Ciencias Matemáticas, Vol. 29, No. 1, Pag. 7-18, 2015
Recibido 05-2015

Ajuste de curvas y superficies por polinomios implícitos
Curve and surface fitting by implicit polynomials

Rubén Interian Kovaliova[1]

Resumen La búsqueda de un polinomio implícito que aproxime cierto conjunto de observaciones X es el objetivo de muchas investigaciones en los últimos años. Sin embargo, la gran mayoría de los algoritmos de ajuste existentes presuponen el conocimiento del grado del polinomio implícito que mejor representa a los puntos. Este trabajo propone un algoritmo capaz de determinar el grado del polinomio necesario para la representación del conjunto de datos. Para este fin, se define una novedosa medida de distancia entre X y el polinomio implícito. El algoritmo planteado se basa en la idea de ir incrementando paulatinamente el grado, mientras haya una mejora en la suavidad de las soluciones.

Abstract Finding an implicit polynomial that fits a set of observations X is the goal of many researches in recent years. However, most existing algorithms assume the knowledge of the degree of the implicit polynomial that best represents the points. This paper proposes an algorithm capable of find the degree of the polynomial needed for the representation of the data set. To this end, a new distance measure between X and implicit polynomial is defined. The proposed algorithm is based on the idea of gradually increasing the degree, while there is an improvement in the smoothness of the solutions.

Palabras Clave
Ajuste — Polinomio Implícito — Estabilidad — Metaheurística

[1] *Departamento de Matemática, Universidad de Habana, Habana, Cuba, ruben@matcom.uh.cu*

Introducción

El problema de representar matemáticamente objetos en dos (2D) y, especialmente, tres (3D) dimensiones aparece en un sinnúmero de situaciones prácticas, y está presente en diferentes campos, tales como gráficos por computadora y visión por computadora. En tareas de modelación, reconstrucción 3D, reconocimiento, una buena representación de un objeto observado, es fundamental.

Lamentablemente, es usual que los datos provenientes del mundo real sean discretos, así como incompletos. Los modelos suelen obtenerse a partir de imágenes, vídeos, escáneres 3D y otros dispositivos de captura. La naturaleza de estos dispositivos permite obtener una cantidad finita y discreta de datos del objeto analizado, comúnmente en forma de puntos. La búsqueda de un modelo que "mejor" se aproxime (ajuste) a este conjunto de observaciones (puntos) es el objetivo de muchas investigaciones en los últimos años[29, 5, 16].

Los polinomios implícitos (PI, o *IP*, por sus siglas en inglés) han demostrado ser un poderoso instrumento a la hora de modelar objetos del mundo real frente a otros tipos de representación, como la explícita o la paramétrica[28], mostrando una sorprendente variedad de formas (Figura 1, "Las Cerezas" y "La Pera"[1]), así como buenas propiedades donde sea necesario:

- Una representación compacta de la superficie
- El mantenimiento de invariantes algebráicas y geométricas, tales como área y volumen, entre otras [24]
- Una forma muy rápida de clasificar los puntos en externo e interno al objeto
- Robustez al ruido y a la oclusión [28]

Polinomios Implícitos

Por una curva (en 2D) o superficie (en 3D) polinomial implícita se entiende, el conjunto de ceros (*zero set*) o el nivel cero de una función polinomial f.

$$\begin{aligned} f_a(x,y,z) &= \sum_{0\le i+j+k\le n} a_{ijk}x^i y^j z^k \\ &= \underbrace{(1\ x\ y\ z\ x^2\ \ldots)}_{m(\mathbf{x})^T}\underbrace{(a_{000}\ a_{100}\ a_{010}\ a_{001}\ a_{200}\ \ldots)^T}_{\mathbf{a}} \\ &= m(\mathbf{x})^T\mathbf{a} \end{aligned}$$

El vector $m(\mathbf{x})^T$ se denomina, vector de monomios, y solo depende del punto (x,y,z); $\mathbf{a}$ es el vector de parámetros, o coeficientes, del polinomio.

Formalmente, una superficie polinomial implícita es el conjunto de soluciones de la ecuación:

[1] Ecuación de "La Pera": $((x^2+y^2+0{,}2*z^2)^2-(x^2+y^2))*((x-0{,}55-0{,}05*(1{,}1*z+1{,}75))^2+y^2+(1{,}1*z+1{,}75)^2-2{,}3)-0{,}5=0$

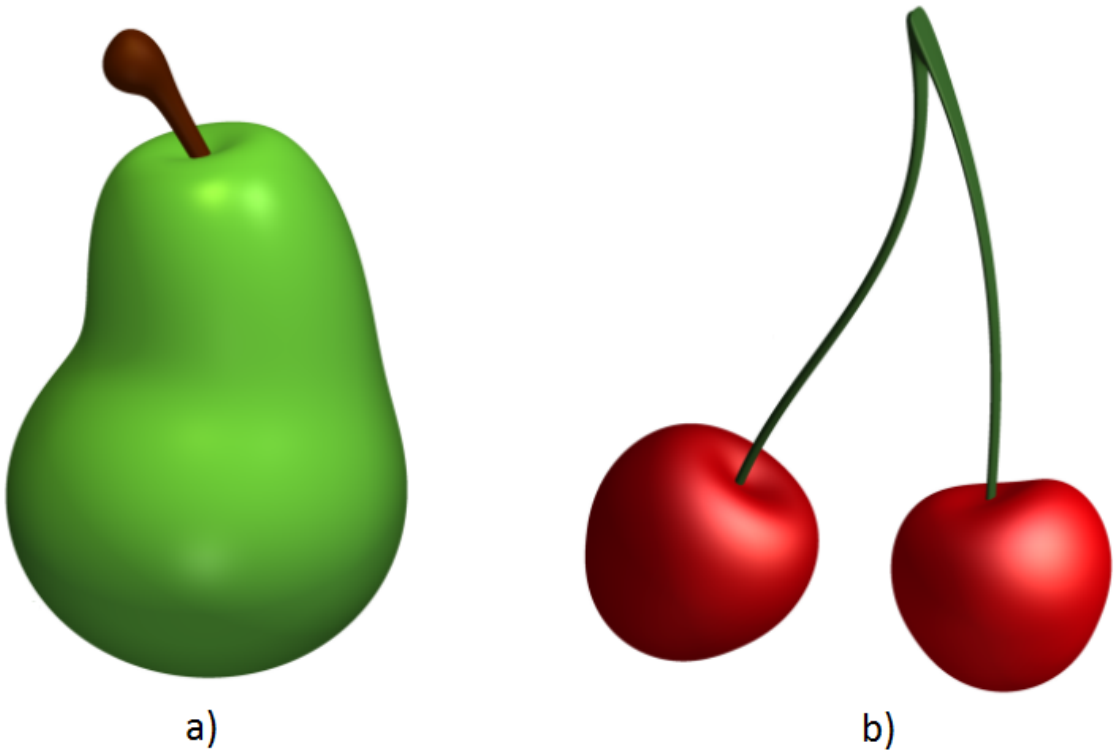

Figura 1. Algunos polinomios implícitos. (a) "La Pera", PI de 6^{to} grado (b) "Las Cerezas". Tomado de [18]

$$m(\mathbf{x})^T \mathbf{a} = 0 \quad (1)$$

donde $\mathbf{x}$ es la incógnita. El conjunto de soluciones de la ecuación (1) se denotará $Z(f_a)$. En ámbos casos - 2D y 3D - usualmente se utiliza el término polinomio implícito.

El objetivo principal de este artículo es desarrollar un algoritmo de ajuste de polinomios implícitos propio *capaz de determinar el grado del polinomio a utilizar*.

Estructura del trabajo

Este trabajo está organizado de la siguiente manera. En la sección 1, se exponen los principales métodos de ajuste de polinomios implícitos, lineales y no lineales, presentes en la literatura. En la sección 2, se introduce una novedosa medida de distancia entre un conjunto de puntos y un polinomio implícito. En la sección 3, se propone un algoritmo de ajuste que utilice esta medida para encontrar polinomios implícitos que tengan ciertas propiedades deseadas, sin conocimiento previo del grado que debe tener el polinomio implícito. En la sección 4, se analizan los resultados experimentales obtenidos a partir de las propuestas realizadas en las secciones anteriores.

1. Métodos de Ajuste de Polinomios Implícitos

De acuerdo a como se define la distancia $dist(x_i, Z(f_a))$ entre un punto x_i del espacio, y el nivel cero de f_a los métodos de ajuste se clasifican en lineales y no lineales.

1.1 Métodos no lineales

Actualmente no existe una vía simple para hallar la distancia de un punto a $Z(f_a)$. Para encontrar la distancia de manera precisa, se requiere el uso de métodos iterativos [27, 11, 15].

Los métodos no lineales fueron históricamente los primeros en aparecer [22, 23]. La idea principal de estos métodos es reemplazar la distancia exacta de un punto a una curva o superficie implícita, por su aproximación de Taubin de primer orden[25]:

$$dist(\mathbf{x}, Z(f_a)) \approx \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$$

Se han utilizado también aproximaciones de orden superior.

A este tipo de distancia usualmente se le identifica como *geométrica*, ya que hace uso de la información contenida en las derivadas parciales de f_a.

1.2 Métodos lineales

Los métodos lineales han sido los más usados en investigaciones [6] y aplicaciones prácticas [30] por proveer una solución que no requiere el uso de métodos iterativos y, por tanto, ser más rápidos y eficientes. Entre los algoritmos más conocidos se encuentran, el algoritmo lineal clásico y el algoritmo 3L.

Los algoritmos lineales utilizan la llamada distancia algebraica:

$$dist(\mathbf{x}, Z(f_a)) \approx f_a(\mathbf{x})$$

Se asume que, dada la continuidad de f_a, en puntos cercanos a $Z(f_a)$), el valor de f_a debe ser cercano a cero, y en puntos más alejados, debe ser mayor.

El problema de ajuste se resuelve en este caso como un sistema sobredeterminado $M\mathbf{a} = \mathbf{b}$, donde M es una matriz en cuyas filas están los vectores de monomios de los puntos del conjunto X (llamada matriz de monomios), y $\mathbf{b}$ (*inicialmente*) es un vector de ceros.

La solución mínimo cuadrática de este sistema sobredeterminado [8] es:

$$\mathbf{a} = (M^T M)^{-1} M^T \mathbf{b} = M^+ \mathbf{b}$$

donde M^+ es la *pseudoinversa de Moore-Penrose[3]*.

Sin embargo, los algoritmos lineales sufren de inestabilidad, ya que a pequeños cambios en las observaciones, pueden corresponder soluciones totalmente diferentes [6]. Además, debido a que $\mathbf{b} = 0$, existe una solución trivial, $\mathbf{a} = 0$, que en la práctica se debe evitar. Por ende, este método lineal clásico es mejorado por otros algoritmos lineales, como el algoritmo 3L[6]. Se incrementa la estabilidad del ajuste, y el vector b es sustituido por otro vector no nulo.

2. Una Medida de Distancia a un Polinomio Implícito

Como se había planteado en la sección 1, para desarrollar un buen algoritmo de ajuste, es necesario, en primer lugar, definir una función de distancia apropiada de un punto a un polinomio implícito. Para esto, primero se definirán dos medidas (una, de disimilitud, y otra, de suavidad) que ponderarán, respectivamente, la "separación" y la "proximidad" de estos puntos a $Z(f_a)$.

2.1 Medida de Disimilitud

La medida de disimilitud que se utilizará debe evaluar *cuantitativamente* la distancia real que existe entre el PI y los puntos. Entre las funciones de distancia presentes en los estudios analizados, se seleccionó la aproximación de Taubin de primer orden [25], que provee una manera rápida de calcular una buena (analíticamente fundamentada) aproximación a la distancia entre un polinomio implícito y un punto:

$$dissim(\mathbf{x}, Z(f_a)) = \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$$

Por consiguiente, para evaluar la distancia que hay entre un *conjunto de puntos* $\mathbf{X}$ y un polinomio implícito, se usará la función:

$$dissim(\mathbf{X}, Z(f_a)) = \frac{1}{N}\sum_{i=1}^{N} \frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} \qquad (2)$$

Llamaremos a esta medida *función de separación* (disimilitud) entre un conjunto $\mathbf{X}$ de N puntos y $Z(f_a)$.

2.2 Medida de Suavidad

En algunos de los últimos trabajos sobre el tema [21, 28, 29, 16], se ha utilizado la idea de controlar en el proceso de ajuste ciertas características geométricas de los polinomios implícitos: la longitud del gradiente cerca de los puntos, la dirección del gradiente con respecto al vector tangente estimado en un punto, etc. Con esto se persigue el objetivo de evitar los efectos del sobreajuste, promoviendo aquellos polinomios que tengan propiedades deseadas.

Por estas razones, además de la medida de distancia, se decidió utilizar también una medida de "suavidad" que establezca un criterio de proximidad entre un conjunto de puntos y un polinomio implícito. Es preciso notar que esta medida debe evaluar *cualitativamente* a un determinado polinomio implícito como aproximación al conjunto de datos. El beneficio de esta medida de calidad se reflejará en el siguiente hecho: en el proceso de ajuste, solo se obtendrán polinomios implícitos que cumplan ciertas propiedades topológicas deseadas, como se comprobará a continuación.

Primeramente, diremos que un polinomio implícito es *suave* en un punto x_i con respecto a un conjunto $\mathbf{X}$, si

$$\frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i \approx 1$$

Aquí, N_i es un vector normal al punto $\mathbf{x}_i$, normalizado. Puede obtenerse de los modelos físicos o estimarse de manera rápida a partir del conjunto de puntos $\mathbf{X}$, como en el método de Sahin [17]. Los vectores $\nabla f(\mathbf{x}_i)$ y N_i se consideran definidos (como vectores fila o columna) de forma tal que el producto tenga sentido.

Ahora, se define como *función de suavidad* de un polinomio implícito con respecto a todo el conjunto de puntos $\mathbf{X}$ de cardinalidad N, de la siguiente manera:

$$smooth(\mathbf{X}, Z(f_a)) = \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i \qquad (3)$$

Se hace notar que esta función toma valores entre -1 y 1, ya que tanto el vector gradiente, como el vector normal al punto $\mathbf{x}_i$, están normalizados.

En la Figura 2, se ilustra la idea intuitiva que está detrás de esta medida de proximidad.

Si el ajuste tiene buena calidad, el vector gradiente normalizado $\frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}$ debe corresponderse, en la medida de lo posible, con el vector normal N_i en el punto $\mathbf{x}_i$. Si esto ocurre con todos los puntos, entonces el ajuste será bueno.

2.3 Estrategia de penalización

La pregunta que se plantea ahora, es: ¿Cómo combinar las nociones de disimilitud y proximidad que expusimos en las secciones anteriores?

La estrategia que se utilizó se basa en la idea de penalizar la no-suavidad del polinomio implícito con respecto al conjunto de puntos $\mathbf{X}$. La no-suavidad con respecto a $\mathbf{X}$ se entiende como: $1 - smooth(\mathbf{X}, Z(f_a))$.

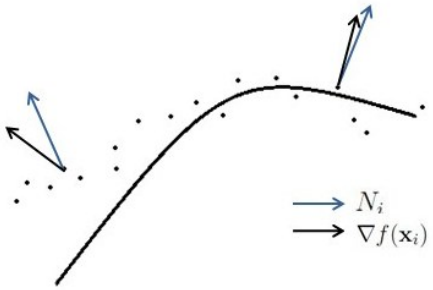

Figura 2. Medida de Suavidad.

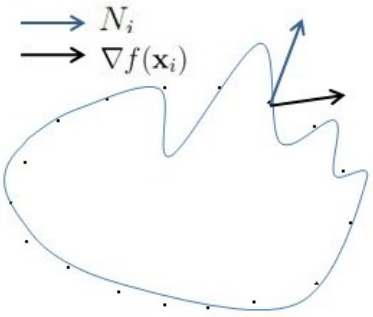

Figura 3. Sobreajuste. El polinomio implícito pasa cerca de todos los puntos del conjunto **X**, pero el ajuste es de mala calidad. La medida de suavidad se encuentra lejos de su valor ideal, la unidad.

Por tanto, la nueva función de distancia del proceso de ajuste será:

$$dist(\mathbf{X},Z(f_a)) = dissim(\mathbf{X},Z(f_a)) + \delta(1 - smooth(\mathbf{X},Z(f_a)))$$

$$dist(\mathbf{X},Z(f_a)) = \frac{1}{N}\sum_{i=1}^{N}\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta(1 - \frac{1}{N}\sum_{i=1}^{N}\frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}N_i)$$

$$dist(\mathbf{X},Z(f_a)) = \frac{1}{N}\sum_{i=1}^{N}\Big(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}N_i)\Big) \quad (4)$$

Aquí, el parámetro δ indica, cuán fuerte es la penalización. Esta medida será tomada como la función objetivo a minimizar en el algoritmo de ajuste:

$$\min_{f_a}\frac{1}{N}\sum_{i=1}^{N}\Big(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}N_i)\Big) \quad (5)$$

Notemos que es posible que el polinomio implícito *pase* por todos los puntos del conjunto **X**, pero que el valor de la función objetivo sea mayor que cero, como por ejemplo, cuando ocurre el efecto conocido como sobreajuste (Figura 3). Por consiguiente, la función de distancia aquí definida debe ser entendida en un sentido más amplio, como la distancia a un polinomio implícito ideal desde el punto de vista de sus propiedades geométricas.

En la próxima sección veremos, qué algoritmos se utilizarán en la búsqueda de un vector de coeficientes del polinomio que se ajuste mejor al conjunto de datos original.

3. Un Algoritmo de Ajuste Adaptativo

Habiendo definido convenientemente la función objetivo que se va a minimizar en el proceso de ajuste, se debe pasar a concretar, que algoritmo se utilizará para optimizar dicha función objetivo.

Se debe tener en cuenta, que no se conoce *a priori* el grado del polinomio implícito que mejor represente al conjunto de observaciones. Este grado, que determina la dimensión del vector de coeficientes del polinomio, debe estimarse durante el ajuste. Un algoritmo que lleve a cabo esta tarea se denominará Algoritmo de Ajuste Adaptativo.

En las próximas secciones, primero se detallará, cómo se realizará el ajuste con el grado del polinomio fijo, y luego este algoritmo se extenderá al caso de ajuste con polinomios de grados diferentes.

3.1 Ajuste de grado fijo

Para resolver el problema de ajuste del vector de coeficientes del polinomio implícito, se debe encontrar el mínimo de la función objetivo planteada en la sección anterior:

$$\min_{f_a}\frac{1}{N}\sum_{i=1}^{N}\Big(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}N_i)\Big)$$

Como se puede observar, esta función es no derivable por la presencia del módulo en el nominador.

Además, se debe destacar la naturaleza multimodal de la función objetivo propuesta, la cual puede tener numerosos mínimos locales. Esto resta utilidad al uso de los métodos exactos en este tipo de problemas, ya que éstos convergen de manera natural a óptimos locales. Es necesario tener métodos que realicen una mayor exploración del espacio de búsqueda.

Por estas razones, para el proceso de optimización, se decidió utilizar algoritmos metaheurísticos, y entre éstos, aquellos que fueron diseñados específicamente para la solución de problemas cuyo dominio es continuo. Siguiendo estos criterios, en la literatura se reportan [9, 19] dos metaheurísticas que tienen estas características: Optimización por Enjambres de Partículas y Evolución Diferencial, las cuales ya han sido empleadas previamente en problemas de ajuste de curvas [13].

3.1.1 Optimización mediante la metaheurística Enjambre de Partículas

La Metaheurística Optimización por Enjambres de Partículas (PSO, por sus siglas en ingles) utiliza la idea de mantener un grupo (enjambre) de N partículas desplazándose en un

espacio d-dimensional [9]. Cada partícula representa una solución al problema, y tiene una posición y una velocidad, la cual representa la dirección y tamaño de paso de su desplazamiento. Cuando las partículas se mueven, reajustan su velocidad. La nueva velocidad depende de la información proporcionada por el enjambre (de manera que la búsqueda se dirija a regiones del espacio, en las cuales los valores de la función objetivo son mejores), y de un parámetro χ, llamado factor de restricción, que permite acotar los valores de velocidad. De esta manera, el éxito de algunas partículas influye en el comportamiento de otras.

Un punto a favor de este algoritmo es el reducido número de parámetros que tiene: χ, el factor de restricción y N, el tamaño del enjambre. Vale la pena aclarar, que existen estudios que recomiendan darle al parámetro χ el valor 0,729, como valor recomendado para la mayoría de aplicaciones [7].

3.1.2 Optimización mediante la metaheurística Evolución Diferencial

La Metaheurística Evolución Diferencial (o DE, *Differential Evolution*), es un algoritmo evolutivo, que se puede ver como una variante del Algoritmo Genético [19]. Al igual que cualquier algoritmo genético, DE utiliza una población de N individuos, representados por vectores d-dimensionales. En esencia, el algoritmo genera nuevos individuos a partir de operaciones de suma y diferencia de vectores de la población actual, lo cual es un nuevo y original operador de mutación. Esta metaheurística en los últimos años ha tenido mucha aceptación al haberse aplicado a una gran variedad de problemas prácticos, y en especial a problemas continuos [14].

Además del tamaño de población N, DE utiliza dos parámetros: F, o factor de ponderación, permite manejar, cuanto se amplifica la variación obtenida a partir de la diferencia de vectores; y CR, constante de recombinación, la cual indica, que porciento de las componentes del vector mutado se tomarán en cuenta en los posibles vectores de la próxima generación.

Como se puede observar, los parámetros que tiene este algoritmo son igualmente escasos: F, CR y N, la cantidad de individuos en la población.

3.1.3 Sembrado de elementos en el algoritmo metaheurístico

Para acelerar la convergencia del algoritmo metaheurístico, así como encauzar la búsqueda ligeramente, es recomendable incluir en la población inicial una pequeña cantidad de elementos, los cuales se denominan *sembrados*, que tienen buenos valores de la función objetivo. Por tanto, se cree que su inclusión es beneficiosa para toda la población.

Para este sembrado inicial se pueden incluir elementos que sean soluciones de algoritmos exáctos, como el algoritmo de ajuste lineal clásico y el algoritmo 3L, obtener los cuales no es costoso computacionalmente.

Si estos elementos resultaran buenos desde el punto de vista de la medida de distancia utilizada, el algoritmo puede contribuir a mejorar estas soluciones desde el punto de vista de sus características topológicas.

Grado del polinomio	Dimensión, 2D	Dimensión, 3D
1	3	4
2	6	10
4	15	35
6	28	84
8	45	165
10	66	286
12	91	455
14	120	680
16	153	969

Tabla 1. Dimensión del problema a partir del grado del polinomio

3.2 Ajuste Adaptativo

En esta sección se responde la siguiente pregunta: ¿Qué criterios deben seguirse para encontrar el grado de un polinomio implícito que mejor represente a un cierto conjunto de observaciones? Para encontrar una respuesta, primero se analizarán de manera independiente varios aspectos de este problema.

3.2.1 Selección del grado de ajuste máximo

Grados de polinomios implícitos que se deben utilizar en la práctica

En los estudios que han abordado el problema de ajuste de polinomios implícitos, se observa que la mayoría los PI que se han utilizado en experimentos o ejemplos prácticos han sido de grado par [6, 17, 10, 28, 29].

En el trabajo de Taubin [26], encontramos la explicación del por que de este hecho. Se demuestra que el nivel cero de un polinomio en más de una variable *de grado impar* es siempre no acotado [2]. Por lo tanto, solo tiene sentido considerar polinomios implícitos de grado par, los cuales pueden (o no) ser acotados. Los conjuntos de observaciones representan siempre objetos finitos. Así, al menos en teoría, se garantiza la posibilidad de eliminar las curvas o superficies auxiliares que se generan durante el ajuste de polinomios implícitos.

Por otro lado, en estos mismos estudios, los grados de los PI más utilizados en la practica, varían entre 2 (elipsoide) y 10. El uso de los grados superiores (12, 14, 16 y 18), es de carácter excepcional. De hecho, el uso de estos grados hace que se pierda la ventaja de tener una representación compacta del objeto mediante el vector de coeficientes del PI (vea el Cuadro 1).

Cantidad de puntos en X y el grado del polinomio

El grado del polinomio implícito que se selecciona para el ajuste, determina la cantidad de coeficientes cuyos valores deben estimarse por el algoritmo. Esta cantidad de coeficientes, es la dimensión del problema. La relación entre el grado del PI y la dimensión, se presenta en el Cuadro 1.

En la práctica, no es recomendable que la dimensión del problema sea mayor que la cantidad de puntos en $\mathbf{X}$. En el algoritmo de ajuste clásico, incluso, es recomendable tener

[2] Por ejemplo, cualquier recta en el plano, o un plano en el espacio, son polinomios de primer grado, no acotados.

de 3 a 5 veces más puntos, que la cantidad de coeficientes del polinomio [6]. Sin embargo, ya en el algoritmo 3L, esta condición se relaja, hasta el punto de requerir *una cantidad de puntos ligeramente superior* a la cantidad de coeficientes.

Esta estrategia, de la que también haremos uso, establece, que no tiene sentido ajustar un conjunto de N puntos con un polinomio implícito de más de N grados de libertad.

3.2.2 Análisis del comportamiento de la Medida de Suavidad al aumentar el grado del PI

La clave para poder determinar el grado del polinomio implícito que mejor representa la complejidad del conjunto de datos dado, se puede encontrar en el comportamiento que tiene la medida de suavidad, al aumentar dicho grado. La Figura 4 nos ilustra este comportamiento para dos algoritmos de ajuste descritos en la sección 1, para dos juegos de datos.

Podemos observar que la medida de suavidad aumenta, hasta alcanzar cierto máximo, y luego desciende por el efecto del sobreajuste, o se mantiene constante con ligeras variaciones. Para otros juegos de datos, la medidad de suavidad se comporta de manera similar, aunque puede alcanzar el máximo fuera del rango de grados analizados (Figura 4).

Este hecho nos lleva a considerar el cambio de la medida de suavidad con respecto al grado del polinomio implícito, como un criterio de cercanía al grado óptimo, que represente mejor a un conjunto de observaciones. Si el aumento es demasiado pequeño o negativo (hay una disminución), alcanzamos el "mejor" grado buscado.

3.2.3 Resumen del Algoritmo de Ajuste Adaptativo

Teniendo en cuenta todo lo antes dicho, se propone el siguiente algoritmo de ajuste:

Algoritmo 1 Algoritmo de Ajuste Adaptativo

$G \leftarrow$ Grado máximo que se utilizará para el conjunto de puntos dado
$g \leftarrow 2$
while $g < G$ **do**
  $best\_elements_g \leftarrow FixedDegree(g)$
  if $g > 2$ and $smooth(best\_elements_g) - smooth(best\_elements_{g-1}) \leq \varepsilon$ **then**
    return $best\_elements_{g-1}$
  else
    $g \leftarrow g+2$
  end if
end while
return $best\_elements_g$

La condición de parada de este algoritmo es la no mejora de la medida de suavidad en un cierto ε con el aumento del grado. Se hace notar que con $\varepsilon = 0$, esto equivale a una disminución de dicha medida.

Una ventaja de utilizar la suavidad en la condición de parada del algoritmo, es que tiene una interpretación geométrica inmediata. Es el promedio de los valores de los cosenos de los ángulos entre entre los vectores gradiente y normal estimado, en cada punto. El valor 0,01 para ε representa un acercamiento en, aproximadamente, un grado, entre estos vectores.

Las siguientes características del Algoritmo de Ajuste Adaptativo pueden ser beneficiosas para las aplicaciones:

- Permite encontrar *sin ningún conocimiento previo de la complejidad del conjunto de datos*, el grado del polinomio implícito que será necesario para un buen ajuste.
- Introduce la variabilidad en las soluciones, ofreciendo un conjunto de éstas al usuario final. Con esto se saca provecho de la naturaleza multimodal [3] del problema de ajuste de polinomios implícitos, lo cual es una clara ventaja frente a algoritmos exactos. En un entorno interactivo, el usuario podría tener opciones, en caso de no estar satisfecho con la "mejor" solución.
- Utiliza y mejora la calidad de los PI que se obtuvieron como solución de los algoritmos clásicos, lo cual se puede ver como un proceso de postoptimización. Las metaheursístias pueden generar nuevas soluciones que no están relacionadas con las soluciones clásicas, si éstas tienen características geométricas deseadas.

4. Resultados Experimentales

Antes de presentar los resultados experimentales debemos aclarar varias cuestiones relativas a la aplicación de los algoritmos presentados en la práctica.

Un aspecto que se debe tener en cuenta al aplicar todos los algoritmos de ajuste, es la necesidad de centrar en el origen el conjunto de puntos $\mathbf{X}$ que se va a utilizar. Además, a $\mathbf{X}$ se le debe aplicar un escalado (por ejemplo, dividiendo los puntos por su distancia promedio con el origen). El resultado de estas operaciones es invariante ante transformaciones Euclidianas en los datos [21].

Estas transformaciones consiguen evitar problemas numéricos en los algoritmos de ajuste.

En los experimentos se evaluarán tanto la validez de la función objetivo propuesta para el problema en cuestión, como el rendimiento de los algoritmos de ajuste de grado fijo y variable. Estos algoritmos deben ser capaces de optimizar la función objetivo, y además dar como resultado curvas y superficies interpretables, geométricamente similares a los objetos en 2D y 3D que sirvan de modelos.

4.1 Juegos de datos

Los modelos 3D y 2D para evaluar la validez de los algoritmos expuestos se seleccionaron de la siguiente manera.

Como modelos 3D, se tomaron nubes de puntos de repositorios conocidos ampliamente, como el *Stanford 3D Scanning Repository*[2], entre otros (Figura 6). En particular, se tomaron los modelos:

- Manzana
- Patico de Hule

[3] Un problema multimodal posee muchos óptimos locales, que pueden tener valores cercanos de la función objetivo.

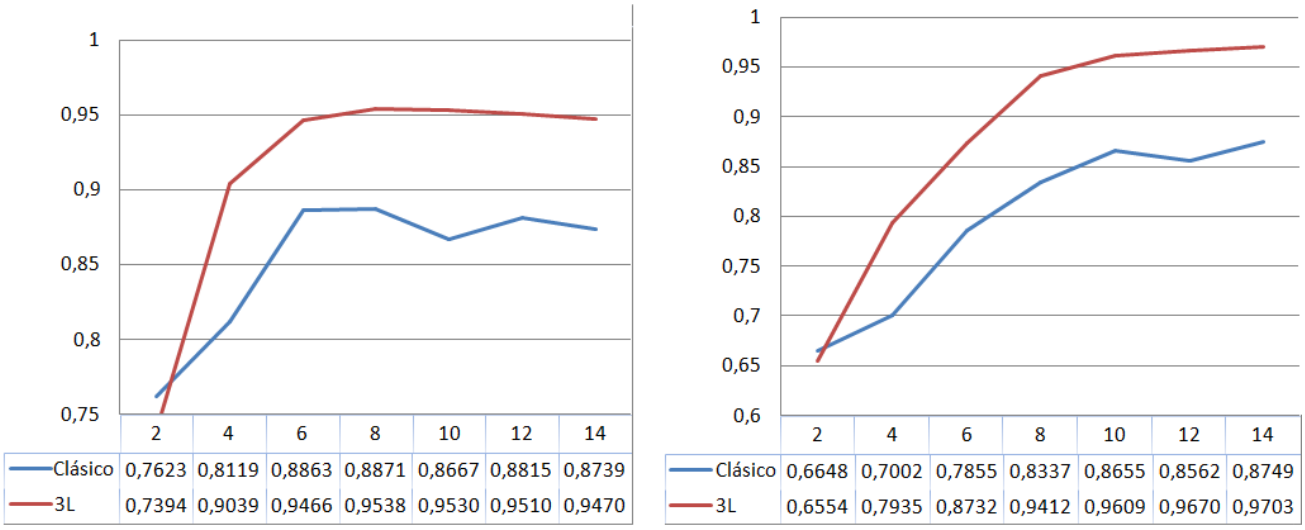

Figura 4. Comportamiento de la Medida de Suavidad al aumentar el grado del PI para los juegos de datos (a) "Bota", (b) "Caballo" (*Large Geometric Models Archive* del *Georgia Institute of Technology*[1])

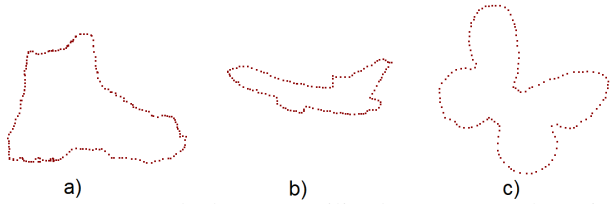

Figura 5. Juegos de datos 2D utilizados. (a) Bota, (b) Avion, (c) Mariposa.

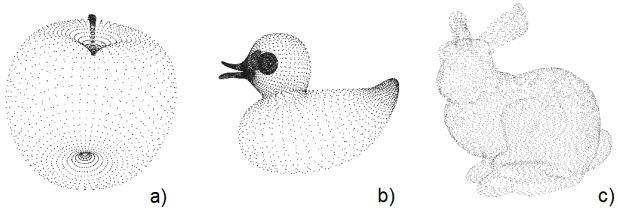

Figura 6. Juegos de datos 3D utilizados. (a) Manzana, (b) Patico de hule, (c) Conejo.

- El Conejo de Stanford ("*Stanford Bunny*")

En el caso de los modelos 2D, se escogieron figuras planas (Figura 5) que han aparecido con frecuencia en los distintos estudios relativos a los polinomios implícitos[29, 21, 6]. Lamentablemente, no existe consenso sobre juegos de datos "ideales" para el ajuste en dos dimensiones, ni repositorios que los contengan. Por otra parte, los juegos da datos en 2D son fácilmente recreables por los investigadores en el área.

4.2 Parámetros de los algoritmos utilizados

En los experimentos realizados, se utilizaron los siguientes parámetros.

Para las metaheurísticas DE y PSO, el tamaño de la población (o enjambre), se fijó en cuatro veces la dimensión del problema [12]. Como condición de parada de ambas metaheurísticas, se utilizó el arribo a una cierta cantidad de iteraciones. Se utilizaron 500 iteraciones en la mayoría de los casos, excepto en la comparación de las metaheurísticas, donde se utilizaron 200 iteraciones en el caso 3D.

En la metaheurística Enjambre de Partículas, se tomó $\chi = 0{,}729$, como se sugiere en [7].

En la metaheurística Evolución Diferencial, se asumió $F = 0{,}7$ (factor de ponderación), $CR = 0{,}9$ (constante de recombinación), como se recomienda en [20].

El parámetro de penalización δ introducido en la función objetivo (5), se estimará en la próxima sección.

Por último, en el caso 3D se tomaron fracciones de las cantidades de puntos en los modelos originales, pues las cardinalidades de estos conjuntos son muy grandes, llegando a tener 35947 puntos el juego de datos "El Conejo de Stanford". Esto se debe a que los modelos empleados se utilizan con frecuencia en investigaciones en el campo de los gráficos por computadora, donde la precisión local es fundamental.

4.3 Ajuste del parámetro δ

El parámetro δ introducido en el algoritmo de ajuste, determina cuán fuerte es la penalización por tener un valor demasiado bajo de la medida de suavidad.

El ajuste de este parámetro es imprescindible para obtener resultados que puedan ser utilizados e interpretados correctamente, ya que un valor demasiado bajo puede eliminar el necesario efecto de la penalización, y un valor demasiado alto puede hacer que el polinomio implícito quede lejos del conjunto original de observaciones.

Este hecho se ilustra en la Figura 7. Se puede observar el comportamiento inestable del algoritmo para valores pequeños de δ, así como resultados poco precisos con respecto al conjunto original de puntos, para valores demasiado gran-

des. También se observa, que existe un rango bastante amplio de valores "aceptables" de δ.

4.4 Análisis del Ajuste de Objetos 2D y 3D por el Algoritmo Adaptativo

A continuación se presentarán los resultados del algoritmo de ajuste adaptativo para los distintos juegos de datos.

4.4.1 Ajuste de curvas en 2D

Seguidamente se muestran las corridas del algoritmo adaptativo para los juegos de datos en 2D (Figuras 8, 9, 10), comparando los resultados con los algoritmos 3L y Lineal Clásico.

Como se observa, el Algoritmo Adaptativo puede determinar de manera razonable el grado del polinomio implícito que se necesita para obtener un ajuste "interpretable".

4.4.2 Ajuste de superficies en 3D

Al igual que en 2D, se realizaron corridas experimentales del Algoritmo de Ajuste Adaptativo, para los juegos de datos en 3D. En las Figuras 11, 12, 13 se muestran los ajustes conseguidos, así como los resultados obtenidos por los algoritmos 3L y Lineal Clásico, para los mismos juegos de datos.

Se debe señalar que el algoritmo es capaz de representar objetos complejos con polinomios implícitos de grado relativamente bajo (2, 4, 6, 8).

Además, se observa que el valor 0,01 para ε aporta buenas soluciones en la mayoría de los casos.

5. Conclusiones

Los polinomios implícitos (PI) no son el esquema de representaión de mayor precisión. Sin embargo, son muy atractivos para aplicaciones que requieran registrar de manera compacta datos de un complejo objeto del mundo real, para posteriormente poder realizar un proceso de reconocimiento del mismo [4], o de otros objetos que se correspondan con el mismo patrón. Son robustos al ruido y la oclusión en los datos [28].

Con el objetivo de hallar un buen polinomio implícito que represente a un conjunto de puntos dado, se realiza un proceso de ajuste, para el cual en la literatura consultada existen disímiles algoritmos. La gran mayoría de estos algoritmos presupone el conocimiento del grado del PI que mejor representa a los puntos. Este trabajo plantea una alternativa a estos algoritmos de ajuste.

En primer lugar, se define una función objetivo que se utilizará durante el ajuste, y que se distingue por incluir una penalización de propiedades geométricas no deseadas (no-suavidad) en el PI.

Luego, se plantea un algoritmo de ajuste heurístico, el cual es capaz de determinar el grado del polinomio necesario para la representación del conjunto de datos. El algoritmo se basa en la idea de ir incrementando paulatinamente el grado del PI, mientras haya una mejora en la suavidad de las soluciones.

Este método es beneficioso frente a otros por tres razones: permite encontrar automáticamente el grado del polinomio, puede ofrecer un conjunto de soluciones en contextos donde se requiera variabilidad (alternativas) durante el ajuste, y permite postoptimizar soluciones de los algoritmos de ajuste conocidos (lineal clásico y 3L), con el fin de obtener soluciones de mejor calidad.

Los experimentos realizados confirman la validez del enfoque expuesto en juegos de datos 2D y 3D escogidos, dado que los ajustes conseguidos por el algoritmo propuesto son interpretables.

Referencias

[1] Large Geometric Models Archive. `http://www.cc.gatech.edu/projects/large_models/`, consultado el 2015-05-05.

[2] The Stanford 3D Scanning Repository. `http://graphics.stanford.edu/data/3Dscanrep/`, consultado el 2015-05-05.

[3] A. Ben-Israel. *Generalized Inverses: Theory and Applications*. Second edition, 2001.

[4] H. Ben-Yaacov. Recognition of 3d objects based on implicit polynomials. *Irwin and Joan Jacobs Center for Communication and Information Technologies*, 2009.

[5] M. Berger, J. Levine, L. Gustavo Nonato, G. Taubin, and C. T. Silva. An end-to-end framework for evaluating surface reconstruction. *Scientific Computing and Imaging Institute*, 2011.

[6] M. M. Blane. The 3l algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:298–313, March 2000.

[7] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation*, 1:84–88, 2000.

[8] G. Golub. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks, Proceedings*, 4, December 1995.

[10] Z. Landa. 2d object description and recognition based on contour matching by implicit polynomials. 2006.

[11] V. Mederos and J. Estrada. A new algorithm to compute the euclidean distance from a point to a conic. *Investigación Operacional*, 23, 2002.

[12] M. Pedersen. Good parameters for particle swarm optimization. 2010.

[13] M.J. Polo-Corpa. Curve fitting using heuristics and bio-inspired optimization algorithms for experimental data processing in chemistry. *Chemometrics and Intelligent Laboratory Systems*, 96:34–42, 2009.

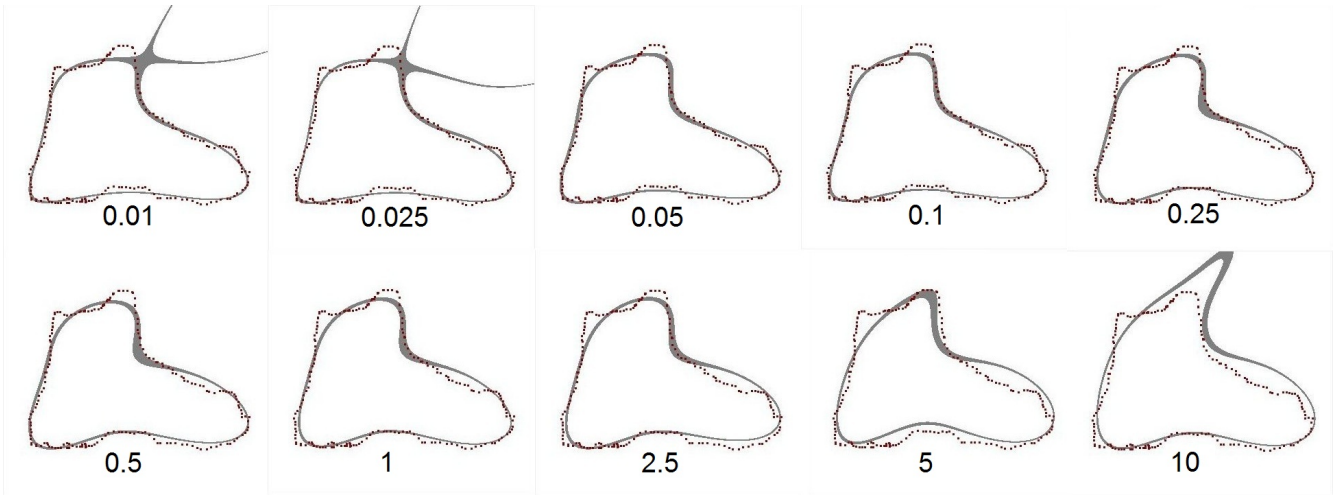

Figura 7. Juego de datos "Bota": ajuste del parámetro δ. Grado 4. Debajo de cada imagen se muestra el valor δ utilizado.

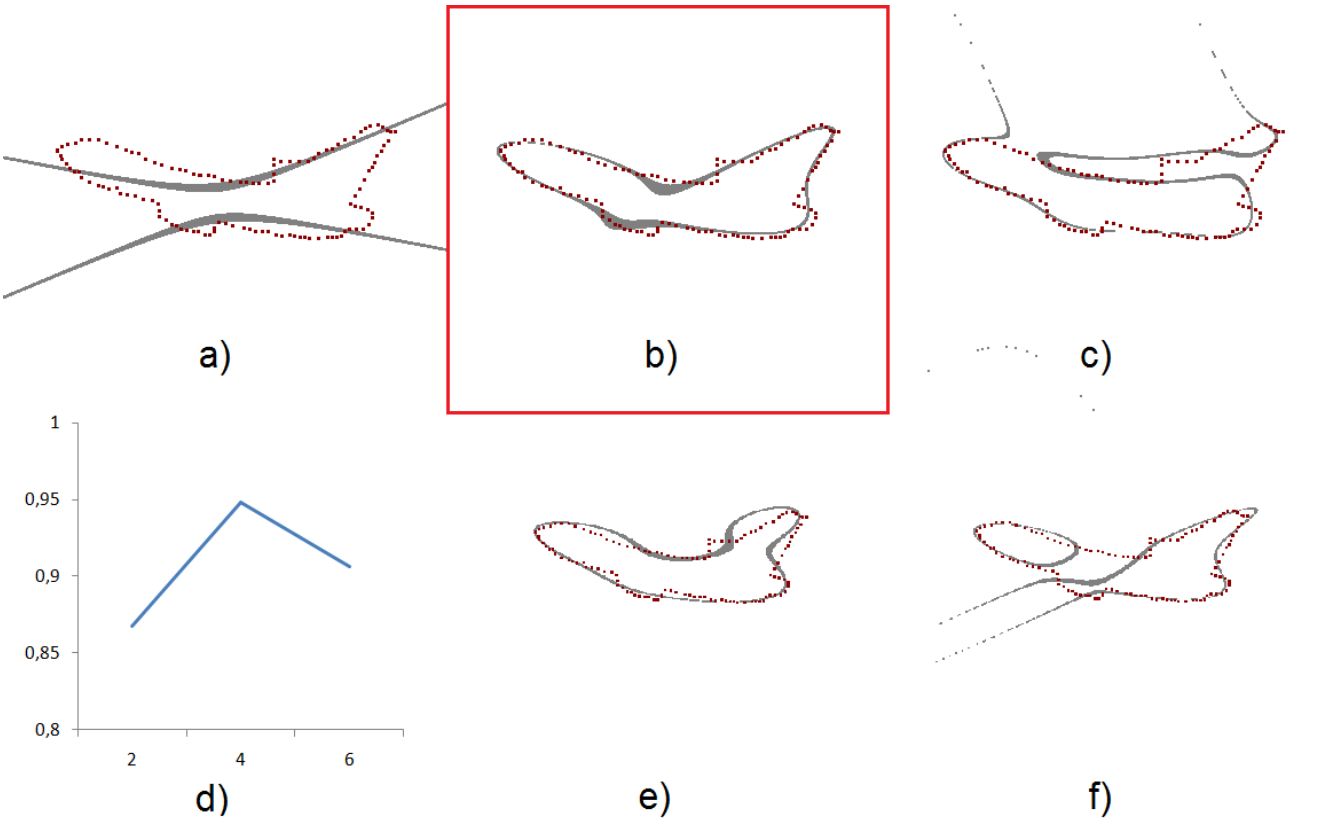

Figura 8. Juego de datos "Avión": corrida del Algoritmo Adaptativo: a) Ajuste de Grado 2, b) Ajuste de Grado 4 (grado óptimo), c) Ajuste de Grado 6, d) Evolución de la medida de suavidad durante la corrida, e) Ajuste 3L de grado 4, f) Ajuste Lineal Clásico de grado 4.

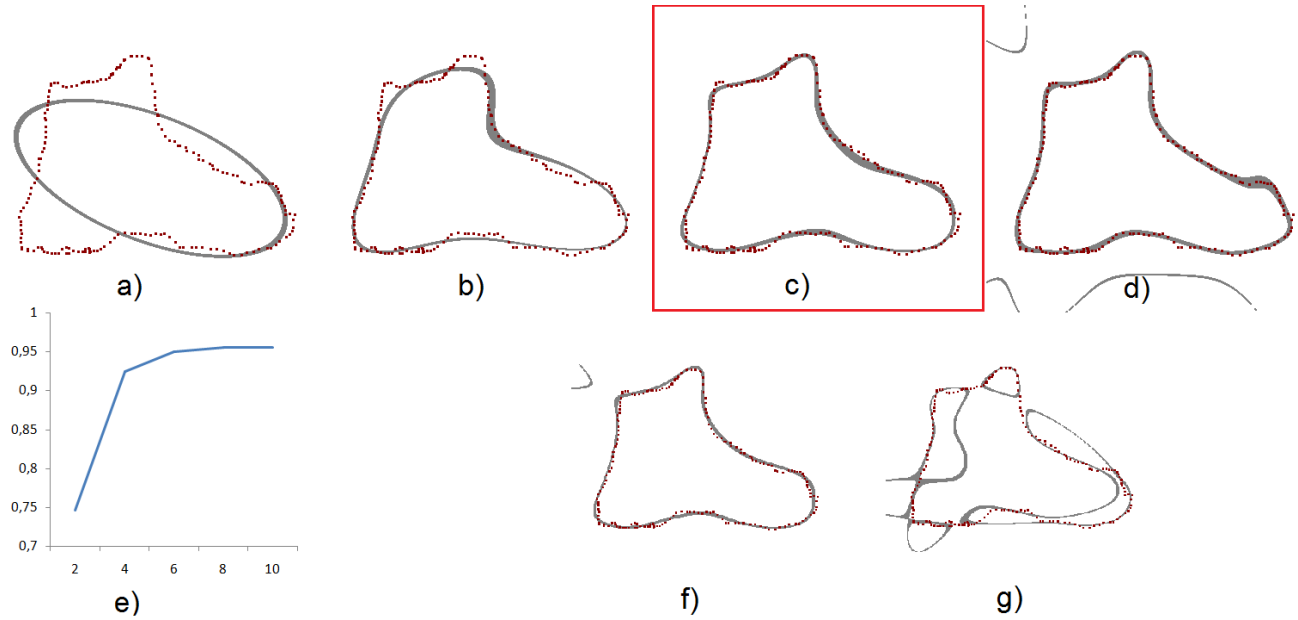

Figura 9. Juego de datos "Bota": corrida del Algoritmo Adaptativo: a) Ajuste de Grado 2, b) Ajuste de Grado 4, c) Ajuste de Grado 6 (grado óptimo para $\varepsilon = 0{,}01$), d) Ajuste de Grado 8, e) Evolución de la medida de suavidad durante la corrida, f) Ajuste 3L de grado 6, g) Ajuste Lineal Clásico de grado 6.

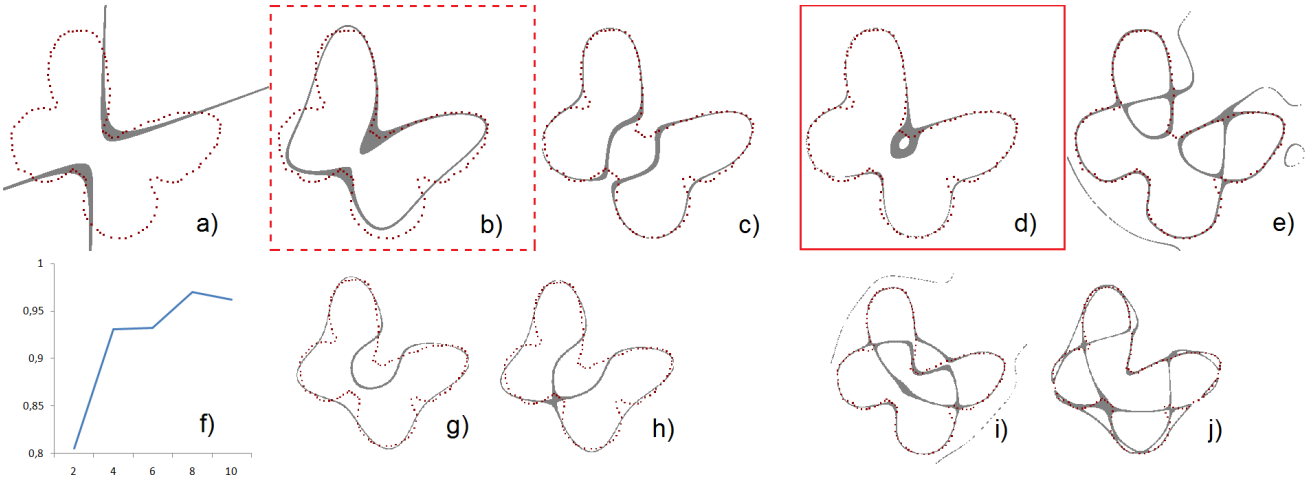

Figura 10. Juego de datos "Mariposa": corrida del Algoritmo Adaptativo: a) Ajuste de Grado 2, b) Ajuste de Grado 4 (grado óptimo para $\varepsilon = 0{,}01$), c) Ajuste de Grado 6, d) Ajuste de Grado 8 (grado óptimo para $\varepsilon = 0$), e) Ajuste de Grado 10, f) Evolución de la medida de suavidad durante la corrida, g) Ajuste 3L de grado 4, h) Ajuste Lineal Clásico de grado 4, i) Ajuste 3L de grado 8, j) Ajuste Lineal Clásico de grado 8.

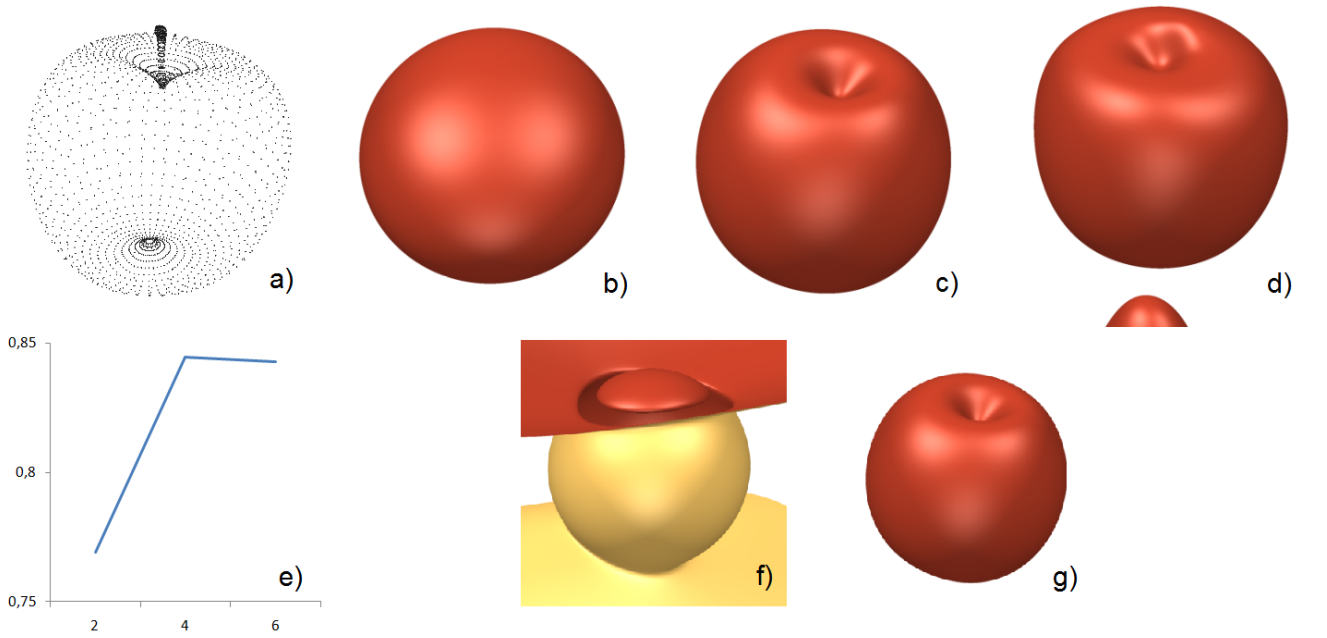

Figura 11. Juego de datos "Manzana": corrida del Algoritmo Adaptativo: a) Nube de puntos original, b) Ajuste de Grado 2, c) Ajuste de Grado 4 (grado óptimo), d) Ajuste de Grado 6, e) Evolución de la medida de suavidad durante la corrida, f) Ajuste Lineal Clásico de grado 4, g) Ajuste 3L de grado 4.

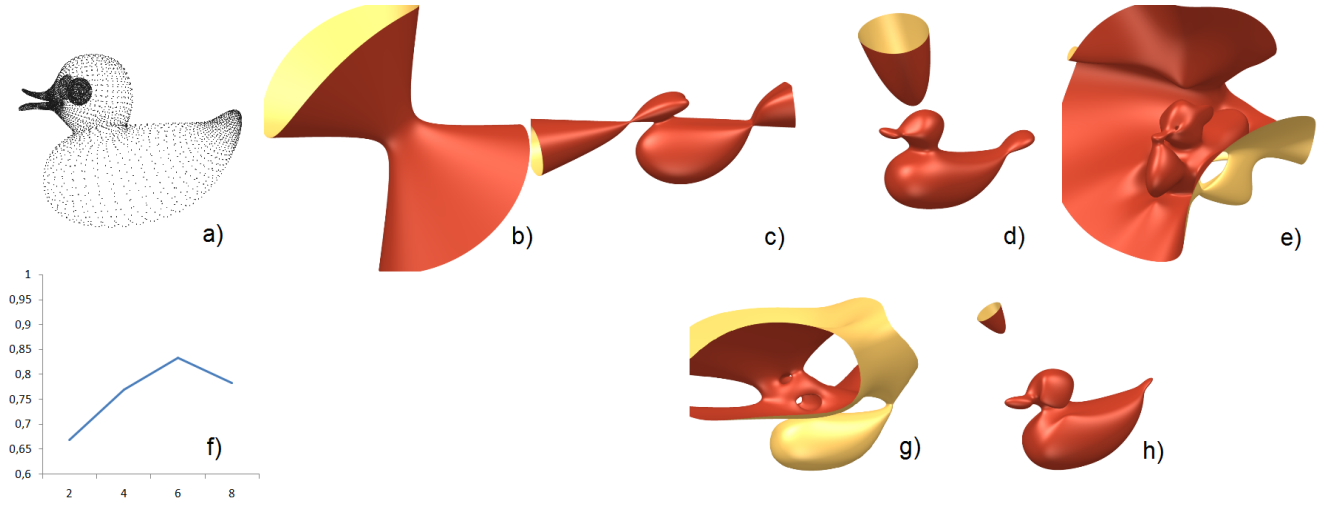

Figura 12. Juego de datos "Patico de hule": corrida del Algoritmo Adaptativo: a) Nube de puntos original, b) Ajuste de Grado 2, c) Ajuste de Grado 4, d) Ajuste de Grado 6 (grado óptimo), e) Ajuste de Grado 8, f) Evolución de la medida de suavidad durante la corrida, g) Ajuste Lineal Clásico de grado 6, h) Ajuste 3L de grado 6.

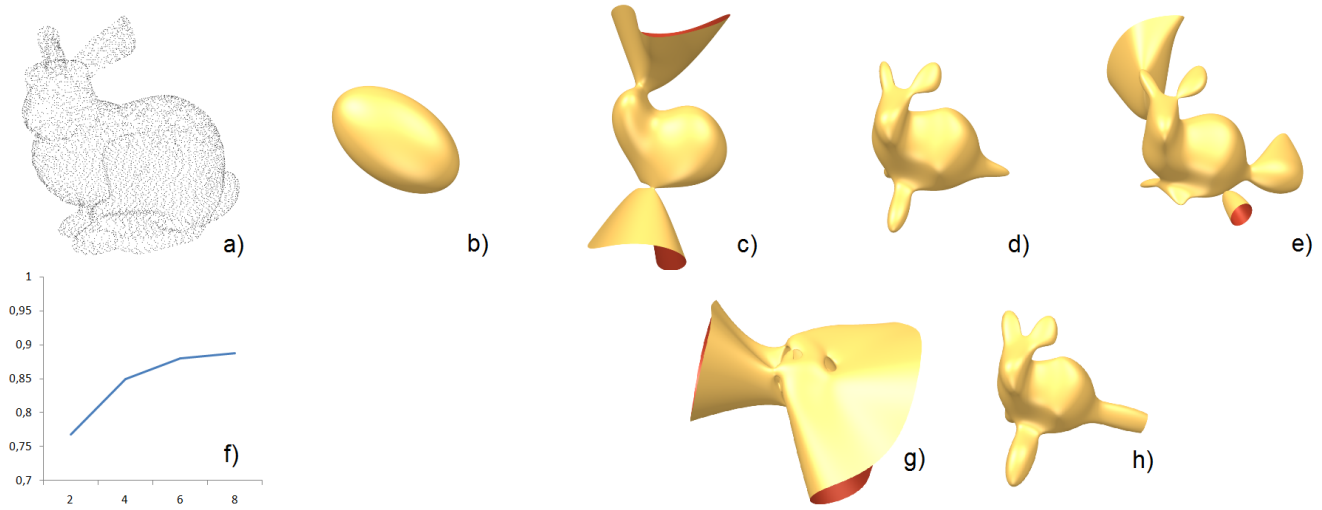

Figura 13. Juego de datos "Conejo": corrida del Algoritmo Adaptativo: a) Nube de puntos original, b) Ajuste de Grado 2, c) Ajuste de Grado 4, d) Ajuste de Grado 6 (grado óptimo para $\varepsilon = 0{,}01$), e) Ajuste de Grado 8, f) Evolución de la medida de suavidad durante la corrida, g) Ajuste Lineal Clásico de grado 6, h) Ajuste 3L de grado 6.

[14] K. Price and R. Storn. *Differential Evolution - A Practical Approach to Global Optimization*. Springer, 2006.

[15] M. Rouhani. Implicit polynomial representation through a fast fitting error estimation. *IEEE Transactions on Image Processing*, April 2012.

[16] M. Rouhani. The richer representation the better registration. *IEEE Transactions on Image Processing*, 22, December 2013.

[17] T. Sahin. Fitting globally stabilized algebraic surfaces to range data. *IEEE International Conference on Computer Vision*, 2, 2005.

[18] P. Schenzel. On the interactive visualization of implicit surfaces. *Martin-Luther University Halle Press*, September 2012.

[19] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 1997.

[20] E. Talbi. *Metaheuristics from design to implementation*. John Wiley & Sons, Inc., 2009.

[21] T. Tasdizen. Improving the stability of algebraic curves for applications. *IEEE Transactions on Image Processing*, 9:405–416, March 2000.

[22] G. Taubin. Nonplanar curve and surface estimation in 3-space. *International Conference on Robotics and Automation, Proceedings, IEEE*, pages 644–645, April 1988.

[23] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *Pattern Analysis and Machine Intelligence, IEEE*, 13:1115–1138, November 1991.

[24] G. Taubin. Object recognition based on moment (or algebraic) invariants. *Geometric Invariance in Computer Vision*, pages 375–397, 1992.

[25] G. Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics*, 13, January 1994.

[26] G. Taubin. Parameterized families of polinomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, March 1994.

[27] G. Taubin and D. Cooper. *Symbolic and Numerical Computation for Artificial Intelligence*. Academic Press, 1992.

[28] B. Zheng. *2D Curve and 3D Surface Representation using Implicit Polynomial and its Applications*. PhD thesis, University of Tokyo, June 2008.

[29] B. Zheng. An adaptive and stable method for fitting implicit polynomial curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 2010.

[30] B. Zheng. Breast MR image fusion by deformable implicit polynomial (dip). *IPSJ Transactions on Computer Vision and Applications*, 5, July 2013.