

Dos algoritmos para simplificación de mallas

Jairo Miguel Marulanda Giraldo
jmmarulang@eafit.edu.co

Samuel Arturo Flórez Rincón
saflorezr@eafit.edu.co

Kevin Mauricio Loaiza Arango
kmloaizaa@eafit.edu.co

Esteban Bernal Correa
ebernalc@eafit.edu.co

May 30, 2022

Keywords: *Algoritmos genéticos, Discretización de imágenes, Optimización multiobjetivo, NSGA-II, Simplificación de mallas*

1 Introducción

Los modelos poligonales de superficie son la representación de visualizaciones y simulaciones en 3D. Se obtienen mediante escáneres láser, sistemas de visión por ordenador o dispositivos de para modelar superficies de objetos muy detalladas. Estos modelos de superficie se utilizan en muchos ámbitos diferentes, como la visión por ordenador, los video juegos, el diseño asistido por ordenador, la medicina y la topografía Bernardini and Rushmeier (2002) Campomanes-Álvarez, Cordon, and Damas (2013) Sun, Paik, Koschan, and Abidi (2003).

Los modelos poligonales de superficie pueden llegar a tener miles o hasta millones de puntos y polígonos. Esto es un problema debido que pueden llegar a ser modelos muy pesados por lo que se vuelve necesario simplificar y reducir la cantidad de puntos de los modelos, conservando la forma, el volumen y los bordes tanto como sea posible Cignoni, Montani, and Scopigno (1998).

Se han propuesto diversas técnicas y acercamientos para llegar a cabo esta simplificación Cignoni et al. (1998), Heckbert and Garland (1997). Un acercamiento interesante es considerar esta simplificación como un problema de optimización multiobjetivo donde se busca minimizar la cantidad de puntos y minimizar el error o diferencia con el modelo original Campomanes-Álvarez et al. (2013). En particular el algoritmo genético de ordenamiento no dominado (NSGA-2) es una buena opción debido a su sencillez y baja complejidad algorítmica Deb, Pratap, Agarwal, and Meyarivan (2002).

También se muestra un segundo algoritmo, llamado edge collapsing. Este realiza una búsqueda entre todos los bordes para encontrar los vértices más cercanos entre sí. Ha llegado a ser incorporado en la configuración gráfica 3D de empresas de videojuegos como Bioware. Como complemento a este ultimo algoritmo, y a manera de ejercicio, se implementa un procedimiento de renderización propio.

Al final, se espera analizar y comparar los resultados de ambas implementaciones.

2 Contexto

2.1 Triangulación

Una triangulación de un dominio $B \subseteq \mathbb{R}^2$ es una partición de B en un conjunto de triángulos internos disjuntos T tal que cumple, con $t_i, t_j \in T$,

1. t_i y t_j son disjuntos o
2. $t_i \cap t_j$ es un vértice común o
3. $t_i \cap t_j$ es un borde común

Notese que una triangulación no permite que el vértice de un triángulo este en medio del lado de otro triángulo. Har-Peled (2011)

2.2 Buena forma y los criterios MinMax angle y MaxMin angle

Una característica generalmente deseable para una triangulación es que los triángulos tengan buena forma. Esto es, sean parecidos a triángulos equiláteros.

Luego, para un conjunto de puntos, será preferida la triangulación que posea el ángulo máximo más pequeño o, alternativamente, aquella que posea el ángulo mínimo más grande. Estos criterios son conocidos como MinMax angle y MaxMin angle respectivamente. Reparaz and Rodríguez (2014)

2.3 Triangulación de Delaunay

Una triangulación T de B es de Delaunay si, para todo triángulo, su circunferencia circunscrita no contiene ningún vértice que no sea del triángulo.

Esta triangulación es comúnmente utilizada en gráficos por computadora y tiene las siguientes propiedades, entre otras,

1. Si B es no degenerado, T es única
2. Es óptima respecto al criterio MaxMin angle.
3. Luego, evita triángulos largos y angostos
4. Es fácil de calcular

Cabe notar, esta triangulación es más sencilla de calcular en \mathbb{R}^2 . Es por estas características que se utiliza en nuestra implementación. Reparaz and Rodríguez (2014) Varshosaz, Helali, and Shojaei (2005)

2.4 Malla poligonal

Una malla poligonal M es una colección de vértices o puntos, bordes y caras que definen la forma de un objeto poliédrico.

Denotamos, para M_i, M_j mayas,

1. $A(M_i) \equiv$ área de $M_i \equiv$ la suma de las áreas de cada cara de M_i
2. $Len(M_i) \equiv$ la cantidad de puntos de M_i
3. $E(M_i, M_j) \equiv |A(M_i) - A(M_j)| \equiv$ el error entre M_i y M_j

Una malla que puede ser construida a través de una triangulación es llamada malla triangular. Están tienen la ventaja de usar la mínima cantidad de puntos necesaria para definir una cara. Weisstein (2005)

2.5 Algoritmo genético

En los años 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial, la de los algoritmos genéticos. Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular Sampson (1976).

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Los algoritmos genéticos se enmarcan dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética Goldberg (1989).

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se decide el reemplazo de los individuos para formar la nueva población, y como se cruzan, mutan y evalúan los individuos de cada generación. En general, consiste de los siguientes pasos:

1. **Inicialización:** Se genera aleatoriamente la población inicial, que está constituida por un conjunto de individuos los cuales representan las posibles soluciones del problema.
2. **Evaluación:** A cada uno de los individuos de esta población se le aplica una función de aptitud para saber que tan cerca está de lo que se quiere obtener.

3. **Condición de término:** Normalmente se usan dos criterios, correr el algoritmo un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:
 - (a) **Selección:** Después de saber la aptitud de cada individuo se procede a elegir los individuos que serán cruzados en la siguiente generación, que suelen ser los más aptos.
 - (b) **Cruce:** Busca combinar dos individuos (progenitores) para generar uno o más descendientes donde se combinan las características de ambos.
 - (c) **Mutación:** Modifica aleatoriamente parte de los individuos de la población, y permite alcanzar soluciones que no estaban cubiertas por los individuos de la población actual.
 - (d) **Reemplazo:** Se seleccionan individuos que conformaran la población de la siguiente generación.

2.6 Algoritmo NSGA-2

NSGA-2 es un algoritmo genético multiobjetivo, que busca aproximarse a una frontera de pareto, con las siguientes características

1. Usa el principio elitista
Esto es, los individuos con mejor desempeño en una población tienen la oportunidad de reproducirse.
2. Usa un principio de soluciones no-dominadas
La dominancia se determina según pareto y dependerá de cuales sean los objetivos.

NSGA-2 fue propuesto para resolver aspectos criticados del NSGA original, incluyendo la ausencia de elitismo. NSGA-2 es, además, mucho más eficiente que su predecesor, siendo altamente competitivo en convergencia al pareto.

Se realiza siguiendo los siguientes pasos

1. Se realiza un ordenamiento no dominado sobre la población combinada de padres e hijos y se clasifican en fronteras
Esto es, se ordenan según un nivel ascendente de no-dominancia.
La resolución del problema muestra como calcular fronteras para dos objetivos.
Cabe notar, en el algoritmo NSGA-2 clásico la cantidad de fronteras es fija para toda iteración del ciclo.
2. Se genera la nueva población según el ordenamiento anterior hasta alcanzar el tamaño deseado.

3. Si la cantidad conjunta de individuos en todos los frentes es mayor al tamaño deseado para la población, se hace uso de la crowding distance y se toman los individuos con mayor puntaje

La crowding distance es una medida relacionada con la densidad de la solución que ayuda a promover diversidad en la población.

La formulación utilizada se muestra en la resolución del problema para dos funciones objetivo.

4. Se genera la descendencia haciendo uso de última población generada

Fang, Wang, Tu, and Horstemeyer (2008) Flórez, Ocampo, and Cabrera (2008)

2.7 Algoritmo NSGA-2 modificado

Para la resolución de nuestro problema se realizan leves modificaciones al NSGA-2. Principalmente, en vez de manejar un número constante de fronteras, se permitirá variar el número de fronteras en cada iteración. Más específicamente, en cada iteración se calcularán fronteras hasta que contengan un número mayor de individuos que el deseado para el tamaño de la población, tras lo cual se parará el calculo y se calcularán las distancias de ser necesario.

Esta formulación disminuye la probabilidad de que sea necesario calcular las crowding distances. En el caso en que sea necesario, garantiza que únicamente deberán ser calculadas para los individuos de la ultima frontera generada.

2.8 Edge Collapse

Edge collapse es un algoritmo tradicional que usa ecuaciones de Euler para la reducción de puntos en una malla poligonal triangular, la forma en la que se reduce los puntos es de la siguiente manera, se toma un punto p y q, se colapsan creando un nuevo punto r con las conexiones de p y q, como resultado de esto, 2 triángulos se degeneran y eliminan del mesh.Floater (2002)

2.9 Edge Collapse modificado

Para esta solución realizamos un cambio a como funciona el algoritmo tradicional edge collapse. El cambio se baso en usar la distancia euclídea entre puntos para remplazar el uso de ecuaciones de Euler. Básicamente la distancia se usa para tomar la mitad entre 2 puntos y ahí poner el nuevo punto con los edges de los puntos anteriores.

3 Simplificación con NSGA-2

3.1 Planteamiento del problema a resolver

Se busca realizar una simplificación de mallas poligonales triangulares, haciendo uso del algoritmo NSGA-2 modificado. Para el NSGA-2 los individuos de cada población serán mallas poligonales generadas a partir de la malla original. A su vez, para cada individuo, llamaremos genes a que puntos de la malla original serán utilizados para generar el individuo (y/o, lo que es lo mismo, cuales serán descartados).

Los objetivos a minimizar serán el error y la cantidad de puntos de los individuos. .

Con el propósito de simplificar y hacer más eficiente el algoritmo, antes de realizar una triangulación para cualquier nube de puntos $B \in \mathbb{R}^3$ se realiza una proyección $B' \in \mathbb{R}^2$ para luego volver a la dimensión original. Para que esto sea posible, todas las mallas utilizadas son tales que puedan ser representadas por una función.

Las mallas utilizadas son dos de libre uso extraídas de yeggi. Las mallas son *face_2.obj* y *lowlaurana.obj*. Nota: las mallas mencionadas fueron procesadas.

3.1.1 Dominancia

Sean I_i, I_j individuos decimos que I_i domina a I_j , denotado por $I_i < I_j$, si se cumple

1. $(E(I_i, M_0) \leq E(I_j, M_0) \wedge \text{len}(I_i) \leq \text{len}(I_j)) \wedge$
2. $(E(I_i, M_0) < E(I_j, M_0) \vee \text{len}(I_i) < \text{len}(I_j))$

3.1.2 Crowding distance

Sea $I_i \in P$. La crowding distance de I_i usando como objetivos $\text{len}(I_i)$ y $E(I_i)$, denotada por $D(I_i)$, se calcula como sigue

1. $D(I_i) = 0$
2. Se organiza P respecto a la cantidad de puntos de sus individuos de manera ascendente. Supongamos, para mantener la notación sencilla, que I_i conserva su índice
3. Si $\text{len}(I_i)$ es máximo en la población, denotado por $\text{len}(\text{max})$, $D(I_i) \rightarrow \infty$
4. Si $\text{len}(I_i)$ es mínimo en la población, denotado por $\text{len}(\text{min})$, $D(I_i) \rightarrow \infty$
5. $D(I_i) = D(I_i) + \frac{\text{len}(I_{i+1}) - \text{len}(I_{i-1})}{\text{len}(\text{max}) - \text{len}(\text{min})}$
6. Si $E(I_i)$ es máximo en la población, denotado por $E(\text{max})$, $D(I_i) \rightarrow \infty$
7. Si $E(I_i)$ es mínimo en la población, denotado por $E(\text{min})$, $D(I_i) \rightarrow \infty$

8. Se organiza P respecto al error de sus individuos de manera ascendente. Supongamos, para mantener la notación sencilla, que I_i conserva su índice
9. $D(I_i) = D(I_i) + \frac{E(I_{i+1}) - E(I_{i-1})}{E(\max) - E(\min)}$

3.2 Resolución del problema

3.2.1 Inicialización

Sea

1. M^0 la maya original
2. $P = P^0$ la población, donde P^0 , de tamaño $2s$, es un conjunto de individuos generados aleatoriamente, donde s es el tamaño deseado para la población
3. $F = \emptyset$ el conjunto de fronteras
4. $m \in [0, 1]$ la tasa de mutación
5. $n \in \mathbb{R}$ el número máximo de iteraciones
6. $w = 1 \in \mathbb{R}$ la iteración actual

3.2.2 Calculo de fronteras

Realizamos

1. $F_k = \emptyset$
2. El primer individuo de P pasa a F_k y sale de P .
3. $\forall I_i \in P,$
 si $\exists I_j \in F_k$, tal que $I_j < I_i$, I_i permanece en P ,
 si $\exists I_j \in F_k$ tal que $I_i < I_j$, I_j pasa a P y sale de F_k ,
 si $\forall I_j \in F_k$ se cumple que no $I_j < I_i$, I_i pasa a F_k y pasa a P

Cabe aclarar que los individuos que regresen a la población en el transcurso del cálculo también deben ser comparados.

4. $F = F \cup \{F_k\}$

Repetimos los pasos anteriores, usando los individuos restantes en P , hasta que la suma de la cantidad individuos de las fronteras supere s .

3.2.3 Calculo crowding distance

Sea $F_k \in F$ la ultima frontera calculada. Extraemos de F_k los individuos con las distancias más pequeñas hasta que la suma de la cantidad individuos de las fronteras iguale s . Si esta condición se cumplía desde el inicio, no se hace nada y se continua al siguiente paso.

3.2.4 Selección de individuos más aptos

1. $P = \emptyset$
2. $P = F_1 \cup F_2 \cup \dots \cup F_k$, con $F_i \in F$, ($i = 1, 2.., k$)
3. $F = \emptyset$

3.2.5 Iteración

Si $w = n$ el algoritmo termina,
en caso contrario $w = w + 1$ y pasamos al siguiente paso.

3.2.6 Generación de descendencia

En la generación de descendencia se lleva a cabo el cruce de individuos y la mutación.
Estos dos procesos se explican a continuación.

Cruce

En el cruce se busca combinación de dos individuos para generar uno nuevo.

1. Se elige aleatoriamente dos individuos de P , estos serán los progenitores del nuevo individuo
2. Se genera un vector R de longitud $\text{len}(M^0)$, cuyos elementos son números aleatorios de una distribución uniforme entre cero y uno
3. Para generar el nuevo individuo se consideran los dos progenitores y el vector R , se dice que el i -ésimo gen del individuo hijo es igual al i -ésimo gen del primer progenitor si el elemento i -ésimo de R es menor que 0.5, en caso contrario es igual i -ésimo gen del segundo progenitor

Lo anterior se repite hasta obtener una cantidad de individuos hijos igual al tamaño de la población. Finalmente, se agregan los nuevos individuos a un conjunto P' , que solo contiene los hijos.

Mutación

1. Se genera un vector R de longitud $\text{len}(M^0)$, cuyos elementos son números aleatorios de una distribución uniforme entre cero y uno
2. $\forall I_i \in P'$,
si el i -ésimo elemento de R es mayor que m entonces el i -ésimo gen de I_i se mantiene igual,
en caso contrario, cambia a su complemento, es decir, si el i -ésimo punto de M^0 se mantenía en I_i ya no lo hará; si antes no se mantenía ahora lo hará

Notese que con esta forma de mutar se obtienen mejores resultados con tasas de mutación bajas, esto debido a que recorremos todos los genes del individuo y, por lo tanto, con tasas de mutación altas es posible alterar todos los genes del individuo y esto puede llegar a ser

contrapruedecente si el individuo resultara ser bastante apto.

Finalmente, $P = P \cup P'$ y volvemos al paso 4.2 (calculo de fronteras).

3.3 Resultados

Los parámetros usados fueron: tamaño de la población = 20, número de generaciones = 500, tasa de mutación = 0.01.

En los resultados se muestra un individuo representativo de las iteraciones 0, 100, 200, 300, 400 y 500.

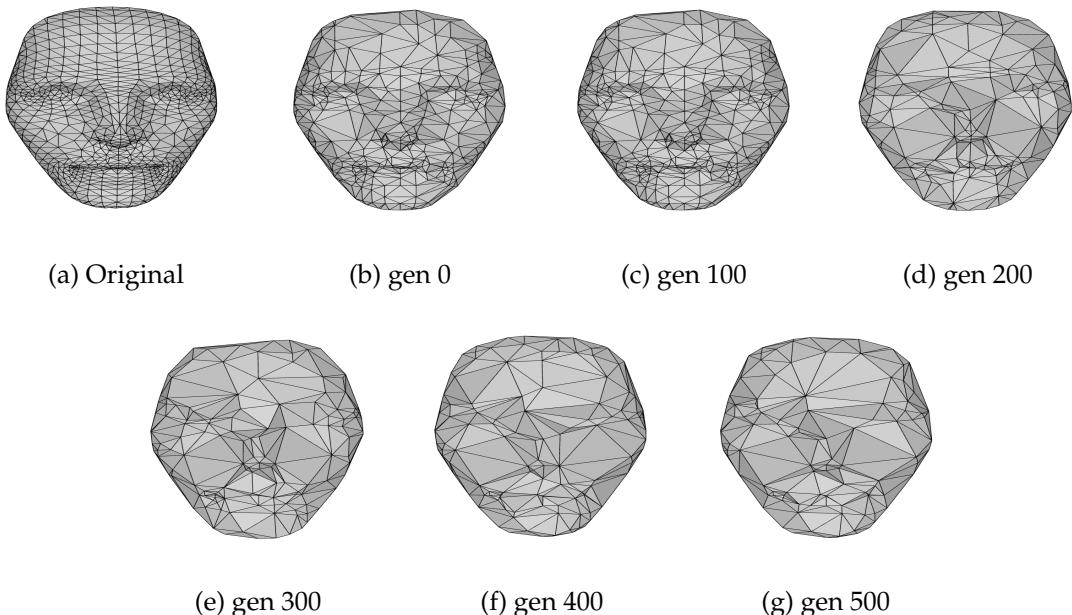


Figure 1: face_2 Visión frontal de un individuo de cada generación con $m = 0.01$

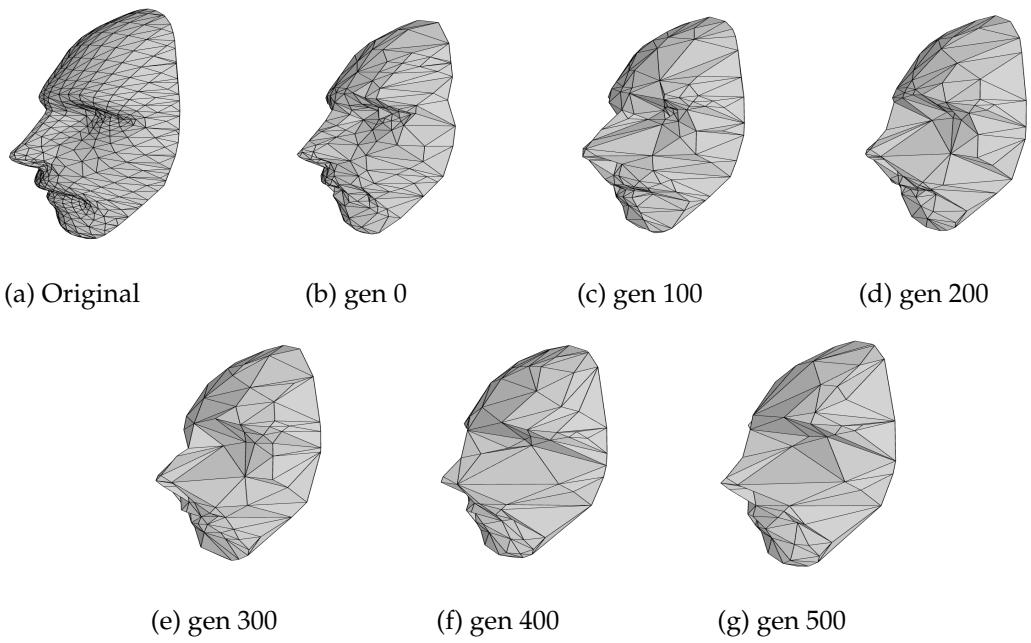


Figure 2: face_2 Visión lateral de un individuo de cada generación con $m = 0.01$

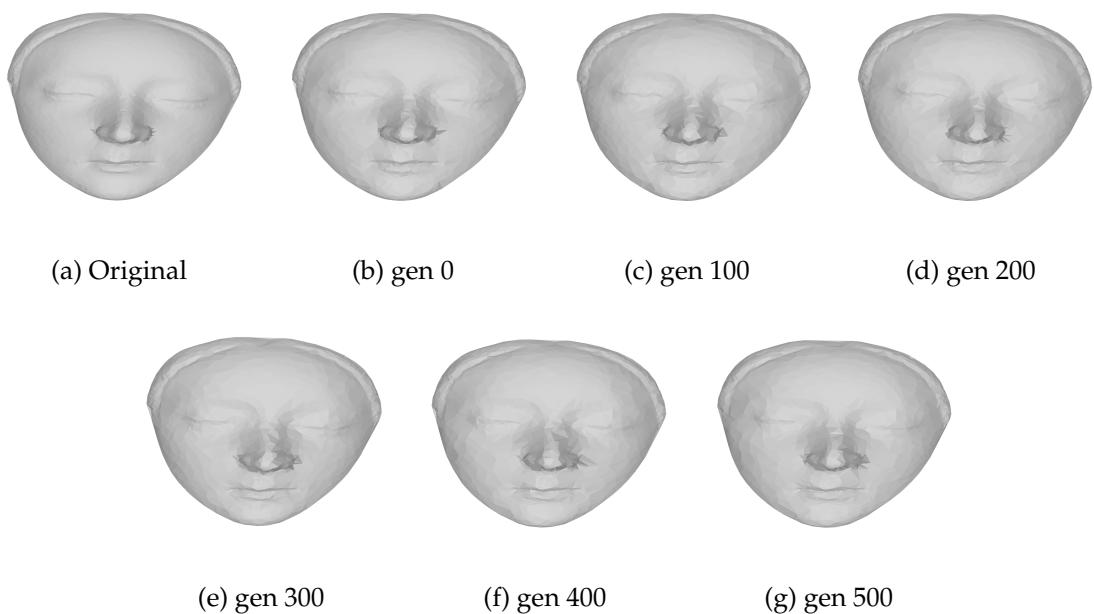


Figure 3: lowlaurana Visión frontal de un individuo de cada generación con $m = 0.01$

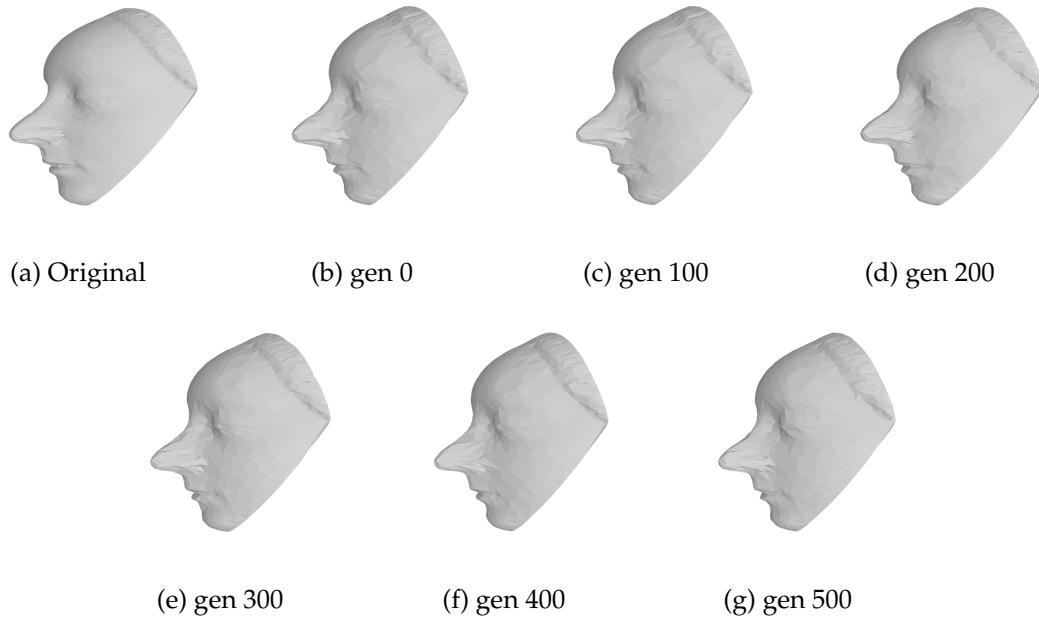


Figure 4: lowlaurana Visión lateral de un individuo de cada generación con $m = 0.01$

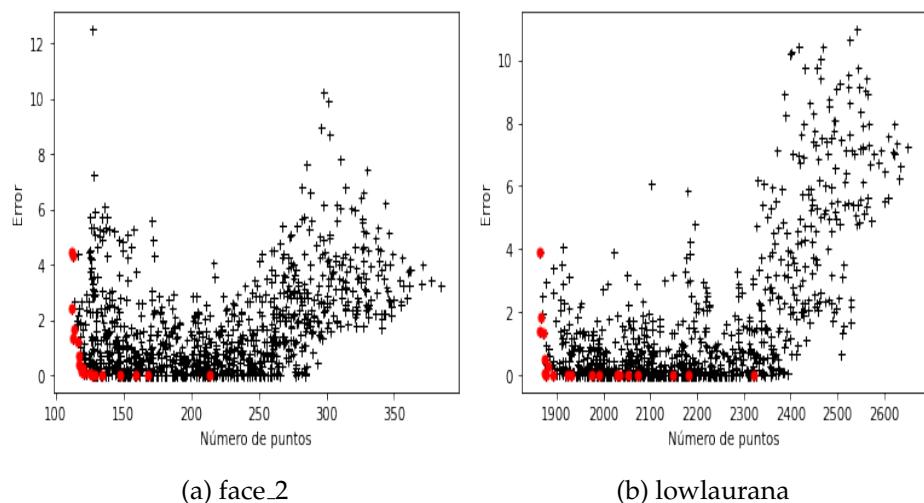


Figure 5: Todos los individuos de todas las generaciones, en rojo la ultima generación

face_2	# Puntos	Diferencia	lowlaurana	# Puntos	Diferencia
Original	697	0	Original	5190	0
Gen 0	326	5.673	Gen 0	2566	7.297
Gen 100	205	1.911	Gen 100	2147	1.374
Gen 200	159	1.541	Gen 200	2021	3.933
Gen 300	136	6.148	Gen 300	1972	0.390
Gen 400	126	2.355	Gen 400	1913	4.080
Gen 500	113	1.339	Gen 500	1864	1.417

Table 1: Número de puntos y error del individuo escogido de cada generación

4 Simplificación por edge collapse

4.1 Planteamiento del problema a resolver

Se busca realizar una simplificación de mallas poligonales triangulares, haciendo uso del algoritmo Edge Collapse. Para este algoritmo se le entrega la cantidad de puntos que se desean eliminar y este entrega una nueva malla.

El objetivo es reducir la cantidad de puntos de la malla original sin perder mucha información de esta.

Uno de los puntos a destacar de este algoritmo es que la renderización se implementó desde cero.

Las mallas utilizadas son dos de libre uso extraídas de yeggi. Las mallas son *face_2.obj* y *lowlaurana.obj*. Nota: las mallas mencionadas fueron procesadas.

4.2 Resolución del problema

4.2.1 Selección de vértices

Se seleccionan los vértices del borde V_i y V_j tal que la distancia entre ellos es mínima y $i < j$. Donde i, j son índices.

4.2.2 Colapso de vértices

1. Se encuentra el punto medio entre los vértices V_3 y se realiza $V_1 = V_3$
2. eliminamos V_j
3. eliminamos la conexión entre V_i y V_j
4. Agregamos a las conexiones de V_i las conexiones de V_j .

Los pasos anteriores se repiten hasta alcanzar el número de iteraciones deseado.

4.2.3 Inicialización del Render

Se inicializan todos los parámetros para poder trabajar con la interfaz de pygame. También definir la matriz de proyección y las matrices de rotación x,y y z. Inicializamos en 0s una matriz en la cual guardaremos todas las transformaciones hechas a los datos iniciales.

4.2.4 Proyección

Recorremos todos los vértices y los multiplicamos por las 4 matrices definidas al comienzo, este resultado lo guardamos en la matriz de transformaciones. También dibujamos el vértice con un círculo azul.

Después de esto, recorremos la lista de conexiones, y llevamos a cabo la conexión entre vértices. Para esta conexión, simplemente utilizamos los índices brindados por la lista

de conexiones y buscamos el vértice en la matriz de transformaciones para acceder a la posición actual de ambos vértices. Entre ambas posiciones dibujamos una línea.

4.3 Resultados

Para la malla lowlaurana se muestra la cara original, una reducción de 1000, 2000, 3000 y 4000 puntos.

Para la malla face_2 se muestra la cara original, una reducción de 100, 200, 300, 400 y 500 puntos.

Para ambas mallas, las figuras mostradas fueron generadas haciendo uso de nuestra implementación de la renderización.

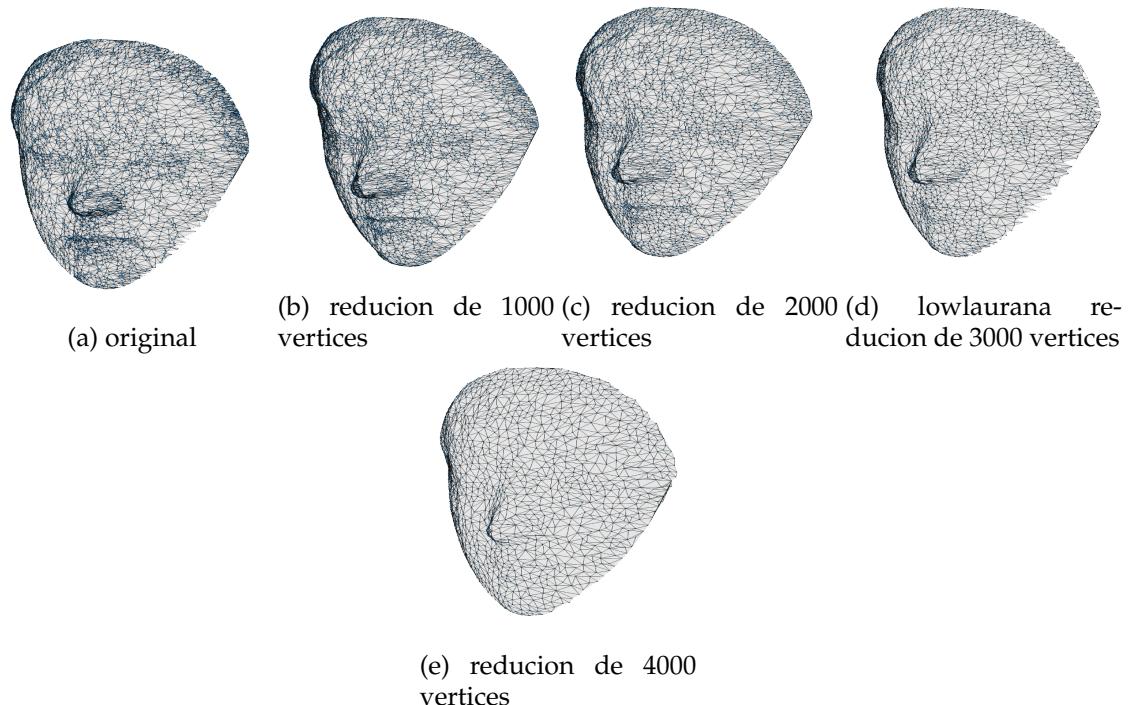


Figure 6: lowlaurana simplificación con Edge collapsing

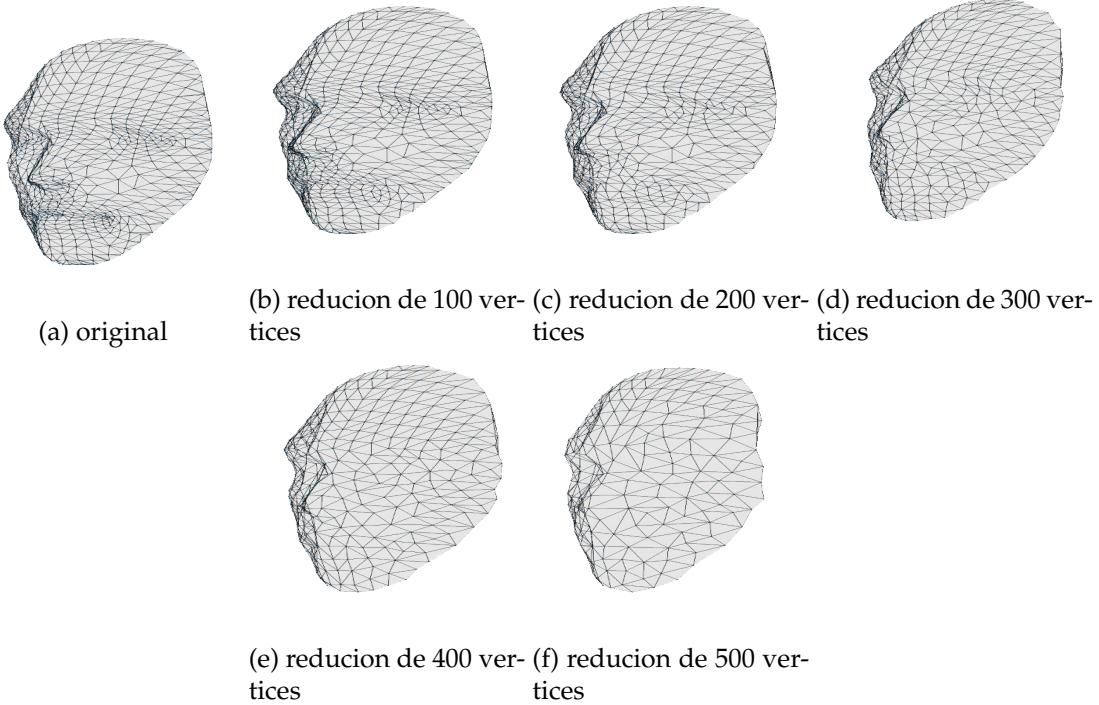


Figure 7: face_2 simplificación con Edge collapsing

5 Conclusiones

Luego de analizar los resultados obtenidos de ambos algoritmos, podemos ver como el egde collapse tuvo un mejores resultados manteniendo la integridad de la imagen luego de la simplificación de puntos, por lo que podemos decir que edge collapse es mejor opción para la simplificación de mallas triangulares, no obstante, el algoritmo NSGA-2 es una alternativa viable para realizar simplicación de mallas triangulares. El algoritmo fue capaz de llegar a simplificaciones de una alta cantidad de puntos. Sin embargo, la métrica utilizada para calcular el error se mostró poco recomendable para la mayoría de aplicaciones de las mallas triangulares, debido a que ignora características deseables de la simplificación como puede ser la simetría del modelo. Para trabajos futuros los resultados podrían verse beneficiados del uso de una métrica alternativa.

Se puede resaltar que edge collapse, aun siendo un algoritmo tradicional sigue mostrando resultados que destacan aún frente algoritmos más innovadores como NSGA-2.

References

- Bernardini, F., & Rushmeier, H. (2002). The 3d model acquisition pipeline. In *Computer graphics forum* (Vol. 21, pp. 149–172).
- Campomanes-Álvarez, B. R., Cordon, O., & Damas, S. (2013). Evolutionary multi-objective optimization for mesh simplification of 3d open models. *Integrated Computer-Aided Engineering*, 20(4), 375–390.

- Cignoni, P., Montani, C., & Scopigno, R. (1998). A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1), 37–54.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2), 182–197.
- Fang, H., Wang, Q., Tu, Y.-C., & Horstemeyer, M. F. (2008). An efficient non-dominated sorting method for evolutionary algorithms. *Evolutionary computation*, 16(3), 355–384.
- Floater, M. S. (2002). *Triangle meshes: Simplification and optimization*.
- Flórez, C. A. C., Ocampo, R. A. B., & Cabrera, A. M. (2008). Algoritmo multiobjetivo nsga ii aplicado al problema de la mochila. *Scientia et technica*, 2(39), 206–211.
- Goldberg, D. (1989). *Genetic algorithms in machine learning, search and optimization*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
- Har-Peled, S. (2011, 01). Geometric approximation algorithms.
doi: 10.1090/surv/173
- Heckbert, P. S., & Garland, M. (1997). Survey of polygonal surface simplification algorithms.
- Reparaz, M., & Rodríguez, N. (2014). Triangulaciones de delunay de alto orden en el terreno práctico de los sistemas de información geográfica. *Buenos Aires: Facultad de Ciencias Exactas y Naturales Departamento de Ciencias de la Computación*.
- Sampson, J. R. (1976). *Adaptation in natural and artificial systems (john h. holland)*. Society for Industrial and Applied Mathematics.
- Sun, Y., Paik, J., Koschan, A., & Abidi, M. (2003). Surface modeling using multi-view range and color images. *Integrated Computer-Aided Engineering*, 10(1), 37–50.
- Varshosaz, M., Helali, H., & Shojaei, D. (2005, 01). The methods of triangulation..
- Weisstein, E. W. (2005). Simplicial complex. *From MathWorld—A Wolfram Web Resource*.