nuevo vértice. En caso de que no sea posible encontrar una solución óptima, se utiliza como posición alternativa el punto medio entre los dos vértices a contraer.

II-D. Construcción de Quadric Matrix

La matriz quadric codifica la suma de los errores al desplazar un vértice respecto a los planos de las caras adyacentes.

Cada cara plana de la malla define un plano con ecuación:

$$ax + by + cz + d = 0$$

Este plano puede representarse como un vector columna:

$$p = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Y su **forma cuadrática** asociada es el producto exterior:

$$K_p = pp^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

La **matriz quadric** de un vértice es la suma de las matrices K_p de todos los planos que lo rodean:

$$Q = \sum K_p$$

En este caso, el algoritmo inicializa la matriz quadric con ceros. Luego, se itera sobre los pares de vértices vecinos del vértice que se está evaluando. Para cada par de vecinos, se calcula el plano del triángulo correspondiente, el cual se representa como: $p = \begin{bmatrix} a & b & c & d \end{bmatrix}$ donde a, b y c son los componentes de la normal del plano, y d es la distancia del plano al origen. Finalmente, se suma a la matriz quadric acumulada el resultado del producto del plano por su transpuesta, es decir, pp^T.

II-E. Algoritmo

En resumen, el algoritmo completo basado en quadric errors sigue los siguentes pasos:

1. **Inicialización**
   - Se inicializan estructuras de datos:
     - `Heap`: Un heap de prioridad para almacenar las aristas ordenadas por su costo de contracción.
     - `mesh`: La estructura que representa la malla, incluyendo vértices, caras y relaciones de vecindad.
2. **Entrada de Datos**
   - Se leen los vértices y caras desde un archivo de entrada.
   - Se construye la estructura de la malla:
     - Se almacenan las coordenadas de los vértices.
     - Se establecen los vecinos de cada vertice.
   - Se calculan los costos iniciales de contracción para todas las aristas y se almacenan en el heap.
3. **Cálculo de Costos de Contracción**
   - Para cada arista, se calcula el costo de contracción:
     - Se suman las matrices cuadráticas de los dos vértices de la arista.
     - Se calcula la posición óptima del nuevo vértice que minimiza el error cuadrático.
     - Se evalúa el error cuadrático en esa posición.
   - Este costo se utiliza para priorizar las aristas en el heap.
4. **Remeshing**
   - Se contraen aristas de menor costo iterativamente:
     - Se selecciona la arista con el menor costo del heap.
     - Se calcula la posición óptima del nuevo vértice.
     - Se eliminan los vértices de la arista y se actualizan los vecinos vecindad.
     - Se recalculan los costos de las aristas afectadas y se actualizan en el heap.
   - Este proceso se repite hasta alcanzar el número deseado de caras eliminadas.
5. **Salida de Datos**
   - Se escriben los vértices y caras restantes de la malla simplificada en un archivo de salida.
   - Los vértices eliminados durante el proceso no se incluyen en la salida.

III. Voronoi Model

En capítulos previos describimos el *Quadric Error Metric* (QEM) y la aproximación general de *vertex-clustering*. Sobre esa base, en esta sección diferenciamos cuatro variantes concretas que implementamos en el backend. Todas siguen el mismo *pipeline*, cargar OBJ, agrupar caras, recolocar vértices y reconstruir la malla, pero difieren en el criterio usado para escoger la posición del vértice representativo de cada clúster y, por tanto, en el tipo de malla resultante.

III-A. Malla Homogénea

Objetivo. Obtener una densidad prácticamente constante de triángulos, ideal para simulaciones donde se exige paso uniforme o para generar niveles de detalle (LOD) simplificados.

Criterio geométrico. Cada clúster se minimiza con respecto al *centroide* puro:

$$c = \frac{\sum_i A_i\, x_i}{\sum_i A_i}, \qquad E_{\text{uni}} = \sum_i \|x_i - c\|^2.$$

Al no ponderar la curvatura, las regiones planas y las curvas reciben la misma cantidad de elementos.

Ventaja. Algoritmo ligero y extremadamente rápido (una pasada suele bastar).

Limitación. Puede perder detalle en zonas de alta curvatura.

III-B. Malla Sensible a Curvatura

Objetivo. Concentrar vértices donde la geometría cambia más, preservando detalles finos con menos recursos.

Cálculo de curvatura. Para cada cara se ajusta un polinomio cuadrático y se obtiene la curvatura media H_i. Las caras con mayor H tienen más probabilidad de iniciar un clúster:

$$P_i = \frac{H_i}{\sum_j H_j}.$$

Efecto práctico. Las áreas planas se simplifican agresivamente, mientras que los bordes filosos o relieves reciben más triángulos. **Uso recomendado.** Modelos orgánicos o escultóricos con grandes regiones planas intercaladas con detalles.

III-C. Malla Tensorial

Objetivo. Alinear los triángulos con las *direcciones principales de curvatura*, reduciendo la distorsión en texturas direccionales o fibrados.

Tensor métrico K_i. Para cada cara se calcula el tensor K_i cuya base propia coincide con $(k_{\max}, k_{\min})$. El vértice ideal se obtiene resolviendo

$$\Big(\sum_i K_i\Big)\, c = \sum_i K_i\, x_i.$$

Resultado. La malla se "estira" de forma coherente con surcos, vetas o fibras del modelo. **Coste adicional.** Requiere invertir la matriz 3×3 por clúster, pero sigue siendo subsegundo para $\leqslant 1000$ clústeres.

III-D. Malla Cuádrica–Tensorial

Objetivo. Combinar la fidelidad geométrica del QEM expuesta en la Sección anterior con la direccionalidad tensorial descrita arriba.

Energía híbrida. Con un factor de mezcla λ,

$$E(c) = \sum_i \Big[(x_i - c)^\top K_i (x_i - c) + \lambda\, [c^\top\ 1]\, E_i \begin{bmatrix} c \\ 1 \end{bmatrix}\Big].$$

Para $\lambda = 0$ recuperamos la malla tensorial; para $\lambda \to \infty$, el algoritmo converge a puro QEM.

Ventaja. Ofrece el menor error RMS de todas las variantes sin sacrificar alineación. **Cuándo usarla.** Escenarios donde se requieren mallas ligeras *y* un control estricto del error, p. ej. compresión de activos 3D en videojuegos.

Cada esquema puede activarse desde la línea de comandos tal y como se detalla en la Tabla **??**. Los resultados cuantitativos (RMS error, tiempos, densidades) se discuten en la Sección de experimentación y resultados.

IV. Experimentación y Resultados

IV-A. Formato de Modelos 3D

Para la evaluación de los algoritmos de remallado, se seleccionaron tres modelos 3D ampliamente utilizados en la literatura por su complejidad geométrica y topológica: `armadillo`, `lucy` y `xyzrgb_dragon`. Estos modelos presentan una variedad de desafíos para los métodos de simplificación, incluyendo alta densidad de triángulos, regiones con alta curvatura, pliegues pronunciados y estructuras detalladas. Las características principales de estos modelos se resumen a continuación:

- **armadillo.obj** 49,990 vértices, 99,976 caras
- **lucy.obj** 49,987 vértices, 99,970 caras
- **xyzrgb_dragon.obj** 125,066 vértices, 249,882 caras

Estos modelos permiten analizar el comportamiento de los algoritmos ante diferentes tipos de geometría:

- **armadillo**: Figura humanoide con superficies suaves y algunas regiones de detalle fino.
- **lucy**: Escultura con gran número de pliegues, lo que introduce complejidad en zonas de alta curvatura.
- **xyzrgb_dragon**: Modelo de alta resolución con gran riqueza de detalles y topología compleja.

Figura 2. Modelo 3D `armadillo.obj`, utilizado como base para la evaluación. Presenta superficie suave con regiones anatómicas definidas.

Figura 3. Modelo 3D `lucy.obj`, escultura compleja con pliegues intensos y detalles topológicos finos.

Figura 4. Modelo 3D `xyzrgb_dragon.obj`, caracterizado por una alta densidad de triángulos y gran complejidad superficial.

La selección de estos modelos garantiza una evaluación robusta del desempeño de los algoritmos en contextos realistas, donde la precisión geométrica, la preservación de detalles y la eficiencia computacional son críticas.

IV-B. Quadric Error Model

Este método de simplificación de superficies es notable por su **eficiencia** y **rapidez**. La velocidad a la que opera el algoritmo depende de varios factores clave, que se pueden dividir en las fases de inicialización y simplificación, así como en las características del modelo y los parámetros elegidos:

1. **Complejidad del Modelo Original (Número de Caras)**: La velocidad está directamente relacionada con el tamaño del modelo de entrada. Modelos con mayor número de caras toman más tiempo en procesarse. Por ejemplo, simplificar el modelo `xyzrgb_dragon.obj` (125,066 vértices) toma considerablemente más tiempo que `armadillo.obj` (49,990 vértices) o `Lucy.obj` (49,987 vértices).
2. **Tiempo de Inicialización**: Incluye la selección de pares válidos, el cálculo de las matrices de error cuádricas Q para todos los vértices y la elección de objetivos de contracción. Esta fase depende del número total de vértices y conectividad del modelo.
3. **Tiempo de Simplificación**: Es la fase iterativa del algoritmo, donde se contraen pares de vértices seleccionados en orden creciente de error. Esta fase representa la mayor parte del tiempo total.
4. **Parámetro de Umbral** t: Este umbral permite emparejar vértices no adyacentes si su distancia es menor a t. Un valor alto mejora la calidad por agregación, pero puede generar muchos más pares posibles, incrementando la complejidad hacia $O(n^2)$ y, por tanto, el tiempo total.
5. **Eficiencia de la Representación Cuádrica**: Cada vértice almacena solo 10 valores de punto flotante, lo que permite cálculos eficientes al simplificar, gracias a la suma simple de matrices Q durante las contracciones.

Gráfico de rendimiento temporal:

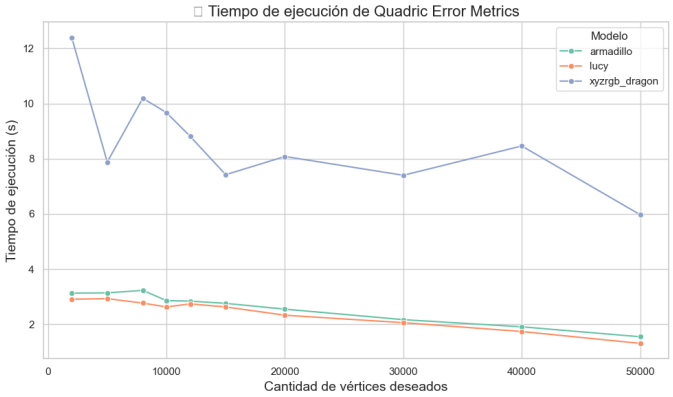

Figura 5. Tiempo total de remallado para distintos modelos y niveles de simplificación.

Análisis del gráfico:

Se observa una relación clara entre el número de vértices objetivo y el tiempo total de ejecución:

- Para los modelos `armadillo` y `lucy`, el tiempo de ejecución aumenta suavemente conforme disminuye el número de vértices. Esto indica una buena eficiencia del algoritmo en modelos medianos.
- En el caso de `xyzrgb_dragon`, el modelo más complejo, los tiempos son significativamente mayores, superando los 12 segundos cuando se remalla a solo 2,000 vértices. Esto confirma que el tamaño del modelo original influye directamente en la carga de cómputo.
- También se detecta una zona de saturación o amortiguación del tiempo cuando se baja de los 10,000 a los 2,000 vértices, especialmente en `armadillo` y `lucy`, donde el tiempo ya no sube tanto. Esto podría deberse a que la mayor parte del esfuerzo se realiza durante las contracciones iniciales.
- Finalmente, se nota que el rendimiento es aceptable incluso para simplificaciones agresivas, lo que valida la eficiencia del algoritmo en un amplio rango de resolución.

Figura 6. Resultado del remallado del modelo `armadillo` con el 20 % de los vértices originales.

Este comportamiento respalda las ventajas de este método frente a otros: combina buena calidad de resultados con tiempos de ejecución manejables, incluso en modelos con cientos de miles de vértices.

Visualización de Resultados:

A continuación se presentan los resultados del remallado utilizando el método Quadric Error, con una reducción al **20 % de los vértices originales**. Esta proporción corresponde aproximadamente a:

- `armadillo.obj`: 9,998 vértices (de 49,990)
- `lucy.obj`: 9,997 vértices (de 49,987)
- `xyzrgb_dragon.obj`: 25,013 vértices (de 125,066)

A pesar de esta reducción significativa, los modelos conservan su forma global, los contornos principales y muchos de sus detalles característicos. Esto evidencia la capacidad del método para priorizar la geometría esencial en el proceso de simplificación.

Incluso con una quinta parte de los datos, la representación sigue siendo visualmente coherente. En pruebas adicionales, se ha observado que reducciones hasta un **30–40 %** mantienen excelente fidelidad visual, lo que valida la efectividad de este enfoque para compresión de mallas en escenarios donde se busca balance entre rendimiento y calidad.

Figura 7. Resultado del remallado del modelo `lucy` con el 20 % de los vértices originales.

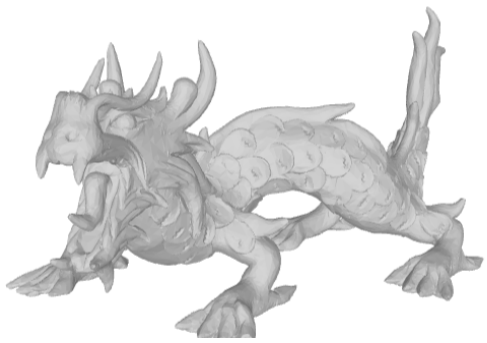

Figura 8. Resultado del remallado del modelo `xyzrgb_dragon` con el 20 % de los vértices originales.

IV-C. Modelo Voronoi

Este enfoque de remallado se basa en el uso de Diagramas de Voronoi generalizados sobre superficies 3D, con optimización basada en Lloyd y partición iterativa. A diferencia del modelo cuadrático tradicional, este procedimiento realiza múltiples ciclos de optimización geométrica, incorporando estructuras como grafos duales, métricas tensoriales, gradientes y mallas auxiliares. Su objetivo es generar una distribución precisa de los vértices, respetando un presupuesto determinado y adaptándose a la geometría local mediante el uso de métricas definidas.

Se analizaron cuatro variantes del modelo, cada una con una métrica específica:

1. Malla Homogénea: En este modo, los sitios de la partición Voronoi se distribuyen uniformemente sobre la superficie sin considerar la curvatura. Se implementa una métrica isótropa con densidad constante.

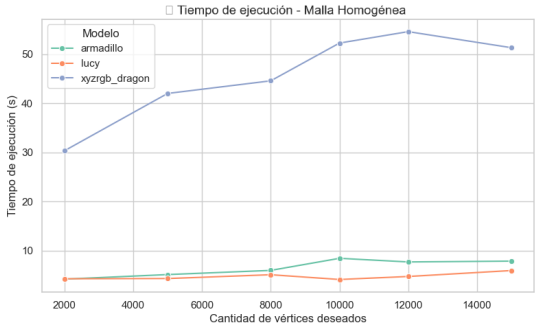

Figura 9. Tiempo de ejecución del remallado Voronoi en modo `Malla Homogénea`.

Análisis: Este modo destaca por su eficiencia computacional. Al no adaptar la densidad, las iteraciones de Lloyd convergen rápidamente, y la calidad de la malla es aceptable para superficies sin alta curvatura. Como se muestra en la Figura 9, modelos simples como `armadillo` y `lucy` son procesados en pocos segundos. El tiempo se mantiene relativamente estable incluso al aumentar el número de vértices deseados.

2. Malla Adaptive: Este modo introduce un campo de densidad basado en curvatura para adaptar la ubicación de los sitios Voronoi. Se usa una métrica isótropa con ponderación según la curvatura principal.

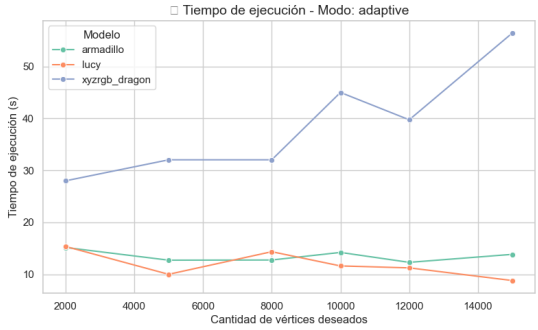

Figura 10. Tiempo de ejecución del remallado Voronoi en modo `Malla Adaptive`.

Análisis: El costo computacional es mayor que en el modo homogéneo, ya que se requiere el cálculo de campos de curvatura y una distribución de sitios no uniforme. La sensibilidad a la curvatura se controla mediante un parámetro de gradación γ, que permite ajustar el grado de adaptación. En modelos como `xyzrgb_dragon`, los tiempos de ejecución superan los 50 segundos. La adaptación mejora la fidelidad visual, pero a costa de una mayor complejidad geométrica en el proceso de optimización.

3. Malla Tensorial: Este modo emplea métricas anisotrópicas, utilizando tensores para alinear las celdas Voronoi según las direcciones principales de curvatura. Esto permite obtener mallas orientadas a la geometría subyacente.

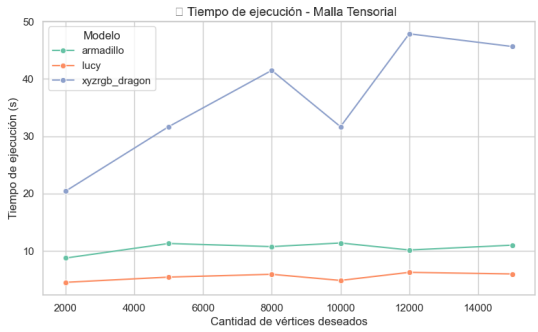

Figura 11. Tiempo de ejecución del remallado Voronoi en modo `Malla Tensorial`.

Análisis: La introducción de métricas anisotrópicas agrega complejidad al cálculo, ya que se deben estimar direcciones principales, tangentes y realizar transformaciones métricas locales. Estas métricas deforman el espacio para favorecer elementos elongados en ciertas direcciones, lo que permite capturar detalles estructurales importantes. El gráfico muestra un incremento sostenido del tiempo con el número de vértices, especialmente notable en modelos complejos.

4. Malla Cuádrica-Tensorial: Este modo combina métricas anisotrópicas con el uso de Métricas de Error Cuádricas (QEM), integrando estimaciones de error geométrico directamente en la fase de minimización.

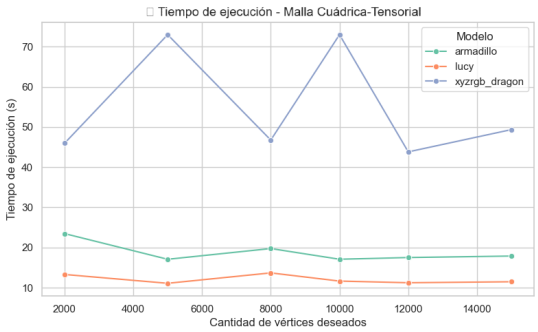

Figura 12. Tiempo de ejecución del remallado Voronoi en modo `Malla Cuádrica-Tensorial`.

Análisis: Este es el modo más sofisticado y costoso. Para cada clúster Voronoi se acumula una matriz QEM que permite reubicar el vértice óptimamente dentro del clúster, minimizando el error de aproximación de forma integrada en la energía. Esta estrategia evita post-procesamientos destructivos y mejora la alineación de vértices con características nítidas. No obstante, los múltiples pasos de evaluación y la actualización de QEM incrementan significativamente el tiempo de ejecución, alcanzando más de 70 segundos en `xyzrgb_dragon`.

Visualización de Resultados :

A continuación, se presentan los resultados del remallado Voronoi sobre el modelo `armadillo`, utilizando las cuatro variantes del método y manteniendo el 20 % del número original de vértices (de 49,990 a 9,998). Este experimento permite observar comparativamente el efecto de cada métrica sobre la forma final, bajo la misma compresión geométrica.

Figura 13. Modo `Malla Homogénea`: remallado uniforme sin adaptación a la curvatura.

Figura 14. Modo `Malla Adaptive`: adaptación a zonas de alta curvatura mediante campo de densidad.

Figura 15. Modo `Malla Tensorial`: orientación de las celdas Voronoi según curvaturas principales.

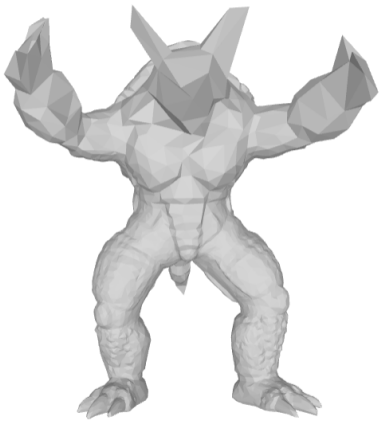

Figura 16. Modo `Malla Cuádrica-Tensorial`: integración de métricas QEM en el campo tensorial.

A simple vista, se aprecia que los modos adaptativos y

tensoriales logran preservar mejor los contornos principales. Sin embargo, todos los modos eliminan detalles finos especialmente visibles en el rostro y las extremidades debido a la baja proporción de vértices utilizada (20 % del total). Esto refuerza la recomendación de emplear presupuestos mayores (por ejemplo, 50 % o más) si se desea una mayor fidelidad en regiones críticas con alta curvatura o geometría detallada.

Conclusión General: El modelo Voronoi extendido es una herramienta poderosa para el remallado controlado de superficies 3D. A través de su formulación métrica y esquema iterativo eficiente, permite obtener mallas de alta calidad con un control fino del error y la orientación de los elementos. No obstante, los tiempos de ejecución varían ampliamente según el modo, lo que exige balancear precisión y costo computacional según el caso de uso. El modo `Malla Homogénea` es ideal para simplificaciones rápidas, mientras que `Malla Cuádrica-Tensorial` ofrece la mayor fidelidad a la forma original, siendo recomendable para aplicaciones donde la calidad geométrica es prioritaria.

V. Discusión

A lo largo de los experimentos, se han aplicado dos métodos distintos de simplificación y remallado: **Quadric Error Simplification** y **Voronoi Remeshing**. Ambos tienen enfoques y objetivos diferentes, lo que se refleja claramente en sus tiempos de ejecución, comportamiento según el modelo y escalabilidad frente a la reducción de vértices.

V-A. Desempeño general entre métodos

- **Quadric Error Simplification** es un algoritmo extremadamente eficiente, mostrando tiempos de ejecución inferiores a **4 segundos** en todos los modelos, incluso cuando se reducen hasta 2,000 vértices. Este comportamiento obedece a su implementación incremental basada en colapsos de aristas y uso de estructuras eficientes como heaps.
- Por otro lado, **Voronoi Remeshing** es un proceso más costoso, con tiempos que superan los **30 segundos** para el modelo `xyzrgb_dragon`, especialmente bajo modos como `adaptive` o `anisoquad`, que incorporan métricas de curvatura o distorsión anisotrópica.
- Se observa que mientras **Quadric Error** mantiene un comportamiento cercano a $O(n \log n)$, el método de **Voronoi** tiende a presentar una complejidad más elevada, posiblemente $O(n^{1,5})$ o incluso $O(n^2)$, en función de la complejidad del modelo y la métrica utilizada.
- Adicionalmente, la relación entre el número de vértices originales y los tiempos finales sugiere que **xyzrgb_dragon** —el modelo más complejo (125K vértices y 250K caras)— impone una carga significativa en los métodos basados en Voronoi, mientras que los modelos `armadillo` y `lucy` se procesan de forma más eficiente.

V-B. Comparación entre métodos en el modelo `armadillo`

Para una comparación representativa entre los métodos *Simplificación por Error Cuádrico* (QEM) y *Remallado Voronoi*, se utilizó únicamente el modo `Malla Adaptive` del segundo. Este modo permite una distribución no uniforme de vértices, concentrándolos en regiones de alta curvatura o detalle geométrico, lo que resulta en una malla más eficiente visualmente.

La gráfica muestra el tiempo de ejecución de ambos métodos sobre presupuestos de vértices comunes para el modelo `armadillo`. Como se observa, el método Voronoi en modo `Adaptive` presenta una mayor carga computacional frente a QEM, debido a su estrategia más compleja de asignación localizada de vértices. No obstante, esta inversión de tiempo puede justificarse cuando se busca preservar mejor la geometría en regiones críticas del modelo.

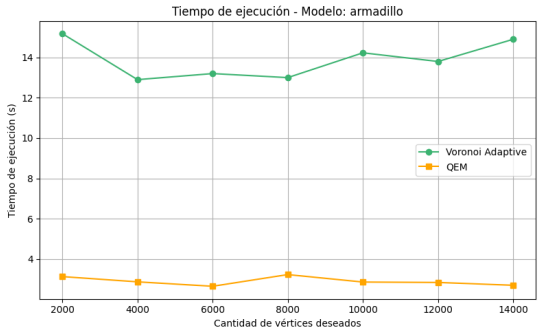

Figura 17. Comparación de tiempo de ejecución entre *Remallado Voronoi (Malla Adaptive)* y *Simplificación por Error Cuádrico* para el modelo `armadillo`.

Referencias

[1] M. Garland, P. S. Heckbert, *Surface Simplification Using Quadric Error Metrics*, Carnegie Mellon Univrsity, 1997.

[2] S. Valette, J. M. Chassery, R. Prost, *Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi Diagrams*, IEEE Transactions on Visualization and Computer Graphics, 2008.

[3] Repositorio de Github: https://github.com/Jeanpierrre/Computaci-n-Grfica–Remeshing.git