Algoritmo Evolutivo Multi-Objetivo Basado en Simplificación de Mallas para el Modelado 3D

B. Rosario Campomanes-Álvarez[1], Sergio Damas[1], Óscar Cordón[2]

Resumen--**Los modelos de superficie poligonales son representaciones gráficas de objetos en 3D. La simplificación de mallas es el proceso para reducir el número de polígonos utilizados en un modelo 3D manteniendo la forma, el volumen y la topología de éste.**

En este trabajo, proponemos resolver este proceso de simplificación utilizando algoritmos evolutivos multi-objetivo, donde la calidad de una solución vendrá definida por dos objetivos contradictorios, la precisión del modelo 3D y el número de polígonos que componen la malla. El algoritmo Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) es el elegido para diseñar la propuesta multi-objetivo. Se han considerado dos operadores de recombinación para obtener una mayor diversidad de soluciones. Compararemos el método multi-objetivo con una propuesta mono-objetivo adaptada a nuestro problema para analizar las diferencias entre ambos. La comparativa de las variantes mono y multi-objetivo se realiza considerando una malla abierta de superficie facial.

I. INTRODUCCIÓN

Los modelos de superficie poligonales son representaciones de visualizaciones y simulaciones 3D. Se obtienen a partir de escáneres láser, sistemas de visión por computador o dispositivos médicos para modelar con mucho detalle superficies de objetos. Este tipo de modelos se utilizan en distintas áreas como visión y diseño guiados por ordenador, medicina, topografía, etc.

Una representación de la superficie de un objeto está formada por miles de polígonos. Éste es el principal problema que presentan. El tamaño tan grande de los ficheros produce tiempos de procesamiento muy altos. La obtención de un modelo reducido con un menor número de polígonos y por tanto, un tamaño de fichero menor pero de similar precisión, es un reto en este área. [9]

La simplificación de mallas es el proceso para reducir el número de polígonos -también denominados caras o triángulos- que modelan una superficie de un objeto manteniendo lo más posible su forma, volumen y límites.

Existen diversas técnicas para simplificar y optimizar una malla. Las técnicas clásicas de eliminación [11], [13], [22] y las de optimización a partir de una función de energía [18] son las más populares. También se han desarrollado otros trabajos más recientes para optimizar un modelo 3D que se basan en algoritmos evolutivos [9], [19], [26].

En general, no hay ningún trabajo dedicado a la simplificación de mallas basándose en algoritmos evolutivos multi-objetivo. Por ello, el enfoque del proceso de simplificación de mallas como un problema multi-objetivo constituye una interesante línea de investigación que se estudia en este trabajo. Para ello, hemos decidido utilizar el algoritmo NSGA-II propuesto en [7]. NSGA-II está considerado como uno de los métodos más importantes en optimización evolutiva multi-objetivo. Nuestra metodología, por tanto, se basa en que un algoritmo evolutivo multi-objetivo lleve a cabo la *simplificación de un modelo tridimensional abierto*. El problema consiste en determinar qué puntos del modelo utilizar para crear una malla que se aproxime con la mayor precisión posible a la original. Se considerará la utilización de dos objetivos contrapuestos por naturaleza como son la precisión y el número de triángulos que forman la malla. Hemos utilizado dos tipos diferentes de cruce: el cruce uniforme [23] y el cruce basado en vectores ortogonales [19].

Llevaremos a cabo una comparación entre el método que aquí se propone y el desarrollado por Huang y Ho en [19] que utiliza un algoritmo genético mono-objetivo para simplificar mallas faciales 3D en modelos abiertos. Además, el estudio experimental considerará varios parámetros como son: el tamaño de la población, el número de generaciones, la probabilidad de mutación y la de cruce.

Este trabajo se estructura como sigue: la sección 2 es una pequeña revisión dedicada a las técnicas clásicas de simplificación de mallas y otras que consideran algoritmos evolutivos para resolver este problema. La sección 3 describe los métodos utilizados en nuestra propuesta. En la sección 4 se presentan los experimentos llevados a cabo y los resultados obtenidos. Y finalmente, en la última sección se concluye el trabajo y se proponen algunos puntos que podrían considerarse en trabajos futuros.

II. SITUACIÓN ACTUAL

Las estrategias de simplificación de mallas pueden ser locales o globales. Las locales son aquellas que simplifican una malla de manera

[1] European Centre for Soft Computing, Mieres, 33600, España. E-mail:rosario.campomanes@softcomputing.es; sergio.damas@softcomputing.es

[2] European Centre for Soft Computing, Mieres, 33600, España. DECSAI y CITIC-UGR, Universidad de Granada, Granada, 18071, España. E-mail:oscar.cordon@softcomputing.es, ocordon@decsai.ugr.es

iterativa mediante la simplificación repetida de ciertas partes de ésta, mientras que las globales aplican operaciones de reducción a toda la malla a la vez [4]. Existen muchos métodos de simplificación de mallas, las siguientes dos sub-secciones revisan cada una de estas familias de métodos.

A. Métodos Incrementales basados en Actualizaciones Locales

Estos métodos ejecutan el proceso de simplificación de mallas como una secuencia de actualizaciones locales. Cada actualización reduce el tamaño de la malla y disminuye la precisión de ésta con respecto a la original.

El algoritmo de eliminación o borrado puede clasificarse en tres métodos diferentes de acuerdo a la diferencia entre los objetos seleccionados para borrar: eliminación de vértices [22], eliminación de lados [17] y eliminación de triángulos [14].

Este algoritmo realiza varias iteraciones sobre los vértices, lados o caras. Aquéllos que cumplan un determinado criterio de borrado previamente establecido son eliminados. Los restantes componentes de la malla se vuelven a triangular. El algoritmo termina cuando se alcanza una condición de finalización dada. Este método se implementa fácilmente, obtiene una buena aproximación y mantiene la topología de la malla original. Sin embargo, no evalúa el error total cometido en la aproximación.

Otro método iterativo es el de Optimización con Función de Energía [18], está basado en la eliminación de los lados de los triángulos que forman los puntos de una malla. La calidad de la malla obtenida se evalúa utilizando una función que se compone de la suma de las distancias al cuadrado entre los puntos de la malla original M y la aproximada, un factor proporcional al número de puntos de la malla y la suma de las longitudes de los lados de ésta. En cada iteración, el algoritmo borra el lado del triángulo cuya eliminación provoque el menor incremento en la función de energía.

El método de Mallas Progresivas [17] es una versión más actualizada de la anterior que mejora la resolución de la malla obtenida. Este algoritmo obtiene resultados de mucha calidad, aunque su implementación es compleja.

Por otro lado, el algoritmo de la Métrica del Error Cuadrático [11] consiste en la eliminación de pares de puntos o vértices de una malla original para obtener una nueva malla reducida. Para la selección de los puntos a borrar se utiliza un valor denominado coste de la eliminación.

B. Métodos Incrementales Basados en Actualizaciones Globales

Perteneciente a este tipo de métodos de simplificación de mallas, se encuentra el algoritmo *Coplanar* [16] que puede clasificarse de dos maneras, *bottom-up* y *top-down*. En general este método se basa en la detección de las caras *coplanares* –aquellas caras de los triángulos que forman la malla que se intersecan en el mismo plano–, la eliminación de los vértices compartidos por ambas caras y la re-triangulación de todos los puntos restantes de la nueva malla. Este método preserva la geometría de la malla original pero no tiene en cuenta el error que se comete durante el proceso de simplificación.

El método *Re-Tiling* [24], a partir de una superficie en 3D ya triangulada, selecciona un número determinado de puntos que forman la malla para volver a crear una nueva triangulación. Los nuevos polígonos a crear han de ser o cóncavos o convexos. Además, cada lado de un triángulo ha de compartir solamente 1 ó 2 lados de otro triángulo vecino. Satisfaciendo estas condiciones el algoritmo mantendrá la topología del modelo original.

La técnica de *Clustering* Adaptativo [21] se basa en la estructura *octree*. Un *octree* es una estructura jerárquica en forma de árbol donde cada nodo tiene 8 hijos que son vértices de la malla. Este método consiste en sustituir los vértices almacenados en una misma celda o nodo por uno solo.

Por último, hay que mencionar los algoritmos basados en *wavelets* [12], [15] y la simplificación basada en *envelopes* [6]. Todas ellas tienen en cuenta el error cometido en la reducción de la malla.

C. Métodos de Simplificación de Mallas basados en Algoritmos Evolutivos

Existen pocos estudios basados en la aplicación de técnicas de computación evolutiva para hacer frente al problema de la simplificación de mallas. En [9], Fujiwara y Sawai abordan este problema tratando de simplificar una superficie facial humana mediante la triangulación de la malla a partir de un número fijo de puntos utilizando para ello un algoritmo genético mono-objetivo. Los autores trabajaron con modelos 3D abiertos.

Por su parte Huang y Ho en [19] proponen un algoritmo evolutivo como una extensión del trabajo de Fujiwara y Sawai. Su método intenta mejorar al anterior utilizando un operador de cruce de vectores ortogonales (OAX) [19].

Otra técnica evolutiva para la simplificación de mallas fue propuesta por Xiandong *et al*. en [26]. Su método se basa en la reducción de la malla utilizando estructuras denominadas *super-face* y en la aplicación de un algoritmo genético mono-objetivo para encontrar la mejor simplificación posible.

III. MÉTODOS

Normalmente, los datos de una imagen se suelen representar utilizando el sistema de coordenadas cilíndricas estándar (ρ, θ, ζ) donde el eje $\rho = 0$ recorre verticalmente la mitad de la malla, θ es el

ángulo de giro y ζ es el eje vertical. Un punto de la superficie se representa mediante la tripleta (ρ, θ, ζ). Es fácil extender la superficie a largo del espacio $\mathbb{R}^2$, de forma que $x = \theta, y = \zeta$ y $z = \rho$ [19], como se muestra en la Fig. 1.

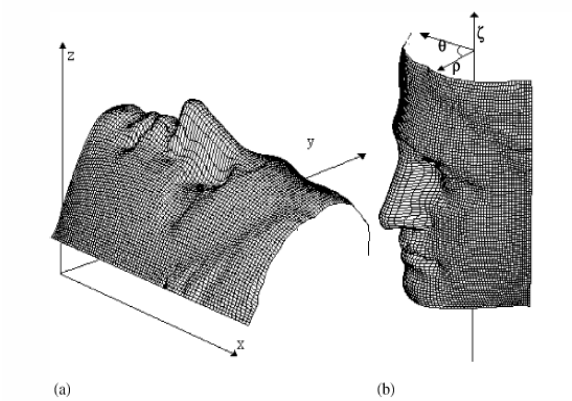

Fig. 1. Representación de superficies. (a) Representación de la malla en plano. (b) Representación de la malla en el sistema de coordenadas cilíndricas.

Dado un modelo M escaneado en 3D, es posible reducir esta superficie poligonal en un problema en 2D. Una superficie en 3D de este tipo puede representarse mediante la función:

$$f: (x, y) \in \mathbb{R}^2 \rightarrow z \in \mathbb{R} \qquad (1)$$

El objetivo principal es crear una malla poligonal nueva que mejor aproxime a la original. Se deben evitar puntos innecesarios y hay que mantener la forma y características de la malla inicial. Por tanto, necesitamos colocar un número de puntos n menor que el total de puntos N de la malla original. El número de puntos a situar puede ser fijado por el usuario o bien ser un valor estimado por el algoritmo. Nuestros experimentos se basan en esta segunda opción ya que creemos que es mejor que el algoritmo intente resolver el problema con el menor número posible de puntos conocidos.

La mayoría de los métodos explicados en la sección anterior utilizan triangulación de Delaunay. Una triangulación de Delaunay [1] de un conjunto de puntos P es una triangulación $DT(P)$ tal que ningún punto en P pertenece a la circunferencia circunscrita de cualquier triángulo de $DT(P)$. Este tipo de triangulación maximiza los ángulos de los triángulos, evita crear triángulos muy estrechos y es la más utilizada en la gran mayoría de los casos.

Si se tiene un conjunto inicial de puntos en $\mathbb{R}^2$ P_n, su correspondiente triangulación de Delaunay se denota como $D(P_n)$.

En nuestro problema, codificaremos un cromosoma como una configuración P_n de n puntos en el plano 3D. El genotipo consiste en estas configuraciones 3D:

Dada una población de N individuos, cada individuo representará la configuración de una malla determinada, es decir, cada individuo es una malla simplificada con respecto a la original, esto es, con menos puntos que la del modelo de referencia.

Para codificar lo anterior, se utilizará un vector global con las coordenadas de los puntos de la malla original, y un vector cromosoma binario de longitud n (número de puntos de la nueva malla) por cada individuo. Cada posición del vector cromosoma codifica un gen, por tanto almacenará un 0 ó un 1. Un 0 en la posición i del vector cromosoma indica que el punto con coordenadas x, y, z de la posición i del vector de coordenadas no está en la nueva malla. Por el contrario, un 1 en la posición i del vector cromosoma, indica que dicho punto forma parte de la nueva malla. Los 4 puntos de las esquinas de la malla serán siempre fijos en todas las configuraciones de los cromosomas. Esto permite mantener la forma y los límites de la malla reducida para que coincida con la original [9], [19]. El proceso de simplificación llevado a cabo por el algoritmo es el siguiente: se transforma el modelo original 3D de N (P_N) puntos a 2D con los mismos puntos. Se realiza simplificación obteniendo una nueva malla 2D con n puntos (P_n), siendo $P_n < P_N$. Se aplica triangulación de Delaunay a esa nueva malla 2D, obteniendo $D(P_n)$. Esta malla final triangulada $D(P_n)$ 2D se convierte a 3D y se obtiene el modelo 3D final de n puntos. El algoritmo ha de seleccionar el modelo que mejor aproxime al original. La siguiente sub-sección explica cómo calcular el error entre dos mallas, la original y la simplificada.

A. Cálculo del Error y Distancia entre Vértices

Hemos calculado el error siguiendo el mismo procedimiento que el presentado por Huang y Ho en [19]. Cada triángulo de Delaunay $T_i \in D(P_n)$ contiene un determinado número de puntos de la malla original (x, y) que no han sido triangulados. La distancia d_p de cada punto p se define como:

$$d_p = |z_p - \tilde{z}_p| \qquad (2)$$

donde z_p es la coordenada z del punto p y $\tilde{z}_p$ es el valor de la coordenada z del punto p en el nuevo plano definido por el triángulo que contiene a ese punto. El error e_i para T_i es la suma de las distancias sobre todos los puntos contenidos en T_i donde:

$$e_i = \sum_{p \in T_i} d_p \qquad (3)$$

El error total e se define como:

$$e = \sum_{T_i \in D(P_n)} e_i \qquad (4)$$

Por tanto, para cada triángulo T_i de una triangulación de Delaunay, se calcula la ecuación

del plano que forman los 3 vértices del triángulo T_i. Después, para cada punto dentro del triángulo, se calcula su correspondiente coordenada $\tilde{z}$ en ese plano. El error es la distancia Euclídea entre la coordenada z original del punto p, y la coordenada $\tilde{z}$ del punto p dentro del plano formado por el triángulo T_i. El error total es la suma de las distancias para todos los puntos que se encuentran dentro del conjunto de triángulos que forman la triangulación de Delaunay.

B. *Recombinación*

Para este trabajo se han utilizado los siguientes operadores de cruce: cruce uniforme [23] y cruce basado en vectores ortogonales [19].

C. *Propuesta Mono-Objetivo*

Nuestro algoritmo mono-objetivo se basa en el desarrollado por Huang y Ho en [19]. Consiste en la inicialización de la población, un esquema de selección, operadores genéticos y un criterio de terminación. La función de evaluación ha sido adaptada a nuestro problema para considerar los dos objetivos planteados:

$$\mathrm{F}(m) = \mathrm{w} * E(m) + (1 - \mathrm{w}) * T(m) \quad (5)$$

donde m es la malla, w es un peso entre $[0,1]$, $E(m)$ es el error dado por Ec. (4) que debe ser minimizado y $T(m)$ es el número de triángulos totales de la malla. Su adaptación consiste en construir dicha función de evaluación combinando múltiples objetivos en una sola función de suma ponderada [2], [5], [8] de objetivos normalizados. Se genera un conjunto de soluciones óptimas de Pareto dando diferentes pesos a la función y ejecutando repetidas veces el algoritmo.

El pseudocódigo del método mono-objetivo se detalla a continuación:

Inicialización de la población: Se inicializa la población aleatoriamente asignando n puntos a cada individuo. Los 4 puntos de las esquinas de la malla se fijan para mantener la forma de la malla inicial.

Esquema de selección: Se utiliza un modelo de selección elitista de forma que el individuo con mejor *fitness* es seleccionado para reproducirse.

Mutación: Se realiza mutación en cada gen del cromosoma de un individuo con una probabilidad $p_m = \frac{1}{chromosome\ length}$.

Cruce: A partir de dos padres elegidos se generan dos hijos mediante cruce uniforme y ortogonal.

Criterio de parada: Cuando se llega al máximo de generaciones fijadas, el algoritmo finaliza.

D. *Propuesta Multi-Objetivo*

En el caso de nuestra propuesta de algoritmo multi-objetivo basada en NSGA-II [7], tendremos dos objetivos a minimizar, el error y el número de triángulos. El resultado de la función de evaluación ha de ser minimizado. La función de evaluación para una determinada malla m será la siguiente:

$$Min\,FM^1(m) = E(m);\ Min\,FM^2(m) = T(m) \quad (6)$$

El esquema del método se detalla a continuación:

Inicialización de la población: Igual que en el algoritmo mono-objetivo.

Esquema de selección: El algoritmo combina la población actual con la descendencia generada mediante recombinación para crear la siguiente generación. Los mejores individuos en términos de no dominancia y diversidad son los elegidos para reproducirse considerando los frentes de no dominancia de NSGA-II.

Mutación: Se realiza mutación en cada gen del cromosoma de un individuo con una probabilidad $p_m = \frac{1}{chromosome\ length}$.

Cruce: A partir de 2 padres elegidos al azar se generan 2 hijos mediante cruce uniforme y ortogonal.

Criterio de parada: Cuando se llega al máximo de generaciones fijadas, el algoritmo finaliza.

IV. EXPERIMENTACIÓN Y ANÁLISIS DE RESULTADOS

Para la evaluación del rendimiento y el diseño del experimento se ha utilizado un conjunto de datos correspondiente a una malla de una superficie facial abierta. El fichero se denomina Laurana.ply [30], está compuesto por 922 puntos y 1667 triángulos. El modelo de superficie tridimensional abierto se muestra en Fig. 2:

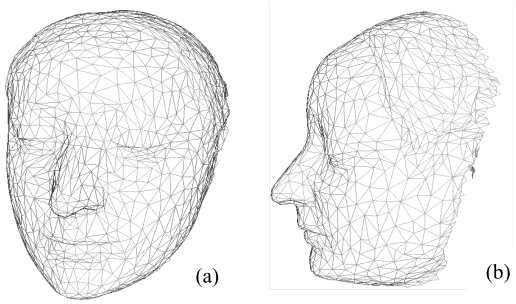

Fig. 2. Mallas 3D iniciales pertenecientes al fichero de datos Laurana.ply.

Los dos algoritmos, mono y multi-objetivo han sido implementados en C/C++ y todos los experimentos se han llevado a cabo en un Intel Core 2 Quad CPU Q8500 2.66 GHz con 4GB RAM, sobre Windows 7 Profesional. Como hemos visto, para realizar el estudio comparativo, el algoritmo

mono-objetivo propuesto en [19] ha sido adaptado para resolver nuestro problema.

Para la experimentación hemos diferenciado dos variantes del problema, la variante 1 utiliza cruce uniforme -ya que genera mayor diversidad que el cruce binario- y la variante 2, cruce ortogonal con matriz L4 (basado en las matrices de Taguchi propuestas en [20]). El motivo por el cual se ha seleccionado cruce ortogonal con una matriz L4 y no L8 o L12 es la eficiencia, L4 tarda alrededor de 15 minutos por ejecución, mientras que las otras dos matrices no las hemos considerado adecuadas en términos de tiempos de ejecución.

Se han ejecutado ambas variantes para cada algoritmo con el mismo conjunto de datos. Los parámetros utilizados han sido: tamaño de población de 100, número de generaciones igual a 50, probabilidad de cruce de 0.8 y probabilidad de mutación de 1/longitud cromosoma. El algoritmo multi-objetivo se ha ejecutado 10 veces con diferentes semillas. El algoritmo mono-objetivo se ha ejecutado utilizando 11 vectores con pesos diferentes para los dos objetivos de la función de evaluación. Los pesos aplicados al primer objetivo varían de 1 a 0, con intervalos de 0.1, y los pesos para el segundo objetivo varían de 0 a 1 con el mismo intervalo.

Para comparar el comportamiento de dos algoritmos diferentes se suele utilizar el frente de Pareto óptimo. Sin embargo, en muchos problemas reales, el frente de Pareto es desconocido y no puede calcularse en un tiempo razonable, como sucede en este estudio. Por ello se considera el frente de Pareto aproximado, que se obtiene mediante la agregación del conjunto de soluciones generado por cada algoritmo en todas las ejecuciones. El frente de Pareto aproximado del algoritmo mono-objetivo se calcula uniendo las 11 soluciones de cada vector ponderado. Las soluciones repetidas y las dominadas se eliminan dando como resultado una aproximación al frente de Pareto. En el caso del problema multi-objetivo, las soluciones obtenidas por las 10 ejecuciones se unen, se eliminan las repetidas y se aplica la dominancia de Pareto para obtener el frente aproximado agregado.

Para medir las diferencias de calidad entre dos conjuntos de soluciones se utilizan tres de los indicadores de calidad habituales. Uno de ellos es el indicador unario *Hypervolume Ratio* o HVR [27]. HVR se utiliza comúnmente para la comparación de optimizadores multi-objetivo [28], [29]. Los otros dos indicadores utilizados son el indicador binario Epsilon [29] y la métrica binaria de Cobertura propuesta por Zitzler y Thiele en [27].

A. Análisis de Resultados de los Indicadores

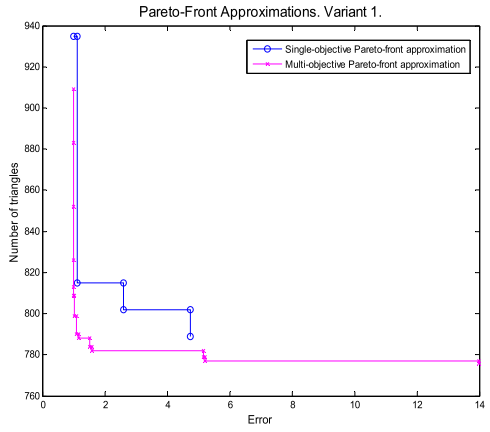

Fig. 3. Gráfica de los frentes de Pareto aproximados obtenidos por cada algoritmo en la variante 1.

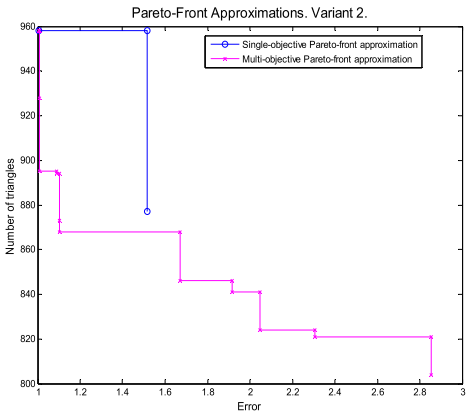

Fig. 4. Gráfica de los frentes de Pareto aproximados obtenidos por cada algoritmo en la variante 2.

Las Figs. 3 y 4 muestran los frentes de Pareto aproximados agregados para cada algoritmo en las dos variantes de los métodos: 1, cruce uniforme y 2, cruce ortogonal. El algoritmo multi-objetivo converge mejor que el mono-objetivo. Se observa que la propuesta multi-objetivo obtiene soluciones en casi todo el espacio de búsqueda (Figs. 3 y 4) mientras que el mono-objetivo encuentra soluciones en el primer cuadrante del espacio de búsqueda (Fig. 3) o bien tiende a encontrar soluciones sólo en la parte superior izquierda del espacio (Fig. 4).

TABLA I
MEDIA Y DESVIACIÓN ESTÁNDAR DE HVR

	Variante 1	**Variante 2**
Mono-Obj.	0.1410(-)	0.0200(-)
Multi-Obj.	0.1484(0.0139)	0.7856(0.0640)

TABLA II
MEDIA Y DESVIACIÓN ESTÁNDAR DE EPSILON Y COBERTURA

Indicador Epsilon		
Variante 1		
Métodos	Mono-Obj.	Multi-Obj.
Mono-Obj.	-	1.1006(0.0054)
Multi-Obj.	0.9998(0.0075)	-
Variante 2		
Métodos	Mono-Obj.	Multi-Obj.
Mono-Obj.	-	1.001(0.0014)
Multi-Obj.	0.3526(0.0001)	-
Indicador Cobertura		
Variante 1		
Métodos	Mono-Obj.	Multi-Obj.
Mono-Obj.	-	0.0539(0.079)
Multi-Obj.	0.834(0.236)	-
Variante 2		
Métodos	Mono-Obj.	Multi-Obj.
Mono-Obj.	-	0.10(0.056)
Multi-Obj.	1(-)	-

La Tabla 1 recoge la media y la desviación estándar (entre paréntesis) de los valores de la métrica HVR para las aproximaciones de los dos métodos, mono y multi-objetivo en las dos variantes. La Tabla 2 muestra el valor medio y la desviación estándar obtenidos para el indicador Epsilon y de Cobertura en los dos métodos y variantes. En el caso del indicador Epsilon un valor más bajo indica un mejor rendimiento. En el indicador de Cobertura un valor más alto indica mejor rendimiento.

Se observa que el algoritmo multi-objetivo obtiene mejores resultados que el mono-objetivo en ambos indicadores. La diferencia entre algoritmos es más alta en el caso del cruce ortogonal (variante 2) que utilizando cruce uniforme (variante 1). Además, se obtiene una mayor diversidad de soluciones con el método multi-objetivo que con el mono-objetivo ya que el número de soluciones del Pareto aproximado es mayor en el caso del algoritmo multi-objetivo en ambas variantes del problema. En cuanto a los tiempos de ejecución, podemos señalar que son similares en ambos métodos (aproximadamente 10 minutos) siendo ligeramente mayor el tiempo de ejecución en la variante 2 (15 minutos) que en la 1 para ambos algoritmos.

B. Análisis y Comparación de Soluciones

A partir de los frentes de Pareto aproximados se han seleccionado tres soluciones diferentes: una es el mejor valor obtenido para el primer objetivo, otra es el mejor valor obtenido para el segundo objetivo y la última se corresponde con el valor de mejor rendimiento promedio. Para calcular esta última solución, se han obtenido aleatoriamente 1000 pesos (w) con valores diferentes comprendidos entre [0,1] y se ha calculado el valor medio de la función de agregación de los dos objetivos:

$$\bar{\mathrm{F}}(s_i) = \frac{\sum_{i=j}^{1000} w_j * Obj1(s_i) + (1 - w_j) * Obj2(s_i)}{1000} \quad (7)$$

La solución seleccionada es la que presenta el valor de la agregación más bajo. Cada solución está compuesta por los valores de los dos objetivos: error y número de triángulos. Debido a que los objetivos no están normalizados es necesario aplicarles un factor de escalado α:

$$\alpha = \frac{\sum_{i}^{|PA_a|} \frac{Obj_2(s_i)}{Obj_1(s_i)}}{|PA_a|} \quad (8)$$

donde $|PA_a|$ es el número de soluciones del frente de Pareto aproximado y s_i es una solución de ese Pareto. Así, la fórmula final de la agregación para calcular el valor medio es la siguiente:

$$\bar{\mathrm{F}}(s_i) = \frac{\sum_{j=1}^{1000} w_j * Obj1(s_i) + (1 - w_j) * Obj2(s_i)}{1000} \quad (9)$$

TABLA III
MEJORES SOLUCIONES EN FRENTES DE PARETO APROXIMADOS EN LAURANA.PLY

	Variante 1			
	Mono-Obj.		**Multi-Obj.**	
Solución	**Obj.1**	**Obj.2**	**Obj.1**	**Obj.2**
Min. error	1.010262	935	1.000083	883
Min. nº de triángulos	4.736825	789	13.98614	776
Mejor rendimiento	1.107537	815	1.594025	782
	Variante 2			
	Mono-Obj.		**Multi-Obj.**	
Solución	**Obj.1**	**Obj.2**	**Obj.1**	**Obj.2**
Min. error	1.006508	958	1.0065508	958
Min. nº de triángulos	1.516511	877	2.854556	804
Mejor rendimiento	1.516511	877	1.916304	841

La Tabla 3 muestra las tres mejores soluciones antes mencionadas, para las dos propuestas por variante. En la variante 1, el método multi-objetivo

gana al mono-objetivo en dos soluciones: obtiene el menor valor para el error: 1.000083 frente a 1.010; y el menor número de triángulos, 776 frente a 789. En cuanto a la solución con mejor rendimiento, el menor error es para el algoritmo mono-objetivo (1.10 contra 1.59) y lo mismo ocurre con el mínimo número de triángulos (782 frente a 815). En la variante 2, se observa que el multi-objetivo obtiene el mismo valor que el mono-objetivo para el error: 1.006508; así como un número menor de triángulos, 804 frente a 877. En la solución de mejor rendimiento, el algoritmo mono-objetivo alcanza el menor error (1.51 contra 1.91) debido a que el número de triángulos es menor que en el método multi-objetivo (841 frente a 877).

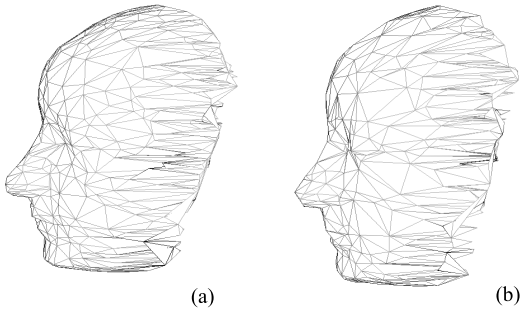

Fig. 5. Modelos 3D simplificados obtenidos por el algoritmo mono-objetivo, variante 1: (a) solución mínimo error y (b) solución mínimo número de triángulos.

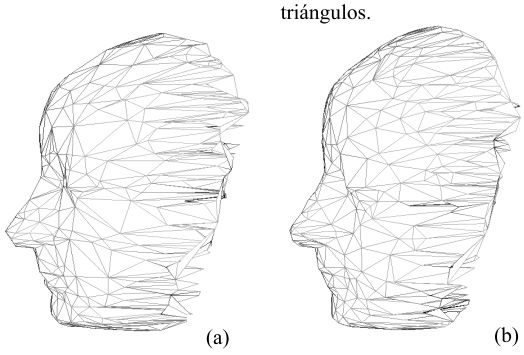

Fig. 6. Modelos 3D simplificados obtenidos por el algoritmo multi-objetivo, variante 1: (a) solución mínimo error y (b) solución mínimo número de triángulos.

Las soluciones obtenidas por el algoritmo mono-objetivo en la variante 1 muestran (Fig. 5) que la malla con la mejor aproximación es la de menor error. Si se observan los modelos 3D se puede ver que la forma de la cara es mejor en la malla que presenta el error más bajo que la de menor número de triángulos. Se aprecia una mayor diferencia en áreas de la boca y la nariz. Se muestran las soluciones de menor error y de menor número de triángulos. En el caso del método multi-objetivo en la variante 1 la mejor aproximación la obtiene la solución de menor error, Fig. 6. En este caso, las diferencias entre soluciones son menores que las que se obtenían con el algoritmo mono-objetivo. La zona nasal es casi igual y la mayor diferencia se presenta en el área de la boca.

C. *Test Estadístico de Wilcoxon*

Para comprobar si existe o no una diferencia significativa entre los resultados obtenidos por ambos algoritmos se ha realizado el test estadístico de Wilcoxon [25]. Este test se utiliza para analizar si los resultados de la comparación entre los indicadores de cada método son significativas o no. Es útil para analizar el comportamiento de los algoritmos evolutivos [10]. Hay que destacar que se ha aplicado el test a las métricas binarias. El procedimiento seguido para aplicar el test a nuestros datos se detalla en [3].

TABLA IV
P-VALORES DE COMPARACIÓN INDICADORES

Indicador Cobertura	
Algoritmos Mono y Multi-Objetivo	
P-valor en variante 1	0.0336
P-valor en variante 2	0.00000935
Indicador Epsilon	
Algoritmos Mono y Multi Objetivo	
P-valor en variante 1	0.005
P-valor en variante 2	0.000016

La Tabla 4 muestra los p-valores de la comparación entre los indicadores Cobertura y Epsilon en los dos algoritmos, en sus dos variantes. Hemos fijado un nivel de significación igual a 0.05 y la hipótesis nula sería la igualdad entre algoritmos. Los resultados (Tabla 4) indican que los p-valores obtenidos son menores que 0.05 y por tanto se rechaza la hipótesis nula de igualdad de comportamiento entre algoritmos en ambos indicadores. Por ello, se puede afirmar que existe diferencia significativa entre el método mono-objetivo y el multi-objetivo con un nivel de significación del 0.05. Según nuestro estudio podemos decir que en el problema que planteamos el algoritmo multi-objetivo es significativamente mejor que el método mono-objetivo con un nivel de confianza del 95 %.

V. Conclusiones

Hemos planteado un algoritmo evolutivo multi-objetivo NSGA-II para resolver el problema de la simplificación de mallas 3D previamente resuelto utilizando un método mono-objetivo.

El algoritmo mono-objetivo está basado en la aproximación de un modelo 3D usando una malla generada a partir de un número de puntos determinado situados de forma adaptativa. Este método utiliza un algoritmo genético para crear diferentes triangulaciones de Delaunay y escoger la más precisa. Nosotros lo hemos adaptado de forma que la función de evaluación sea un escalar que

combine el error total de la malla solución con su número de triángulos. La solución multi-objetivo, consiste en aplicar un algoritmo NSGA-II para encontrar las mejores configuraciones de puntos: mallas cuya precisión sea mayor y contengan el menor número de triángulos, aproximándose lo más posible a la original.

El rendimiento de ambos métodos ha sido analizado utilizando un conjunto de datos y los frentes de Pareto aproximados obtenidos por cada algoritmo. Para la comparación de resultados se han empleado tres métricas diferentes, la unaria HVR, y las dos binarias, Epsilon y Cobertura.

Tras los experimentos desarrollados, es evidente que nuestra propuesta aproxima mejor que el algoritmo mono-objetivo. El frente de Pareto aproximado del método multi-objetivo muestra una mayor diversidad y convergencia de soluciones.

Finalmente, surgen varias ideas enfocadas a trabajos futuros. Por una parte, podría ser interesante crear un método que genere aproximaciones de superficies 3D completas y, por otro lado, podría diseñarse una tarea de post-procesamiento como complemento a la simplificación que mejore la calidad del modelo final.

REFERENCIAS

[1] F. Aurenhammer. Voronoi Diagrams. A Survey of a Fundamental Geometric Data Structure, *ACM Computing Surveys*, 23(3), pp. 345-405, 1991.

[2] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*, North-Holland, Amsterdam, 1983.

[3] Chica, M., Cordón, O., Damas, S., & Bautista, J. (2010). Multi-objective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. *Information Sciences*, 180, pp. 3465–3487.

[4] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms, *Computers & Graphics*, 22(1), pp. 37-54, 1998.

[5] C.A. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multiobjective Problems*, Springer, Berlin, 2007.

[6] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes, *Computer Graphics (SIGGRAPH '96 Proceedings)*, *ACM Press*, pp 119-128, Aug. 6-8 1996.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6(2), pp. 182-194, April 2002.

[8] A.E Eiben and J.E Smith. Introduction to Evolutionary Computing. 2003. Springer Berlin Heidelberg New York.

[9] Y. Fujiwara and H. Sawai. Evolutionary computation applied to mesh optimization of a 3-D facial image, *IEEE Transactions Evolutionary Computation*, 3(2), pp. 113-123.

[10] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case of study, *CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics* 15, pp. 617–644, 2009.

[11] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics, *Computer Graphics (SIGGRAPH '97 Proceedings),* pp. 209–216, 1997.

[12] M.H. Gross, O.G. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures, *IEEE Transactions on Visualization and Computer Graphics,* 2(2), pp. 130-144, June 1996.

[13] M.J. Haemer and M.J. Zyda. Simplification of objects rendered by polygonal approximation, *Computers and Graphics*, 15(2), pp. 175–184, 1991.

[14] B. Hamann. A data reduction scheme for triangulated surfaces, *Computer Aided Geometric Design*, 11(20), pp. 197–214, 2004.

[15] D.J. Hebert and H.-J. Kim. Image encoding with triangulation wavelets *Proceedings SPIE*, 2569(1), pp. 381-392, 1995.

[16] P. Hinker and C. Hansen. Geometric optimization. Gregory M. Nielson and Dan Bergeron, editors, *Proceedings Visualization '93*, pp. 189–195, October 1993.

[17] G. Hoppe. Progressive meshes, *SIGGRAPH'96 Conference Proceedings*, pp. 99–108, 1996.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization, *ACM Computer Graphics Proceedings,* pp. 19- 26, 1993.

[19] H.L. Huang, S. Y. Ho. Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm, *Pattern Recognition* 36, pp. 1065-1081, 2003.

[20] T. Mori. Taguchi techniques for image and pattern developing technology, Prentice-Hall, New Jersey, 1995.

[21] G. Pengdong, L. Ameng, L. Yongquan, H. Jintao, L. Nan and Y. Wenhua. Adaptative Mesh Simplification Using Vertex Clustering with Topology Preserving, *CSSE '08 Proceedings of the 2008 International Conference on Computer Science and Software Engineering,* 2, pp. 971-974, December 2008.

[22] W. J. Schroeder, J. A. Zarge and W. E. Lorensen. Decimation of triangular meshes, *Computer Graphics (SIGGRAPH '92 Proceedings)*, pp. 65–70, 1992.

[23] G. Syswerda. Uniform crossover in genetic algorithms, Schaffer 1989, pp. 2-9.

[24] G. Turk. Re-tiling polygonal surfaces, *Edwin E. Catmull*, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26, pp. 55–64, July 1992.

[25] F. Wilcoxon. Individual Comparisons by Ranking Methods, *Biometrics Bulletin*, 1(6), pp. 80-83, 1945.

[26] T. Xiaodong, W. Yuexian, Z. Xionghui and R. Xueyu. Mesh Simplification Based on Super-Face and Genetic Algorithm, *Reverse Engineering. Advanced Manufacturing Technology*, 29, pp. 303-312, 2002.

[27] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation*, 3(4), pp. 257–271, 1999.

[28] E. Zitzler, L. Thiele, and K. Deb. Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computation*, 8(2), pp. 173–195, 2000.

[29] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Transactions on Evolutionary Computation*, 7(2), pp. 117–132, 2003.

[30] http://meshlab.sourceforge.net/