Doctoral Thesis

"Obtaining Free-Form Shapes With Global-Support Functions Via Bat Algorithm"

Author:

Marta Collantes Viaña

Directors:

Dr. Andrés Iglesias Prieto
Dra. Akemi Gálvez Tomida

Departament of Applied Mathematics
and Computational Sciences
UNIVERSITY OF CANTABRIA

Santander, February 2016

A mis padres,
a mis hijos y a mi marido, mi luz en la oscuridad.

Agradecimientos

En primer lugar quiero dar las gracias a mis padres, Rosario Viaña Vélez y Tomás Collantes Terán, no sólo por la educación y formación académica que me han otorgado, sino, por la comprensión y el apoyo incondicional recibido.

Gracias a ti mamá por tu esfuerzo para hacer de mi hermana y de mí mujeres profesionales e independientes, que es lo que siempre pretendiste, ¡enhorabuena!, creo que lo conseguiste con creces.

Gracias por animarme siempre a seguir adelante en momentos desfavorables, por confiar en mí, por darme los mejores consejos y sobre todo por escucharme siempre. De corazón, gracias.

En segundo lugar, quiero agradecer a Jose Manuel Gutiérrez, Begoña Sánchez y Andrés Iglesias la confianza depositada en mí, brindándome la oportunidad de trabajar en el departamento de Matemática Aplicada y Ciencias de la Computación de la Universidad de Cantabria, un sueño hecho realidad.

Gracias a Begoña Sánchez y Joaquín García por su compañerismo y por su cariño. Haber trabajado con vosotros ha sido muy gratificante y muy enriquecedor.

Gracias a Ángel Cobo por haber contado conmigo para formar parte de diferentes proyectos.

Agradecida eternamente a mi co-director de tesis Andrés Iglesias, quien ha hecho posible la existencia de esta tesis. Gracias por guiarme, aconsejarme y animarme siempre.

Gracias también a mi co-directora de Akemi Gálvez por su gran ayuda y sus aportaciones para la elaboración de la Tesis.

Por último, y no por ello menos importante, quiero agradecer a mi marido, Santiago Pérez Díaz, mi gran compañero de viaje en esta vida, su confianza, su paciencia y su implicación a lo largo de estos años. Sin ti, nada de esto hubiera sido posible. Ha sido un placer, y lo sigue siendo, compartir contigo momentos malos y momentos inolvidables. Juntos lo hemos logrado. Gracias por estar siempre ahí.

Y como no, quiero mencionar a dos seres maravillosos, mis joyas más valiosas, Alejandro y Martín, mis hijos. Gracias por comprender de alguna manera, el tiempo que os he robado.

Marta Collantes Viaña

Santander, Febrero de 2016.

El trabajo presentado en esta tesis ha sido financiado por dos proyectos del Plan Nacional de Investigación, Desarrollo e Innovación:

- Proyecto del Programa Nacional de Tecnologías Informáticas *"Inteligencia Artificial para Modelado Geométrico y Gráficos por Computador"* (Ref. TIN2006-13615), de la Dirección General de Investigación del Ministerio de Educación y Ciencia.

- Proyecto del Programa Nacional de Tecnologías Informáticas *"Metaheurísticas para Reconstrucción Automática de Curvas y Superficies de Forma Libre en Ingeniería Inversa"* (Ref. TIN2012-30768), de la Dirección General de Investigación del Ministerio de Economía y Competitividad.

Esta tesis se ha realizado en el Departamento de Matemática Aplicada y Ciencias de la Computación de la Universidad de Cantabria, contando para su realización con apoyo económico y logístico tanto del Departamento como de la Universidad.

Contents

List of Figures

Lista of Tables

Part I

RESUMEN GENERAL

Chapter 1

Resumen General

Este primer capítulo de la tesis pretende dar respuesta a uno de los requerimientos básicos del formato de tesis doctoral de la Universidad de Cantabria para aquellas tesis escritas en lenguas distintas de la española. Dicho requerimiento establece que:

> *En el caso de Tesis escritas en una lengua distinta a la española, además de lo indicado en el párrafo anterior, deberán contener un resumen global, en español, de los resultados y una discusión de los mismos, y en el que queden plasmadas las conclusiones que podrían extraerse de la línea de investigación, así como los posibles desarrollos futuros de dichas investigaciones. También contendrá, en su caso, los procedimientos y materiales empleados en las investigaciones objeto que hayan servido de base para la elaboración de la Tesis. El resumen quedará encuadernado como parte constitutiva de la Tesis.*

La estructura de este resumen general es como sigue: en primer lugar, se describe la motivación de este trabajo en la Sección 1.1. A continuación, se determina el objetivo principal de la tesis en la Sección 1.2. Seguidamente, se describe la metodología seguida en este trabajo en la Sección 1.3. La Sección 1.4 explica la estructura de esta tesis, consistente en 6 partes y 11 capítulos. La sección 1.5 indica los principales resultados alcanzados, las principales conclusiones de esta línea de investigación y las publicaciones a las que ha dado lugar. Finalmente, la Sección 1.6 indica algunas ideas sobre trabajos futuros en el campo. El capítulo se cierra con la bibliografía más relevante utilizada en este trabajo. Dicha bibliografía será luego ampliada en el grueso de la tesis, escrita en inglés.

1.1 Introducción del Problema

Esta tesis se centra en el problema de la reconstrucción de curvas y superficies a partir de una colección dada de puntos dato. Básicamente, en este problema asumimos que el usuario dispone de una colección de puntos 2D o 3D que se presupone describen la forma de una curva o una superficie. Idealmente, el objetivo final es obtener dicha curva o superficie. Sin embargo, muy a menudo obtener dicha curva o superficie no es viable, porque los puntos dato están muy frecuentemente afectados por ruido introducido durante el proceso de obtención de los mismos (este fenómeno es especialmente notable en los casos de aplicaciones del mundo real) o por otros problemas, como un muestreado irregular u otros. Por ello, se suele considerar la alternativa de obtener un meta-modelo en la forma de una curva o superficie de ajuste.

Dicho ajuste suele realizarse siguiendo un modelo de interpolación o de aproximación, en función de si se impone o no que la curva o superficie deba pasar necesariamente por todos los puntos dato (caso de la interpolación) o solamente cerca de ellos (esquema de aproximación). En esta tesis nos decantamos por el esquema de aproximación, dado que funciona mejor para los problemas reales de la industria, usualmente afectados por ruido en los datos y otros problemas que provocan que los datos no sean totalmente fiables y sea mas recomendable capturar la tendencia general de los datos que ajustar todos los datos con precisión extrema.

Los esquemas de aproximación se basan en escojer una función o familia de funciones más o menos adecuada que serán utilizadas como funciones aproximantes o de ajuste. En general, la función es única, pero suele estar expresada como una combinación lineal de un conjunto de funciones básicas, cuya condición de partida es que sean linealmente independientes. Esto implica que serán una base del espacio vectorial de todas las combinaciones lineales de dichas funciones, de ahí la denominación de funciones básicas. El caso más típico y más sencillo de entender de funciones básicas sería un polinomio de grado n en una única variable, $p_n(x)$, expresado como:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

En este caso, la función aproximante $p_n(x)$ está descrita como una combinación lineal de los monomios $\{1, x, x^2, \ldots, x^n\}$. Como es sabido, dado que dichos monomios son linealmente independientes, forman una base del espacio vectorial de las funciones polinómicas de grado menor o igual que n con coeficientes reales, llamada la base canónica. Los elementos de dicha base (es decir, los distintos monomios) reciben el nombre de funciones básicas de dicha base.

Evidentemente, existen otras muchas bases y funciones básicas que pueden utilizarse para este propósito. Atendiendo al tipo de soporte de las funciones,

existen dos tipos: funciones de soporte global y funciones de soporte local. Recordamos que el soporte de una función es el subconjunto de puntos del dominio sobre los que la función no se anula. Por ejemplo, el soporte de la base canónica resulta ser: $\mathbb{R} - \{0\}$. Por ello, si realizamos la aproximación de una nube de puntos en el intervalo $[1, 2]$ usando la base canónica, estamos ante un caso de base de funciones cuyo soporte coincide en todos los casos con el del dominio del problema (a su vez, subconjunto del dominio de la función). Diremos entonces que esta base tiene soporte global (evidentemente, las funciones de soporte global son aquellas en las que dicha condición no se cumple). En esta tesis, nos centraremos en todo momento en funciones de soporte global.

El problema de la obtención de formas libres mediante funciones de soporte global tiene una larga tradición en el mundo de los gráficos por computador, con numerosas aplicaciones en muy diversos campos. Dos ejemplos clásicos de dominios de aplicación de este problema se dan en el diseño geométrico asistido por computador (CAGD) (Farin, 2002; Hoschek and Lasser, 1993) y en diseño y manufactura asistidos por computador (CAD/ CAM), (Alhanaty and Bercovier, 2001) en los cuales ese tipo de funciones han sido ampliamente utilizadas para representar y modelar un gran número de elementos geométricos de diseño, tanto en el entorno industrial (piezas de carrocerías de automóviles, fuselajes de aviones, casos de buques, etc.) (ver, por ejemplo, (Patrikalakis and Maekawa, 2002; Barnhill, 1992; Farin, 2002)) como en otros campos como la arquitectura (modelos de paneles, modelos de aproximación triangular, etc.), arqueología (reconstrucción, archivado digital y restauración de restos arqueológicos (Levoy et al., 2000)), medicina (reconstrucción y visualización por computador de estructuras como los huesos y órganos internos de nuestro organismo mediante técnicas como la tomografía axial computerizada - TC - o las imágenes por resonancia magnética - MRI -, las cuales tienen como base bien secciones transversales de la superficie de dichos órganos (Meyers et al., 1992), bien nubes de puntos tridimensionales, bien una combinación de ambos tipos de datos), ingeniería biomédica (generación de prótesis e implantes personalizados y otros elementos biomecánicos), en ingeniería inversa en campos como la industria de la manufactura, en procesos como el mecanizado por control numérico (CNC) asistido por computador (Varadi and Martin, 2002; Pottmann et al., 2005), en problemas de aproximación de funciones (Rice, 1969), en el ajuste con curvas y superficies spline (Dierckx, 1993), en la creación de tipografías para el mercado editorial (como las tipografías vectoriales de fuentes del tipo *TrueType* por ejemplo), y muchos otros.

El campo de interés de este proyecto ha recibido un fuerte impulso recientemente con la popularización de los escáneres 3D, tanto de tipo óptico (escáneres láser) como de tipo táctil, y la posibilidad de generar físicamente

el modelo final obtenido mediante diversas técnicas de impresión 3D (tales como la estereolitografía, el sintetizado selectivo por láser, o el modelado por deposición fusionada), con aplicaciones en procesos como la generación de objetos personalizados (como prótesis médicas, diversos objetos de consumo, etc.) hasta la obtención de modelos iniciales del proceso de diseño, como ocurre generalmente en diversos problemas que requieren o involucran procesos de prototipado rápido.

Otros desarrollos recientes se han dado en el campo de la animación por computador y los videojuegos, donde estas técnicas han sido aplicadas a la mejora de las técnicas de interpolación para procesos de animación del movimiento mediante cinemática inversa (Mukai, 2012) y su aplicación a videojuegos de última generación, como *Final Fantasy XVI*, de *Square Enix*, actualmente en desarrollo y con lanzamiento previsto para 2017-18.

1.2 Objetivo de la Tesis

Este proyecto de tesis pretende analizar en detalle el problema de la obtención de formas, partiendo de los desarrollos iniciales descritos en (Pratt, 1987; Schmitt et al., 1986; Sclaroff and Pentland, 1991; Gu and Yan, 1995; Hoffmann, 2005; Barhak and Fischer, 2001). Para ello nos enfocaremos, como se ha mencionado anteriormente, en el caso de las funciones de soporte global, las cuales presentan propiedades tales como el control global de la forma, en claro contraste con las funciones de soporte local. Estas funciones serán utilizadas para realizar la aproximación de un conjunto (posiblemente masivo) de puntos 2D o 3D, que asumimos como datos de entrada del problema. Asumimos también que dichos datos están afectados por ruido y han sido muestreados irregularmente, generando por tanto topologías irregulares, de forma que reproduzcan las condiciones que se dan usualmente en las aplicaciones prácticas del mundo real.

Para dicho objetivo, planteamos la aplicación de una técnica metaheurística (Engelbretch, 1993) a fin de resolver el problema de minimización del error de aproximación de la función objetivo. El sistema de ecuaciones resultante de este proceso de minimización estará sobre-determinado, pues esperamos obtener un modelo con muchos menos parámetros que el problema de partida. Por ello, la función objetivo vendrá dada por un problema de mínimos cuadrados, por lo cual resultará, en general, continua en sus parámetros, nonlineal, multimodal (es decir, con varios posibles óptimos, tanto locales como globales) y de alta dimensión.

1.3 Metodología de la Tesis

Nuestra propuesta estará basada en el algoritmo bat (Yang, 2010; Yang and Gandomi, 2012), un potente método metaheurístico inspirado en la naturaleza desarrollado recientemente por el Prof. Xin-She Yang (Universidad de Cambridge, Reino Unido) para resolver problemas nolineales continuos en espacios de alta dimensión, incluso con variantes para problemas multiobjetivo (Yang, 2011). Analizaremos tanto el caso de curvas como el de superficies. Pretendemos también abordar el caso de entidades con pesos, al menos en el caso de curvas, a fin de evaluar las posibles diferencias entre los casos polinomial y el racional. Por último, planteamos un estudio tanto teórico como computacional de la eficiencia de los algoritmos resultantes con respecto a otras metodologías recientes en el campo y que han mostrado un buen comportamiento para el problema abordado en este proyecto.

1.4 Estructura de la Tesis

Esta tesis está organizada en seis grandes partes, cada una de ella conteniendo uno o varios capítulos (ver la Figura 2.1 para una idea general de la estructura de la tesis). Describimos a continuación someramente cada una de las partes de la tesis.

- *Parte I - Resumen General*: integrada por un único capítulo (Capítulo 1). Es la parte correspondiente a este capítulo. Contiene el resumen general de la tesis en español, de acuerdo con la normativa.

- *Parte II - Introducción*: integrada por 3 capítulos (Capítulos 2 al 4). Ofrece una visión general de la tesis y de los principales conceptos e ideas integradas en la misma. El Capítulo 2 describe el entorno general de la tesis, su motivación y objetivos, así como una breve descripción de la estructura de la misma. El Capítulo 3 proporciona al lector una introducción suave al tema de la reconstrucción de curvas y superficies, sus principales aplicaciones, y las principales ventajas y limitaciones de estas tecnologías. El Capítulo 4 describe los conceptos generales sobre las metaheurísticas. También describe algunos ejemplos de las principales técnicas en el campo. La parte final del capítulo describe el algoritmo bat, que será la técnica utilizada en esta tesis para la resolución de los problemas de optimización ligados a la rconstrucción de curvas y superficies a partir de nubes de puntos dato ruidosos.

- *Parte III - Contribuciones*: integrada por cinco capítulos (Capítulos 5 al 9). Esta es la parte central de esta tesis. Contiene las aportaciones más importantes del trabajo de investigación vinculado a esta tesis. Los

Capítulos del 5 al 8 abordan el problema de la reconstrucción de curvas tanto polinomiales (Capítulo 5) como racionales (Capítulo 6), y superficies, nuevamente tanto polinomiales (Capítulo 7) como racionales (Capítulo 8). Finalmente, el Capítulo 9 aborda la hibridación del algoritmo de optimización global bat con técnicas de búsqueda local para mejorar la eficiencia del algoritmo de optimización.

- *Parte IV - Conclusiones y Trabajo Futuro*: integrada por dos capítulos (Capítulos 10 y 11). El Capítulo 10 describe las principales conclusiones del trabajo realizado, destacando brevemente las aportaciones de la tesis, así como las publicaciones a las que dicho trabajo ha dado lugar. El Capítulo 11, por su parte, señala algunas ideas para trabajo futuro en el tema.

- *Parte V - Apéndices*: corresponde a los apéndices de la tesis dedicados a la notación empleada, acrónimos utilizados, y algunas definiciones básicas.

- *Parte VI - Bibliografía*: contiene toda la bibliografía general utilizada en la tesis.

1.5 Principales Resultados, Conclusiones y Publicaciones de la Tesis

En esta sección resumimos brevemente los principales resultados de esta tesis, así como sus conclusiones más importantes. Finalmente, indicamos también las publicaciones a las que esta tesis ha dado lugar.

1.5.1 Principales resultados de la tesis

Los resultados principales de esta tesis son:

- *La principal contribución de la tesis es la aplicación de una potente metaheurística llamada algoritmo bat para resolver el problema de la reconstrucción de curvas y superficies de forma libre con funciones de soporte global a partir de nubes de puntos dato ruidosos.* La metodología ha sido aplicada a resolver este problema en cuatro casos relevantes, correspondientes a las curvas de Bézier y a las superficies de Bézier, tanto polinomiales como racionales, dando lugar a cuatro métodos distintos, analizados en los capítulos 5 a 8 de esta tesis. Esta metodología ha sido además testeada sobre diferentes bancos de prueba asociados a los diferentes casos de curvas y superficies analizados.

- Em este sentido, una contribución relevante ha sido la extensión de la metodología del caso polinomial al racional. La mayoría de los métodos descritos en la literatura consideran sólo el caso polinomial, por ser el más habitual en la industria. Sin embargo, ello deja fuera muchas formas, como las cónicas y las cuádricas, que no pueden ser representadas fielmente mediante funciones polinomiales. Nuestros métodos no están afectados por esta limitación.

- Nuestros métodos pueden ser aplicados a nubes de puntos dato afectados por ruido, muestreado irregular y otros problemas típicos de aplicaciones del mundo real. Por ello, pueden ser aplicados directamente a problemas que surgen habitualmente en los entornos industriales y de producción.

- Esta tesis constituye el primer caso de aplicación del algoritmo bat a problemas del mundo de los gráficos por computador y el modelado geométrico.

- Nuestros métodos no son sólo aplicables a resolver el problema de la reconstrucción de curvas y superficies. Por el contrario, al ser métodos basados en el concepto de optimización, pueden ser aplicados también a otros problemas, como el problema de la parametrización resuelto en esta tesis.

- Otra contribución relevante es la hibridación del algoritmo bat con otras técnicas de búsqueda local para mejorar la eficiencia de la optimización. En el capítulo 9 de este tesis proponemos 4 métodos meméticos distintos, basados en dicha hibridación con los métodos de Luus-Jaakola y ASSRS en sus variantes adaptativa y auto-adaptativa.

1.5.2 Conclusiones más importantes de la tesis

Las conclusiones más importantes de esta tesis son:

- *La principal conclusión de la tesis es que resulta posible aplicar la metodología propuesta en la tesis, basada en el algoritmo bat para optimizacion, a fin de resolver el problema de la reconstrucción de curvas y superficies de forma libre con funciones de soporte global a partir de nubes de puntos dato ruidosos.* En general, los métodos desarrollados han funcionado muy bien para todos los ejemplos considerados en los bancos de pruebas analizados, y para todos los casos abordados, tanto para curvas como para superficies, y tanto en el caso polinomial como en el racional.

- La segunda conclusión relevante es que es posible la extensión de los métodos para el caso polinomial al caso racional. No obstante, éste segundo caso es mucho más dificil, por la fuerte dependencia entre los distintos conjuntos de variables a calcular (parametros de los datos, puntos de control, pesos) en una forma altamente compleja y no lineal.

- Por otro lado, como se ha mencionado anteriormente, los métodos pueden ser aplicados en contextos de fuerte ruido y muestreado irregular. Por ello, pueden ser aplicados directamente en el entorno industrial y a otros problemas que surgen en aplicaciones del mundo real.

- Finalmente, resaltamos que nuestra metodología es muy general; no exige ni presupone ninguna condición sobre la función de ajuste (tal como continuidad, diferenciabilidad, o similar). De hecho, el método no asume ningún conocimiento ni información a priori sobre el problema a resolver más allá de los puntos dato.

1.5.3 Publicaciones de la tesis

El trabajo de investigación realizado en este tesis ha dado lugar a un conjunto de publicaciones internacionales. Ello nos ha permitido, por un lado, determinar el grado de originalidad de nuestras propuestas así como la calidad de las mismas, y por otro lado, obtener un valioso feedback de los expertos en el campo que nos ha posibilitado mejorar nuestros métodos en varios aspectos a partir de sus aportaciones.

La tesis ha dado lugar a 5 artículos, de los cuales 3 ya han sido publicados y los otros 2 están aún en proceso de revisión. Aunque éstos últimos no se pueden considerar aún publicaciones confirmadas de la tesis, han sido incluidos aquí porque corresponden a dos artículos invitados a dos especial issues de revistas asociadas a congresos, y cuya selección ha venido determinada por la calidad de la contribuciones originales a los congresos. Pensamos que ello supone, ya de por sí, un claro indicio de la calidad de las propuestas. Por otro lado, en ambos casos, los artículos enviados a las revistas han pasado ya una primera ronda de revisión con resultado satisfactorio, y se encuentran ahora en la última fase de revisión de las modificaciones introducidas después de esta primera ronda.

Dado que las publicaciones corresponden una a una a los métodos propuestos en los distintos capítulos de la tesis, las presentamos en el entorno de cada capítulo al que están asociadas.

Capítulo 5:

Iglesias, A., Gálvez, A., Collantes, M.: *"Bat Algorithm for Curve Parameterization in Data Fitting with Polynomial Bézier Curves". Proc. of Int. Conference on Cyberworlds, CW 2015*, Visby (Sweden). IEEE Computer Society Press, Los Alamitos CA (2015) 107-114.

Este artículo fue presentado en el congreso Cyberworlds 2015, celebrado en Visby (Suecia), en Octubre de 2015. El congreso está patrocinado por *ACM Siggraph*, *Eurographics*, e *IFIP - Technical Group on Computer Graphics*. Los proceedings del congreso han sido publicados por *IEEE Computer Society Press*. *Cyberworlds'2015* es un congreso de categoria **ERA-B conference**.

Capítulo 6:

Iglesias, A., Gálvez, A., Collantes, M.: *"Global-Support Rational Curve Method for Data Approximation with Bat Algorithm". Proc. of Int. Conference Artificial Intelligence and Applications, AIAI'2015*, Bayonne (France). IFIP Advances in Information and Communication Technology, **458** (2015) 191-205.

La conferencia AIAI es una de las más prestigiosas en el mundo de la inteligencia artificial y está patrocinada por *IFIP - Technical Group on Artificial Intelligence* La edición del 2015 tuvo lugar en Bayona (Francia), en Setiembre de 2015. Los proceedings del congreso han sido publicados por *Springer-Verlag*, en su colección *IFIP Advances in Information and Communication Technology*.

Capítulo 7:

Iglesias, A., Gálvez, A., Collantes, M.: *"A Bat Algorithm for Polynomial Bézier Surface Parameterization from Clouds of Irregularly Sampled Data Points". Proc. of Int. Conference Natural Computation 2015, ICNC'2015*, Bayonne (France). Zhangjiajie (China). IEEE Computer Society Press, Los Alamitos CA (2015) 1034-1039.

Esta conferencia, patrocinada por las sociedades *IEEE Computer Society* y *IEEE Computational Intelligence*, es un clásico en el campo de la inteligen-

cia artificial. La edición de 2015 tuvo lugar en Zhangjiajie (China), en Agosto de 2015. Los proceedings del congreso han sido publicados por *IEEE Computer Society Press*.

Capítulo 8:

Iglesias, A., Gálvez, A., Collantes, M.: *"Iterative Sequential Bat Algorithm for Free-Form Rational Polynomial Bézier Surface Reconstruction"*. *Int. Journal of Parallel Programming*, (special issue of ICNC'2015 conference). (*Submitted on Nov. 30th 2015, currently under review*).

Este artículo corresponde a la invitación del congreso ICNC'2015 para el envío de un artículo a un special issue de una revista JCR, previa selección post-conferencia de sus mejores trabajos. En nuestro caso, en lugar de enviar una versión ligeramente modificada del articulo del congreso, hemos optado por extender el método sustancialmente al caso racional para ampliar más su calidad y originalidad. El artículo ha sido ya enviado a fines del 2015. La notificación final se espera para Junio de 2016.

Capítulo 9:

Iglesias, A., Gálvez, A., Collantes, M.: *"Four Adaptive Memetic Bat Algorithm Schemes for Bézier Curve Parameterization"*. *Transactions on Computational Science*, (special issue of CW'2015 conference). (*Submitted on Dec. 31st. 2015, currently under review*).

Este artículo corresponde a la invitación del congreso CW'2015 para el envío de un artículo a un special issue de una revista, previa selección post-conferencia de sus mejores trabajos. En nuestro caso, en lugar de enviar una versión ligeramente modificada del articulo del congreso, hemos optado nuevamente por extender el método sustancialmente considerando la hibridación del algoritmo bat con métodos de búsqueda local, para ampliar más su calidad y originalidad. El artículo ha sido ya enviado a fines del 2015. La notificación final se espera para Julio de 2016.

1.6 Trabajos Futuros en el Campo

En esta sección resumimos brevemente algunas ideas para trabajo futuro en este campo de investigación. Para una descripción más detallada, remitimos

al lector al Capítulo 10 de esta tesis.

El trabajo de investigación de esta tesis doctoral puede ser ampliado y extendido de varias formas distintas. Algunos aspectos importantes que permitirían su continuación son:

- un tema abierto todavía es la creación de una batería de ejemplos que constituyan un "banco de pruebas" estandarizado sobre el cual probar futuros desarrollos de otros métodos de resolución del problema de reconstrucción de curvas y superficies. A día de hoy no existe nada ni remotamente parecido. Este banco de pruebas facilitaría enormemente la tarea de la comparación de la eficiencia de distintos métodos y permitiría establecer más fácilmente las ventajas y limitaciones de los distintos métodos desarrollados.

- la extensión de la presente metodología a otras familias de funciones de ajuste con interés en la industria, como los B-splines y los NURBS.

- la aplicación de la metodología desarrollada en esta tesis a problemas reales de la industria en diversos entornos productivos

- el desarrollo de estudios teóricos que permitan establecer las condiciones de convergencia del método. Hasta la fecha no existe ningún estudio teórico sobre el algoritmo bat. Todos los resultados referentes a la convergencia del método son puramente empíricos. Resultaría muy útil obtener desarrollos teóricos que fundamenten el método y lo doten de un mayor formalismo matemático.

- la determinación de los mejores valores de los parámetros del método de forma automática o semi-automática, en lugar de la forma manual y empírica utilizada hasta la fecha, que supone mucho tiempo y un enorme esfuerzo.

1.7 Bibliografía del Resumen

ARTICULOS EN REVISTAS:

- Alhanaty, M., Bercovier, M.: Curve and surface fitting and design by optimal control methods. Computer-Aided Design 33 (2001) 167-182.

- Barhak, J., Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Trans. on Visualization and Computer Graphics, 7(1) 1-16 (2001)

- Barnhill, R.E.: Geometric Processing for Design and Manufacturing. SIAM, Philadelphia (1992)

- Gu, P., Yan, X.: Neural network approach to the reconstruction of free-form surfaces for reverse engineering. Computer-Aided Design, 27(1) 59–64 (1995)
- Hoffmann M.: Numerical control of Kohonen neural network for scattered data approximation. Numerical Algorithms, 39, 175–186 (2005)
- Levoy, M. Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The Digital Michelangelo Project: 3D scanning of large statues. In SIGGRAPH 2000, New Orleans, (2000) 131-144
- Meyers, D., Skinnwer, S., Sloan, K.: Surfaces from contours. ACM Transactions on Graphics, 11(3) (1992) 228-258
- Pottmann, H., Leopoldseder, S. Hofer, M., Steiner, T., Wang, W.: Industrial geometry: recent advances and applications in CAD. Computer-Aided Design, 37, 751–766 (2005)
- Pratt, V.: Direct least-squares fitting of algebraic surfaces. Proc. of SIGGRAPH87, Computer Graphics, 21(4) (1987) 145-152.
- Schmitt, F., Barsky, B.A., Du,W.: An adaptive subdivision method for surface fitting from sampled data. Proc. of SIGGRAPH86, Computer Graphics, 20(4) (1986) 179-188.
- Sclaroff, S., Pentland, A.: Generalized implicit functions for computer graphics. Proc. of SIGGRAPH91, Computer Graphics, 25(4) (1991) 247-250
- Yang, X. S..: Bat algorithm for multiobjective optimization. Int. J. Bio-Inspired Computation, 3(5), 267-274 (2011).
- Yang, X. S., Gandomi, A. H.: Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 29(5), 464–483 (2012).

LIBROS:

- Dierckx, P.: Curve and Surface Fitting with Splines. Oxford University Press, Oxford (1993)
- Engelbretch, A.P.: Fundamentals of Computational Swarm Intelligence. John Wiley and Sons, Chichester, England (2005)
- Farin, G.: Curves and Surfaces for CAGD (5th ed.). Morgan Kaufmann, San Francisco (2002)

- Hoschek, J., Lasser, D.: Fundamentals of Computer Aided Geometric Design. A.K. Peters, Wellesley, MA (1993).
- Patrikalakis, N.M., Maekawa, T.: Shape Interrogation for Computer Aided Design and Manufacturing. Springer Verlag, Heidelberg (2002)
- Rice, J.R.: The Approximation of Functions. Vol. 2. Addison-Wesley, Reading, MA (1969).

CAPITULOS DE LIBROS:

- Varady, T., Martin, R.: Reverse Engineering. In: Farin, G., Hoschek, J., Kim, M. (eds.): Handbook of Computer Aided Geometric Design. Elsevier Science (2002).
- Yang, X. S.: A new metaheuristic bat-inspired algorithm. In: J. R. Gonzalez et al. (Eds.) Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).

Part II

INTRODUCTION

Chapter 2

Introduction

This chapter introduces the general ideas about the work carried out during this doctoral thesis. The discussion begins with the presentation of the general framework of the thesis. Then, the motivation to carry out this research work is discussed in Section 2.2. The section also outlines the main goals to be achieved in the research work of this thesis. Finally, Section 2.3 describes the general structure of this thesis, with a brief explanation about the contents of each chapter of this document.

2.1 General Framework of the Thesis

This book corresponds to the PhD thesis of Marta Collantes, carried out at the Department of Applied Mathematics and Computational Sciences of the University of Cantabria, Santander, Spain and co-supervised by Dr. Andrés Iglesias Prieto and Dr. Akemi Gálvez Tomida.

The thesis is based on the application of a powerful nature-inspired metaheuristic approach for global optimization called the bat algorithm. This algorithm has been recently introduced by Prof. Xin-She Yang to solve difficult continuous optimization problems that cannot properly addressed by traditional mathematical optimization techniques (see Chapter 4 for further details).

In this thesis, this method is applied to solve the problems of curve reconstruction and surface reconstruction from given sets of data points by using global-support basis functions. The reader is kindly referred to Chapter 3 for a detailed explanation about the subject of curve and surface reconstruction as well as the issue of global-support basis functions.

The research work in this thesis has been carried out in the framework of two research projects from the Spanish National Research and Development Plan, Computer Science National Program. The first of these projects was:

> *Artificial Intelligence for Geometric Modeling and Computer Graphics.* (Ref. TIN2006-13615). **Spanish Ministry of Education and Science**, Computer Science National Program (September 2006-September 2009). Principal Investigator: Andrés Iglesias.

This project was focused on the applications of different artificial intelligence techniques to several problems in computer graphics and geometric modeling, including the issue of curve and surface reconstruction. The proposal in that project was based on recent works suggesting the use of artificial intelligence (AI) techniques such as neural networks (often combined with numerical methods based on either ordinary or partial differential equations) provide better solutions to problems such as surface reconstruction (the determination of a surface from a set of unorganized points such that it satisfies a prescribed set of functional constraints). This problem arises in reverse engineering for computer-aided manufacturing via three-dimensional laser-scanning but also in many other fields, such as medicine (generation of cross-section surfaces in computer tomography, magnetic resonance imaging), virtual and augmented reality, architecture, archaeology, and many others (see Chapter 3 for a detailed presentation of different applications in several fields). Until that project, we obtained very promising results by using functional networks, a generalization of neural networks in which the weights are replaced by multivalued functions, thus allowing more flexibility. In that project, further research was proposed in order to clearly determine the advantages and drawbacks of this approach. We also pursue to obtain more powerful and more efficient methods to solve these problems.

The research work in this initial project was later continued in a second project:

> *Metaheuristics for Automatic Free-Form Curve and Surface Reconstruction in Reverse Engineering.* (Ref. TIN2012-30768). **Spanish Ministry of Economy and Competitiveness,** Computer Science National Program (January 2013-December 2016). Principal Investigator: Andrés Iglesias.

This second project was particularly focused on the application of metaheuristic techniques to the problem of curve and surface reconstruction by using free-form geometric entities. The motivation of that project is that it has been shown that the application of neural networks (and even their extension, functional networks) is not able by itself (i.e., working alone, without the hybridization with other approaches) to solve the problem in its generality. On the other hand, recent results reported in the literature have shown that some recent optimization techniques such as the metaheuristics can solve very difficult continuous optimization problems, such those addressed in this thesis. In addition, we pursue to solve these problems in an autonomous and fully automatic way.

This thesis contains part of the research results carried out during the completion of these two national research projects. All the results achieved during this PhD thesis and reported in this document have been obtained from the research conducted in such projects.

2.2 Motivation and Main Goals of the Thesis

The automatic reconstruction of free-form curves and surfaces (the most common ones in industrial settings and design) from clouds of data points is one of the most important problems in geometric modeling and processing and in computer-aided geometric design (CAGD). It also has a very high relevance in a number of industrial processes such as CNC (computer-numerically controlled) milling and machining for manufacturing, and in CAD/CAM for fields such as automotive, aerospace, and shipbuilding industries. Also in shoe industry, medical imagery, archaeological assets reconstruction, and many others. The wide range of applications of this technology (described in detail in Section 3.2 of this dissertation) is the main motivation for the research work carried out in this thesis.

The problem to be solved can be explained as follows:

> *Given a set of noisy data points assumed to lie on an unknown curve/surface, the goal is to construct, to the best possible extent, a compact representation of such a curve/surface that fits those data points better.*

This problem appears ubiquitously in *reverse engineering*, a field that aims at transforming real parts (acquired through optical or tactile 3D scanning or other digitizing or measurement devices) into engineering models and concepts for further use and manipulation by computer. This is a key technology in manufacture, since the digital models can be stored, modified and

transmitted more efficiently than their real counterparts. Moverover, digital models can be used even although the real objects are lost or become unavailable or unusable. Furthermore, digital models are also often easier and cheaper to modify and analyze than the physical objects themselves. By using a mathematical curve/surface model, the same complex shape can be economically represented by as few as tens of parameters.

In this context, the primary goal of this thesis is *to investigate the application of a powerful metaheuristic technique to free-form curve/surface reconstruction from clouds of (possibly unorganized and noisy) data points*. In addition to this major goal, we seek *to develop methods to obtain a mathematical representation of such a curve/surface from minimal information*, namely, a set of data points lying on it, decreasing substantially the approximation errors of current methods. The third goal is *to achieve the previous goals in a fully automatic way* (i.e., without human intervention) while simultaneously *being able to reconstruct with high accuracy very complex shapes unfeasible with current methods*. Such *methods should also be robust, numerically stable, and computationally fast and efficient*. The methods developed for this thesis will be described in detail in Chapters 5 to 9 of this written dissertation.

2.3 Structure of the Thesis

Figure 2.1 shows the general structure of this thesis. This thesis is structured into six parts, each one comprised at its turn of one or several chapters. In the figure, the different parts are enclosed in the rectangles with colored background, and identified with capital letters indicating the corresponding part, from I to VI, along with the title of each part. At their turn, the different chapters are indicated by white rectangles with their chapter number and title within. The arrows in the figure indicate the flow of the chapters for easier identification.

The first part, labelled as *Part I: Resumen General* and fully written in Spanish to comply with the regulations of the University of Cantabria regarding the written PhD dissertation, contains only one chapter. As required by such regulations, it provides a general summary of the thesis as well as its main contributions, research results and conclusions. For total completeness of this part, the bibliography of this particular chapter is included at the end of hte chapter. This differs from the rest of the thesis, where the bibliography for the rest of chapters is collected altogether in final chapter for the sake of clarity.

The second part, labelled as *Part II: Introduction*, is devoted the general concepts and offers a gentle yet comprehensive introduction to the main fields involved in this research work. As such, this part must be understood

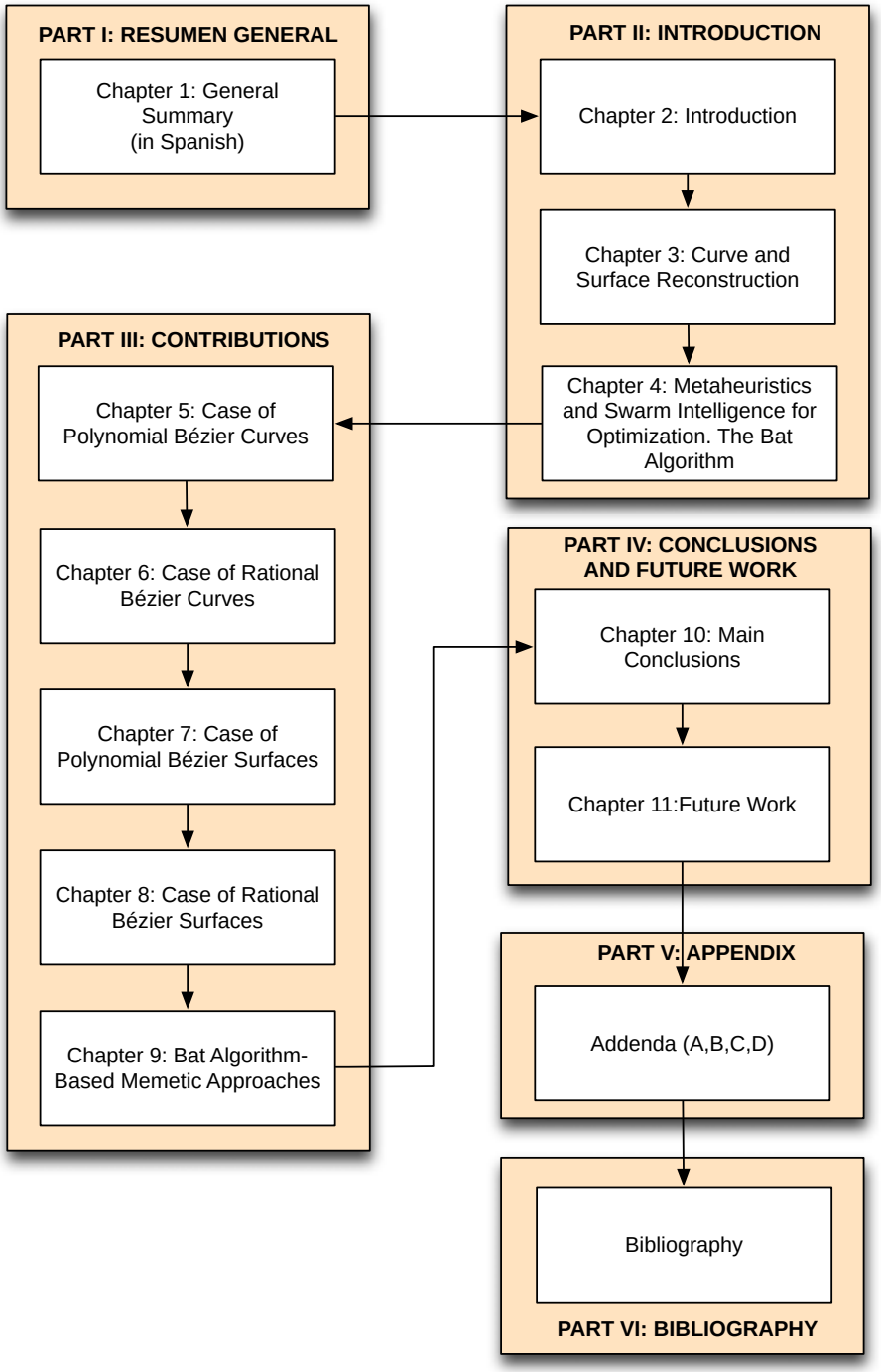

Figure 2.1: General structure of this thesis with indication of the different parts and chapters.

as the preliminaries part providing the required background to be able to follow the other parts of the thesis properly. This part is comprised of three chapters. Chapter 2 (this one) describes the general framework of this thesis along with its motivation and main goals. It also explains briefly the general structure of this thesis (this section). Chapter 3 provides a comprehensive introduction to the problem of curve and surface reconstruction, its main applications, advantages and limitations. It also provides a general review about the state of the art in the field. Finally, Chapter 4 describes the general concepts about metaheuristics and swarm intelligence. It also describes some of the most popular and most relevant metaheuristic approaches. We start with simulated annealing as a classical example of a single-particle approach. Then, our description follows with some classical and well-known approaches in swarm intelligence, such as particle swarm optimization and genetic algorithms. Some recently introduced swarm intelligence approaches, such as firefly algorithm and cuckoo search are also reported. Finally, we also describe the bat algorithm, the bio-inspired metaheuristic technique used in this thesis.

The third part of this thesis, labelled as *Part III: Contributions*, provides the real core of this research work, as it is devoted to explain the main contributions of this thesis. The part is comprised of five chapters, corresponding one-by-one to the five bat algorithm-based methods developed in this thesis. They are arranged, for clarity, from the most simple to the most complicated case, addressing firstly the case of curves and then the case of surfaces. Thus, Chapter 5 describes our method for the case of polynomial Bézier curves, while Chapter 6 extends our technique to the case of rational Bézier curves. Chapters 7 and 8 are devoted to the cases of polynomial Bézier surfaces and its extension to the case of rational Bézier surfaces, respectively. Finally, Chapter 9 considers the hybridization of the technique described in Chapter 6 with four local search methods, the Luus-Jaakola and the adaptive step size random search for the cases of both adaptive and self-adaptive versions. All chapters from 5 to 9 follow a similar structure: firstly, the problem to be solved is introduced, then the technique applied to solve it is explained; finally, the main results obtained from our computational experiments are fully reported.

The fourth part of this thesis, labelled as *Part IV: Conclusions and Future Work*, is comprised of two chapters. The former one, Chapter 10, reports the main conclusions derived from the research work of this thesis, while the latter outlines some interesting research lines for future work in the field.

The fourth part of this thesis, labelled as *Part V: Appendix*, contains four addenda to this thesis, labelled as Addendum A to D, and devoted to the conventions and names used in this thesis, the general mathematical notation, and some concepts and properties about the dot product and the

cross product, respectively.

Finally, the last part, labelled as *Part VI: Bibliography*, contains a collection of more than 100 bibliographic entries corresponding to the references to previous works used during the completion of this thesis.

Chapter 3

Curve and Surface Reconstruction

3.1 Motivation

One of the most visible effects of globalization is the growing global competition among manufacturers and industrial corporations to deliver more competitive products with better quality and lower prices. Due to this globalization process, manufacturers are constantly challenged to optimize processes in order to provide the best and most efficient cost/quality ratio. Under these new rules of competition in global market, design is taking a central role in the product development lifecycle. The prior emphasis on designing whole product lines for customers has experienced a radical shift as companies increasingly enable customers to mass customize their own products. Under this new paradigm, products are manufactured in lesser amounts but with greater product diversity. As a result, the geometric model of the product has to be changed frequently during the design process, making CAD/CAM systems inevitable tools for the design and manufacturing processes.

As part of the initial conceptual design process, it is common in many industries (automotive, aerospace, ship building) to build prototypes in a real workshop with materials such as foam, clay, wood, metal or plastics, in order to explore ideas for shape, size, proportions and the human sense of interacting with the model. The model thus obtained is digitally stored through data sampling by means of technologies such as 3D laser scanning. Data points are then fitted to mathematical curves and surfaces, a process called *data fitting*.

Reverse engineering is a crucial technology in the current manufacturing pipeline. In its most comprehensive meaning, reverse engineering consists of obtaining a digital replica of an already existing physical object or component. While conventional engineering transforms engineering concepts and

Figure 3.1: Example of a handheld laser scanner.

models into real parts, in reverse engineering real parts are transformed backward into engineering models and concepts. This is a typical procedure in medical and health areas, where non-invasive techniques such as magnetic resonance imaging (MRI) or computer tomography (CT) are commonly used to visualize inner organs or different parts of the human body for medical check, diagnosis, and therapy. Reverse engineering is also a common practice in consumer products, microchips, and other electronic components for different purposes. For instance, to analyze how a new device or machine in the market is built or how a particular component works. Also to determine whether or not a new product infringes a patent or intellectual property rights.

Another relevant application arises in automotive and aerospace industries, where prototypes are built to help designers and engineers explore new ideas and get a visual insight of a new part to be designed. This is a challenging task, since nowadays designs are becoming more and more organic in shape, making them more difficult to be replicated by computer from scratch. A typical approach in this regard is to obtain a set of measurements of the object or workpiece and then reconstruct it as a 3D model. Typical ways to measure include scanning technologies such as 3D laser scanners, touch scanners, coordinate measuring machines, light digitizers, and industrial computer tomography. Typically, the process yields a cloud of data points, which have to be fitted in order to recover the topological information of the original model.

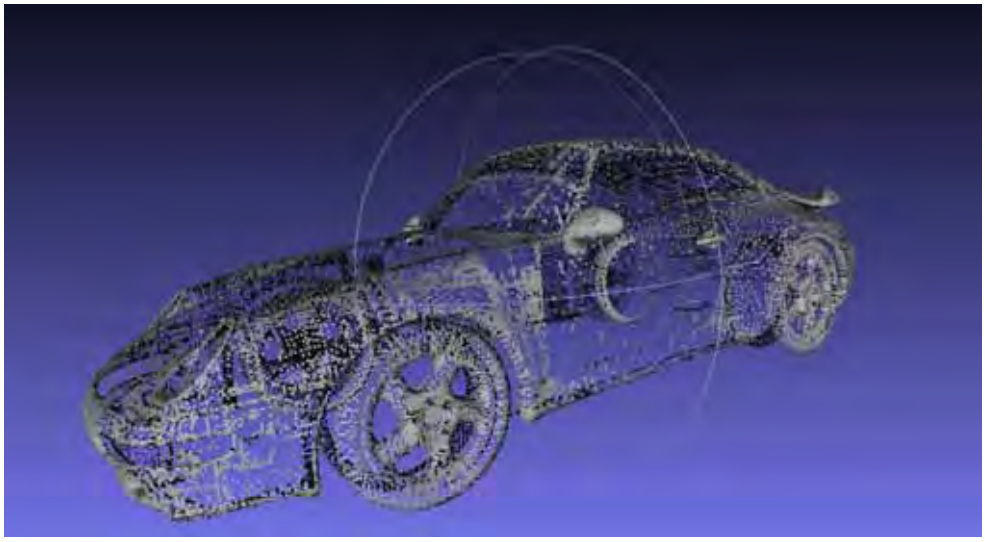

Figure 3.2: Example of a point cloud corresponding to a car model by Porsche.

Advantages of this process are obvious: digital models are easier and cheaper to modify and analyze than the physical counterparts. In addition, digital data are very well suited for efficient storage and transmission among providers and manufacturers and can still be used even when the real objects are lost or become unavailable. In addition, digital data are very well suited for efficient storage and transmission among providers and manufacturers and can still be used even when the real objects are lost or become unavailable.

Currently, laser scanner is the most common method used in reverse engineering applications because it is fast and robust relative to other methods. Laser scanner systems yield a (possibly massive) cloud of 3D data points from which a 3D model is to be reconstructed. This large collection of measured points is usually represented as a *point cloud* typically comprised of hundreds of thousands, and even millions of data points. Figure 3.2 shows an example of a light point cloud comprised of 75,422 points, obtained from a model of a car by German carmaker Porsche.

Similarly, Figure 3.3 shows a cloud point obtained from a blade of a workpiece (a propeller in this case). This point cloud is comprised by 1,831,155 3D points.

Typically, the point clouds lack topological and geometrical information beyond the data points, and must therefore be further processed. The classical approach is to create a model from data by using either a polygonal (usually triangular) mesh, a constructive solid model, or a set of mathematical curves and surfaces. For instance, Figure 3.4 shows the reconstruction of the point cloud in previous Figure 3.3 by using a polygonal mesh, comprised

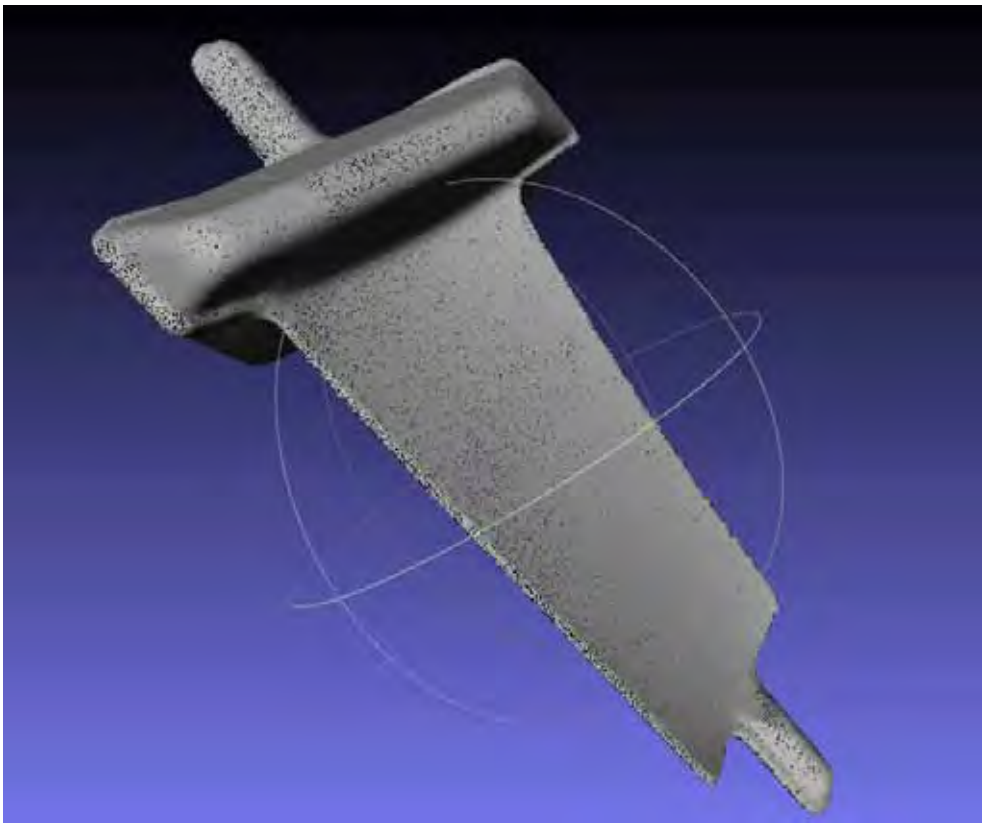

Figure 3.3: Example of a point cloud corresponding to a blade. The model consists of more than 1 million data points.

of more than 1 million of vertices (as many as the number of data points in the cloud) and more than 10 million of faces. Note that in this case, the complexity of the model is not reduced but drastically increased instead. This is a general behavior for polygonal meshes. Because of this reason, in this thesis we will focus on the latter case, usually referred to as *curve and surface reconstruction*.

Curve/surface reconstruction has many remarkable advantages. By using a mathematical surface model, the same complex shape can be economically represented by as few as 50-500 parameters. This issue becomes increasingly important with the rapid development of powerful communication technologies, making it possible to send those models all over the world in just a matter of seconds. Furthermore, parametric mathematical models are the preferred representations in engineering design applications because they can be readily modified by changing only a small set of parameters, such as the control points.

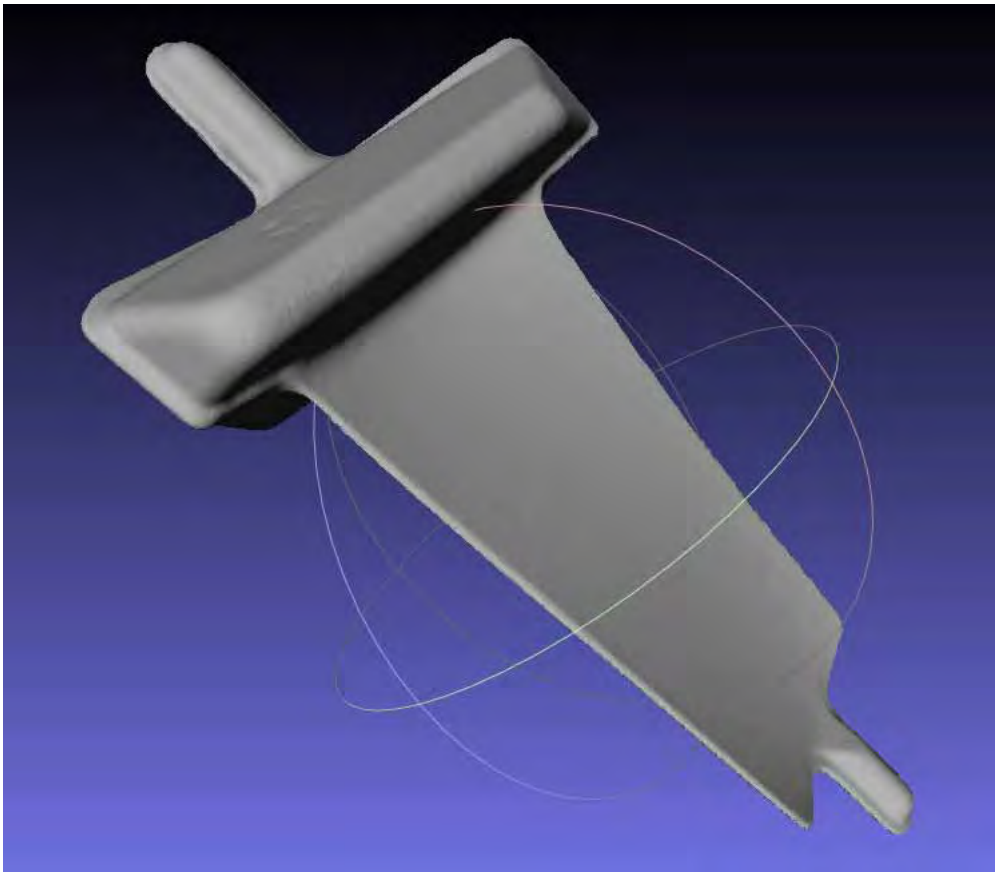

Figure 3.4: Reconstruction of the point cloud in Figure 3.3 by using a polygonal mesh. In this case, the complexity of the model (i.e., the number of data) does not decrease, but it drastically increases instead.

3.2 Approaches for Curve/Surface Reconstruction

The problem of recovering the shape of a curve or a surface, also known as curve (respectively surface) reconstruction, is one of the most difficult and challenging "unsolved" problems in geometric modeling during the last two decades. In this thesis, when we talk about the issue of curve (surface) reconstruction we refer to the following problem:

> *Given a (usually very large) collection of (possibly noisy) data points in 2D or 3D, how to compute a parametric curve (surface) that "follows" the shape of these data points so that it faithfully recovers the underlying shape of the point cloud.*

Depending on the nature of these data points, two different approaches can be employed for this problem: *interpolation* and *approximation*. In the former, a parametric curve or surface is constrained to pass through all input data points. This approach is typically employed for sets of data points that come from smooth shapes and that are sufficiently accurate. On the contrary, approximation does not require the fitting curve or surface to pass through all input data points, but just close to them, according to some prescribed distance criteria. The approximation scheme is particularly well suited for the cases of highly irregular sampling and when data points are not exact, but subjected to measurement errors. In real-world problems the data points are usually acquired through laser scanning and other digitizing devices and are, therefore, subjected to some measurement noise, irregular sampling, and other artifacts [152, 156]. Consequently, a good fitting of data should be generally based on approximation schemes rather than on interpolation [122, 134, 151, 188].

This curve/surface reconstruction problem has received much attention during the last few decades. One of the main reasons to explain such interest is the wide range of applicability of this problem in many theoretical and applied domains (see our discussion in Section 3.3 below). In fact, this issue has been considered a very hot topic in research for almost 50 years. It is also a very elusive and challenging problem. In spite of the intensive research effort carried out since the first theoretical developments in the 50s and 60s, the problem is still far from being solved in all its generality. There are two main reasons to explain this situation:

- on one hand, the problem can be formulated in many different ways, according to the specific criteria used for classification. For instance, the problem has been addressed in completely different ways depending on the input of the problem (cross-sections, clouds of points, isoparametric curves, mixed information, etc), the geometric structures used for representation (polygonal meshes, solid primitives, parametric curves and surfaces), the mathematical approach employed (interpolation, approximation), the mathematical constraints (if any), and many other factors. In other words, there is no universal solution for this issue, as different conditions require different approaches to this problem. So instead of a *"for all"* single method, we are facing a myriad of specialized methods to deal with.

- on the other hand, most of the techniques developed so far to solve this problem have been usually based on the application of traditional mathematical optimization techniques. Although they are still very powerful and able to cope many different situations, the particular conditions of this problem (noisy data, little or no information about the problem, no

continuity or differentiability of objective function ensured) limit the applicability of such techniques in this particular field. Furthermore, it has been shown during the last two decades that such methods cannot be successfully applied to solve very difficult nonlinear continuous optimization problems. In particular, they fail to solve the curve/surface reconstruction problem with free-form parametric fitting functions such as Bézier, B-splines and NURBS, by far the most common geometric entities in industrial and applied domains.

Owing to these critical limitations, the scientific community in the field have turned their attention to artificial intelligence and other modern optimization techniques, such as metaheuristics. This is also the approach taken in this thesis. These kind of techniques will be explained in detail in Chapter 4 of this thesis.

3.3 Applications

As stated in previous section, the curve/surface reconstruction problem is a classical problem in many different fields. In this section, we summarize some of the most relevant applications of this subject in both theoretical and applied domains. The list is by no means exhaustive, but it still provides our readers with a gentle overview about a number of interesting applications.

3.3.1 Theoretical fields

Numerical and functional analysis: Curve and surface reconstruction is a key technique in approximation theory, a major topic in numerical and functional analysis [13, 47, 49, 50, 73, 86, 133, 134]. *Spline interpolation* is an issue of particular importance because it yields similar results to polynomial interpolation but does not require high degree polynomials thus avoiding the instability associated with Runge's phenomenon [168, 169]. Spline functions are also very flexible and simple enough, so they are usually preferred over canonical polynomial functions for data interpolation. Spline functions are also widely used in data approximation, as their flexibility make them an excellent tool to capture the underlying shape and structure of noisy data [6, 12, 29, 94, 96, 196].

Statistics and machine learning: Curve and surface reconstruction with splines is a major procedure in *data fitting* and *regression* [28, 30, 31, 122, 126, 129, 144], as they are more stable than other fitting functions. Besides, some splines can perform very well for these tasks even for low-degree polynomial fitting functions, particularly when piecewise functions are involved. This

ability to tune the degree of the fitting functions is essential to prevent overfitting in statistical and machine learning predictive models and to minimize the number of free parameters of the model [33, 79]. Finally, it is a key ingredient for other problems such as *smoothing*, *filtering*, *nonparametric regression*, and many others [34, 94, 95, 98].

3.3.2 Applied fields

Most applications of curve/surface reconstruction in industrial and applied settings are associated with a process commonly known as *reverse engineering*. While conventional engineering transforms engineering concepts and models into real parts, reverse engineering represents the other side of the process. In reverse engineering real parts are transformed backward into engineering models and concepts, typically stored in digital form. Some relevant examples include:

Medical imaging: two typical problems in medical imaging are to construct a curve (or collection of curves) from data points and to obtain the external surface of a 3D object from a set of cross-sections of that object. For instance, authors in [4, 5, 109, 128, 141, 149] address the problem of obtaining a surface model from a set of given cross-sections. The starting point is a dense cloud of data points of the surface of a volumetric object (an internal organ, for instance) acquired by means of noninvasive techniques such as computer tomography, magnetic resonance imaging, or ultrasound imaging. The primary goal in these cases is to obtain a sequence of cross-sections of the object in order to construct the surface passing through them, a process called *surface skinning*. In other cases, the input is directly a sequence of 2D cross-sections or thin layers (see, for instance, Figures 3.5 and 3.6 for two examples corresponding to computer tomography and magnetic resonance, respectively). Then, the reconstruction method aims to generate an accurate *volumetric representation* of the surface of the object to be displayed for diagnosis and other medical purposes. Other classical input data include iso-parametric curves on the surface [76] and even mixed information, such as scattered points and contours [3, 9, 135, 175] or iso-parametric curves and data points [95, 97].

Biomedical engineering: The use of reverse engineering techniques is very common in biomedical engineering for the design and manufacturing of prosthesis of different types. Other typical problem is the generation of customized medical implants, where the basis geometric is acquired from the real model by using some digitizing devices and/or medical imaging. These orthopedic implants are designed to replace a missing joint or bone or as a

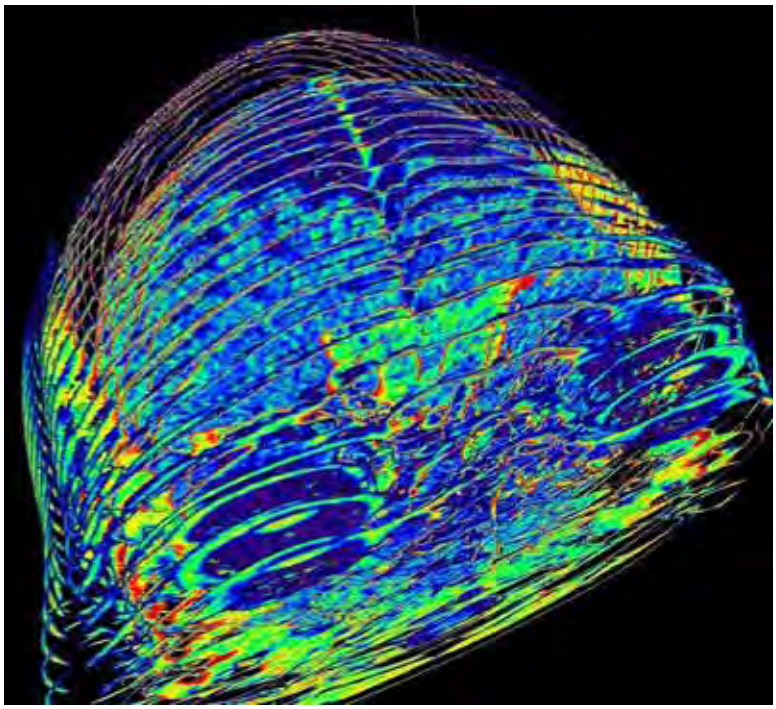

Figure 3.5: Example of a computer tomography (CT) scanned volume: 3D volumetric reconstruction of the brain and eyes comprised of several slices of 5mm stored in DICOM format (source: *Wikipedia*).

supporting element of a damaged rigid body structure. In some instances, the reconstruction techniques in this field make use not only of parametric functions but also of implicit functions [128, 141, 149, 175, 178].

Automotive, aerospace and ship hull building industries: In automotive, aerospace, and ship hull building industries it is common to build prototypes in a real workshop as part of the initial design process [187, 188]. Those prototypes, usually made with materials such as foam, clay, wood, metal or plastics, help the designers to explore conceptual ideas for shape, size, proportions, and the human sense of interacting with the model. The resulting model is digitally stored through data sampling by means of technologies such as 3D laser scanning. Data points can readily be manipulated by computer and then converted into digital models, generally described in terms of mathematical curves and surfaces [79, 80, 152, 156]. Advantages of this process are obvious: digital models are easier and cheaper to modify and analyze than the physical counterparts. In addition, digital data are very well suited for efficient storage and transmission among providers and manufacturers and can still be used even when the real objects are lost or be-

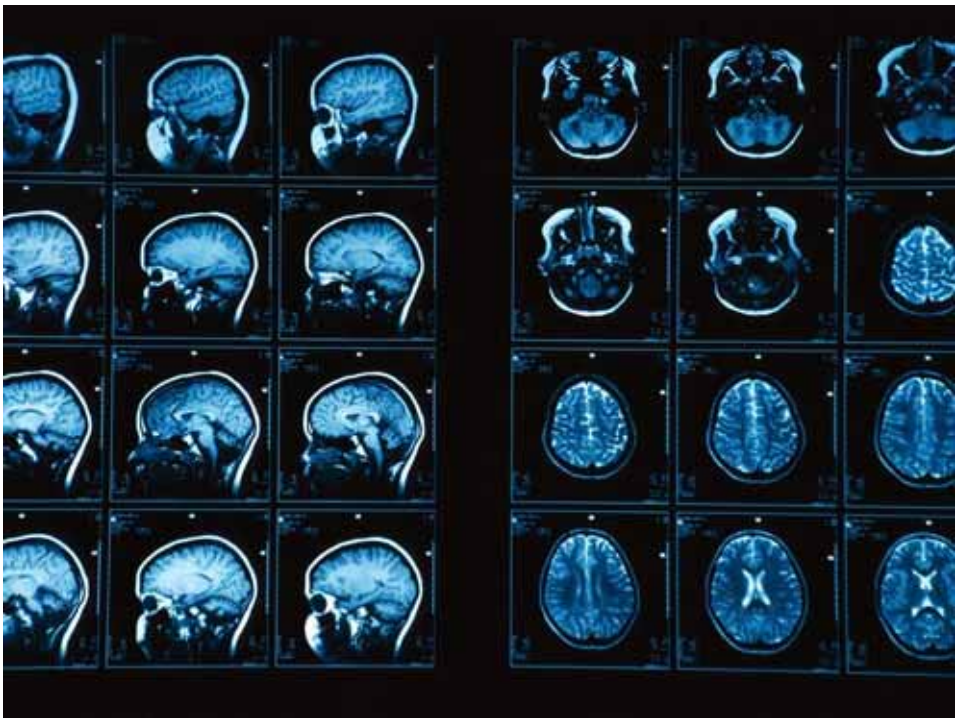

Figure 3.6: Example of magnetic resonance imaging of the human brain (source: *National Geographic*).

come unavailable. Classical examples include turbine blades, airplane wings, car body parts, boat parts, and many others (such as those shown in Figures 3.3 and 3.7).

Computer design and manufacturing (CAD/CAM): Surface reconstruction is also used to improve some CAD/CAM processes [29, 38, 58, 62, 81, 152, 187, 192] such as the determination of trajectories for *tool-path generation* in *computer-numerically-controlled* (CNC) operations such as *milling* and *machining* and other manufacturing-related processes (*filing*, *turning*, *grinding*, etc). Curve and surface reconstruction is also involved in variational analysis for *fairing* and *smoothing* [78, 82, 92, 111].

Rapid prototyping and rapid manufacturing: the application of curve and surface reconstruction techniques for *rapid prototyping* (the generation of scale models of physical parts from CAD data) is becoming increasingly popular during the last few years [7, 152, 156, 187]. A major reason of this popularity is the quick availability of very efficient methods for generating physical prototypes, particularly *3D printing* and *additive layer manufacturing* technology. In such technologies, successive layers of material are laid

down until the entire object is created. Each of these layers can be seen as a very thin horizontal cross-section of the eventual object. Printing of such layers is performed through different methods such as selective laser sintering (SLS), fused deposition modeling (FDM), and stereolithography (SLA). 3D printing is also used for *rapid manufacturing*, where 3D printers are used to produce the actual end user products rather than merely prototypes. This is an emerging field nowadays, particularly regarding the mass customization of personal goods.

Metrology and product assessment: reverse engineering is widely used for metrology purposes (measurement of physical properties of a manufactured product). Classical technologies in this field are *coordinate-measuring machines*, *laser scanners*, *structured light digitizers*, or *industrial computer tomography scanners* [156, 187, 188]. Figure 3.7 shows an example of an engine component (a throttle) scanned for product assessment and quality control purposes. The model consists of more than 30 million 3D data points.

Other uses involve the creation of digital models of manufactured goods, the assessment of competitors' products and identification of possible patent or copyright infringement. The range of applications can even go to microscopic sizes. For instance, the paper in in [118] reports an interesting application of a genetic algorithm for the reconstruction of 3D surface topography in scanning electron microscopy.

Computer animation and computer-generated movies: Curve/surface reconstruction methods are an essential ingredient of modern computer animation techniques for computer-generated movies and videogames. A typical example of application are the motion capture (*mo-cap*) systems where movements of real actors are recorded through sensors and the resulting information is used to animate digital characters by computer by using sophisticated computer animation techniques. Some remarkable examples of this kind of technology can be found in blockbusters movies such as James Cameron's *Avatar* (see Figure 3.8) where a new technology for real-time facial animation called FACS (Facial Action Coding System) for performance capture and mapping onto an arbitrary character face was developed, *Pirates of the Caribbean* (see Figure 3.9) or for the famous critically-acclaimed character of *Gollum* in the *Lord of the Rings* film trilogy (see Figure 3.10).

In this regard, curve reconstruction is used to generate the trajectories of individual parts of the character through inverse kinematics, a process that interpolates the positions of joints at specifically selected frames (keyframes) to generate the frames in between, a process usually called *keyframing*, *inbetweening*, or *tweeing*. On the other hand, surface reconstruction is applied to generate the 3D model of the digital character. Curve reconstruction tech-

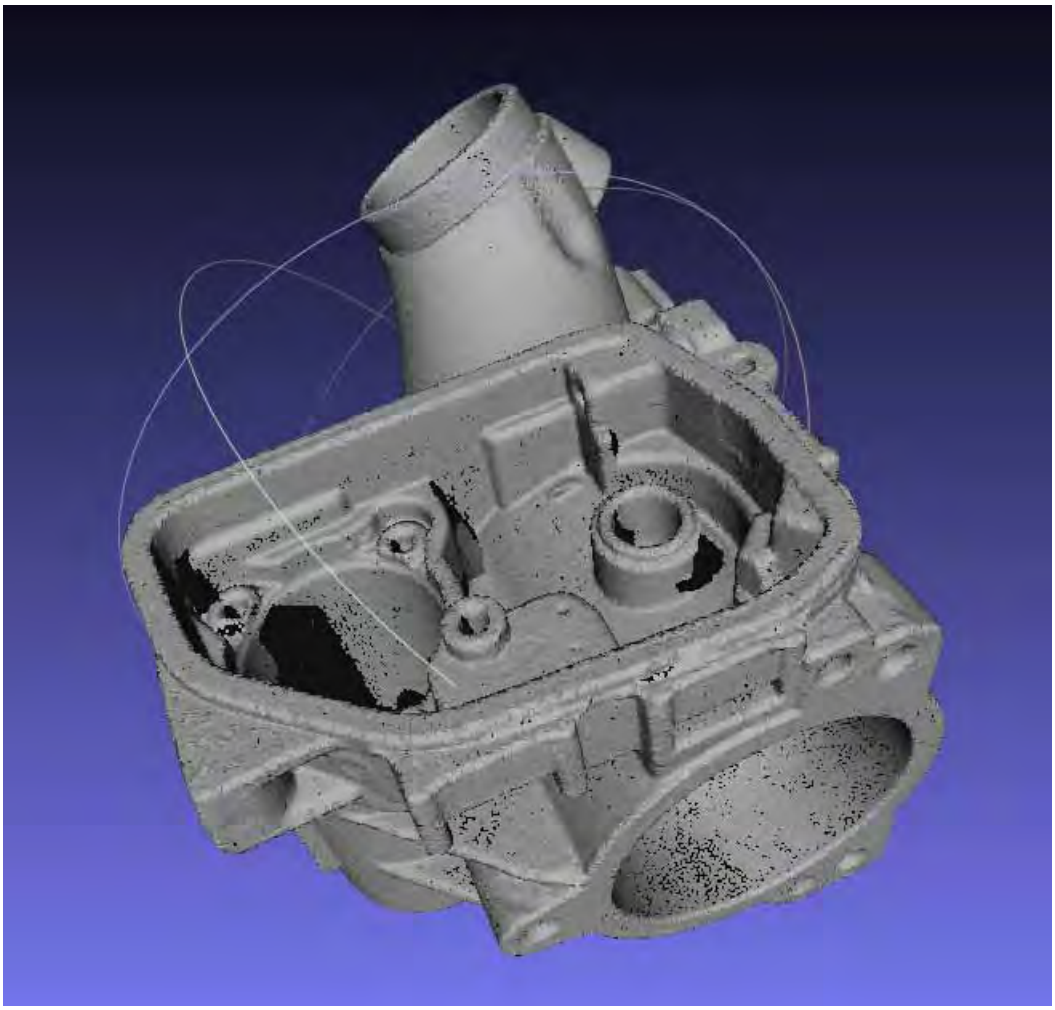

Figure 3.7: Example of an engine component (a throttle) scanned for product assessment and quality control purposes.

niques are also used to generate paths for the digital characters moving in a virtual environment for computer movies and videogames, and to determine trajectories for *camera walkthrough* rendering in computer animation and scientific visualization, a technique in which a predefined walkthrough animation of a scene is obtained by placing the camera on a path according to a prescribed trajectory or procedural sequence. Also, curve reconstruction techniques appear in animation approximation, a field that aims to compactly represent an animation sequence (such as mesh animations or skeletal animations) through non-uniform B-splines, taking advantage of the temporal coherence of the animation data [146].

Figure 3.8: Example of the use of FACS technology for the blockbuster movie *Avatar*. Facial markers are placed at features points of the actor's face and surface reconstruction is applied to recover the face of the digital character for real-time rendering (source: *Lightstorm Entertainment / Dune Entertainment / Ingenious Film Partners / 20th Century Fox).*

Geometric modeling and processing: curve/surface reconstruction methods are at the core of many processes in geometric modeling and processing [2, 24, 25, 38, 93, 111]. Approaches in these fields have typically relied on polygonal meshes, for which very powerful, advanced techniques have been developed [35, 90, 91]. More recently, schemes based on solid geometry [123, 195] and mathematical curves and surfaces have also been developed [54, 59, 69, 74, 99, 150, 151, 171, 199, 210, 211, 214].

Archeology and digital heritage: A major step in this area was given by the ambitious "Digital Michelangelo" project [124]. Carried out in 1998-99 by a team of 30 faculty, staff, and students from Stanford University and the University of Washington who spent one year in Italy scanning the sculptures and architecture of Michelangelo, it represented the starting point for a number of similar projects aimed at applying shape reconstruction for the digital representation, storage, and analysis of historical and cultural artifacts. The project received a lot of attention not only from the scientific community but also from mass media, and contributed significantly to the popularization of the application of these techniques to archeology and digital heritage.

The field has also benefited from the popularization and miniaturization

Figure 3.9: Example of motion capture technology in the fantasy swashbuckler film series *Pirates of the Caribbean*. Reference points are used to track the motion of the different joints of the characters and their replacement by computer-generated characters following the animations of real actors (source: *Walt Disney Pictures / Jerry Bruckheimer Films / Walt Disney Studios / Motion Pictures*).

Figure 3.10: Two example of facial and motion capture technology for the Gollum character in the *Lord of the Rings* film trilogy. This technology can capture even the subtle changes in facial expressions as well as body movements with high accuracy (source: *WingNut Films / The Saul Zaentz Company / New Line Cinema*).

of affordable handheld laser scanners and portable coordinate-measuring machines, which have assisted the archeologists in capturing the shape and details of engravings of historical objects and artifacts. Figure 3.12 shows the point cloud associated with an ancient vase captured with a handheld laser

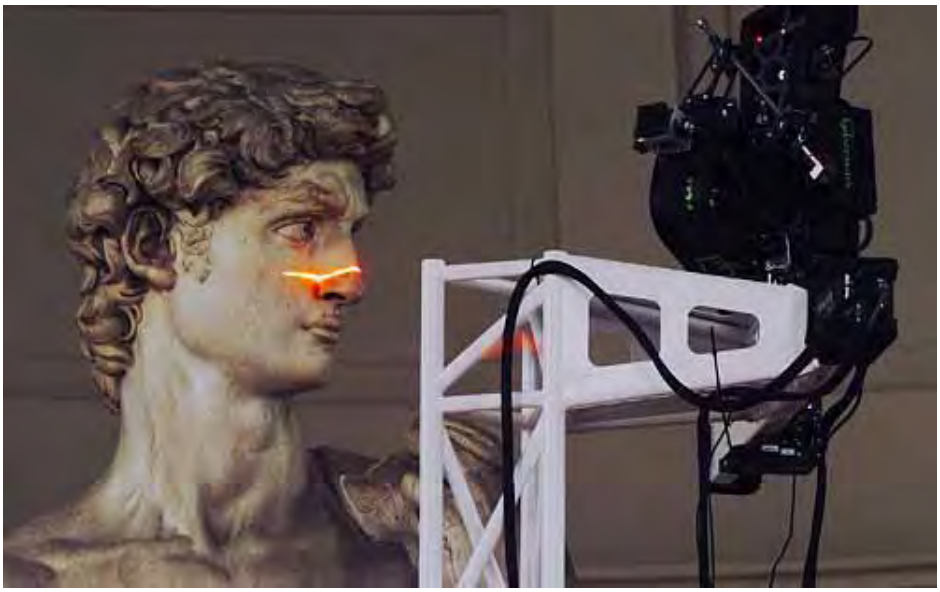

Figure 3.11: The laser scanner used in the *Digital Michelangelo* project to create a digital reconstruction of the famous statue of David (source: *Marc Levoy/Digital Michelangelo Project.*

scanner. The point cloud is pretty massive, as it contains more than 8 million data points in order to capture even the subtle details of any engraving on the surface of the vase.

In addition, the use of photogrammetry techniques have revealed to be very useful in capturing the geometry of historical buildings. Also, LIDAR technology (see Figure 3.13) is widely used to create high-resolution digital elevation models (DEMs) of archaeological sites that help archeologists to discover micro-topography hidden by vegetation and/or additional natural or artificial actions. The combination of this technology with Geographical Information Systems (GIS) have proved to be capital for the analysis, interpretation and documentation of historical heritage and its preservation for future generations [21, 22, 23].

Architecture and land surveying: Laser 3D scanning technology provides a wealth of realistic 3D modeling buildings with valuable applications to architectural and land surveying studies. For instance, in Figure 3.14 laser scanning was used to determine if the facade of this building (destroyed by fire) can still be saved in rebuilding the structure, as it actually happened at the end. This example shows how beneficial this kind of technology can actually be in real-world architectural projects.

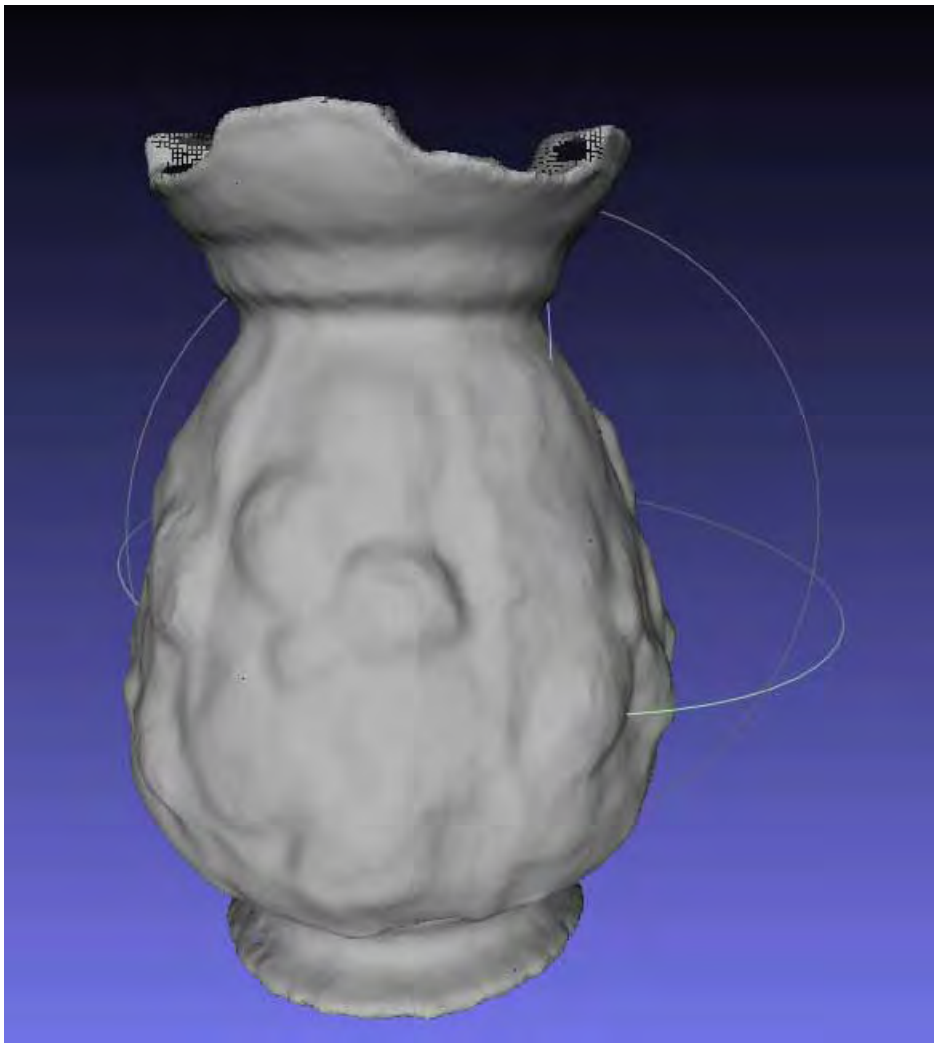

Figure 3.12: Point cloud extracted from an ancient vase captured with a handheld laser scanner.

3.4 Parametric Shapes

Several families of functions can be used for curve and surface reconstruction. Classical mathematical functions used for this task are the so-called parametric free-form shapes such as Bézier, B-splines and NURBS curves and surfaces [38]. Actually, owing to their powerful mathematical properties, their great flexibility and the fact that they can represent smooth shapes with only a few parameters, they are the most common ones in CAD/CAM (Computer-Aided Design/ Manufacturing), computer graphics and animation, virtual reality, and many other fields. Mathematically speaking, they consist of a

Figure 3.13: Example of a total station for LIDAR applications.

linear combination of a set of basis functions (usually called *blending functions*) where the coefficients (usually called the *poles* or *control points*) can readily be used to control the shape of the fitting function, an extremely valuable feature for interactive design.

In general, these blending functions can be classified into two groups: global-support functions and local-support functions. In the former case, the support (the subset of the function domain where the function does not vanish) of the blending functions is the whole domain of the problem, while in the latter case different blending functions can have different supports, which are usually a strict subset of the whole domain. As a consequence, global-support fitting functions exhibit *global control*: any modification of the shape of the curve/surface by moving any pole is automatically propagated throughout the whole curve/surface, something that does not happen in general for local-support fitting functions. In this thesis we will only dis-

Figure 3.14: Laser scanning used to determine if the facade of this building (destroyed by fire) can still be saved in rebuilding the structure (source: *Diamond Land Surveying*)

cuss the case of global-support fitting functions.

In particular, in this thesis we will focus on the Bézier curves and surfaces, which have been intensively used in automotive industry for decades. Mathematically, they are given by a linear combination of basis functions called the Bernstein polynomials with vector coefficients called poles or control points. The curve/surface follows approximately the shape of its control polygon (the collection of segments joining the poles), and hence, it reacts to the movement of its poles by following a push-pull effect. This nice feature was fundamental for the popularization of free-form curves and surfaces for interactive design.

Although nowadays Bézier curves and surfaces have been mostly deprecated in that field, being overtaken by the more powerful B-splines, they are still in use in many other areas. For instance, True Type fonts use composite curves comprised of quadratic Bézier curves. Similarly, all Postscript font outlines are defined in terms of cubic and linear Bézier curves. Other recent applications include, for instance, robot path planning [181] and the determination of the airfoil geometry from a given C_p-distribution [115]. Similarly, Bézier surfaces are still used to represent some particular objects, such as the famous Utah teapot. Because of their simplicity, they are also used today for tasks such as surface blending (the construction of a surface connecting to different surfaces at two ends, usually imposing some kind of geometrical continuity condition at the boundaries). A mathematical definition of the

Bézier curves and surfaces for both the polynomial and the rational cases will be given in Chapters 5 to 8 for the corresponding cases. We refer the reader to these chapters for a detailed mathematical discussion about these curves and surfaces.

Chapter 4

Metaheuristics and Swarm Intelligence for Optimization. The Bat Algorithm

The research work in this thesis is based on the application of a powerful metaheuristics to the curve and surface reconstruction problems. Therefore, it is necessary to understand the fundamentals of the metaheuristics and some of their inherent properties that make this methodology so interesting for this research work. This chapter will provide the reader with a general overview about the metaheuristics, their origin, taxonomy, and main advantages. The discussion will be enriched by the description of some of the most popular metaheuristics described in the literature. Finally, the metaheuristics used in the research work of this thesis, the bat algorithm, will be explained in detail in Section 4.9.

4.1 Metaheuristics

In this section, we discuss the origin of the metaheuristics, their definition and taxonomy, and the main reasons to apply them to our shape reconstruction problem.

4.1.1 Origin of the metaheuristics

Very often, we can find problems that cannot be solved by traditional mathematical optimization techniques, particularly in many (either discrete or continuous) problems from the real world. By "traditional mathematical optimization techniques" we understand techniques ranging from the early works in the XVII, XVIII and XIX centuries, such as the Newton's method (already described in 1669, later published in 1711) to Newton-Raphson (1690),

Langrange multipliers (1778) or least-squares method by Gauss (1795), to more recent techniques such as hill climbing/steepest descent (1847), simplex method (1947), Karush-Kuhn-Tucker multipliers (1939,1951), conjugate gradient (1952), the Nelder-Mead downhill simplex method (1965) or the interior-point method (1984).

Although all those techniques are very powerful and able to solve many different problems, several papers in the literature published during the XX century have reported problems both in combinatorial optimization and in continuous optimization that cannot be properly addressed by these standard mathematical techniques, either because no analytical solution was available, or because the numerical approximation considered instead was unattainable due to extreme computational requirements in terms of time, storage capacity, complexity of the problem, number of parameters involved, or other factors (and very often, even a combination of several factors occurring simultaneously).

This fact can be explained by the limitations of these approaches. A common factor of these methods is that they usually gradient-based, meaning that they require the use of derivatives of the function in order to obtain global/local optima. Consequently, these "classical" methods tend to fail when:

- The objective function is not differentiable
- The objective function is computed through a black-box procedure
- Little or no information about the problem is given
- Derivatives are too difficult or expensive to obtain
- It is expected that many optima exist
- Data are affected by noise, missing data or other artifacts

These drawbacks can sometimes be overcome by using *heuristics*, a kind of rules of procedures aiming at finding solutions to problems by greedy search or "brute-force" strategies. According to [166]:

> *A heuristic is a technique which seeks good (i.e., near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.*

However, heuristics are still too limited for many problems, and fail to provide suitable solutions or they require infinite or too much time to be

obtained. Metaheuristics have been introduced a few decades ago as a way to solve this problem by enhancing the heuristics with more powerful features.

The problems of curve and surface reconstruction posed in these thesis are two of such challenging problems for which neither classical mathematica optimization techniques nor alternative heuristics have provided good enough solutions. Therefore, this thesis proposes the application of a recent metaheuristics to address this problem, as it will be explained in next sections and chapters.

4.1.2 What is a metaheuristics?

By *metaheuristics* we refer to a high-level procedure or strategy designed to help a low-level search strategy or heuristic so that it can find a good solution to an optimization problem. The metaheuristics are usually applied under challenging conditions such as little or incomplete information about the problem to be solved or strong constraints regarding the computational resources [36, 71, 114, 143, 203, 204].

Although the field of metaheuristics is very diverse and we can find many different strategies under its umbrella, a common factor is that they do make few (or none) assumptions about the problem to be solved. This means they are very general and can be applied to many different problems with only minor modifications (if any). The counterpart for this exuberance of applications is the fact that they do not guarantee to reach a global optimum solution.

Most metaheuristic methods aim to obtain the optimum by successive iterations of a given population of candidate solutions that evolve according to some kind of quality metric (the fitness function). Classical examples of this strategy are ant colony optimization (ACO), genetic algorithms (GA), particle swarm optimization (PSO), artificial bee colony (ABC), firefly algorithm (FFA), and many others [87, 113, 116, 200, 202]. In other cases, the method considers only one individual, which is modified along the iterations seeking to improve the solution. Typical examples of this single-population approach are simulated annealing, iterated local search, variable neighborhood search, and guided local search. Finally, metaheuristics can be combined with other local/global search methods or procedures to yield hybrid metaheuristics, including memetic approaches, where individual learning or local improvement search is combined with a global metaheuristics for further improvement.

4.1.3 Taxonomy of metaheuristics

To provide a taxonomy of the metaheuristics is not an easy task, because they are too varied and there are too many different criteria that can be used to

TYPE	METHOD	COMMENTS
EARLY WORKS	Tabu Search	Glover (1986)
	Random Search	Rastrigin (1963)
	Pattern Search	Hooke & Jeeves (1961)
	Grammatical Evolution	Ryan et al (1998)
	Path Relinking	Glover (1996)
PHYSICS & CHEMISTRY INSPIRED	Simulated Annealing	Kirkpatrick et al. (1983)
	Stochastic Diffusion Search	Bishop (1989)
	Harmony Search	Geem et al. (2001)
	Intelligent Water Drops	Shah-Hosseini (2007)
	Electromagnetism Approach	Birbil & Fang (2003)

Figure 4.1: Taxonomy of metaheuristics (Part I).

classify them. Furthermore, the list is very dynamic, as new methods are appearing every year on a regular basis. However, we include here a possible classification as a first attempt to organize the field. They are shown in the tables in Figures 4.1 and 4.2, respectively. The tables shown the different methods arranges in rows and listed in the second column. First column shows the family of methods they belong to, whereas last column shows the name of main developer and the year they were firstly published in the literature. The list is by no means exhaustive, but just a preliminary attempt aimed at showing the high diversity of methods and techniques in the field.

The metaheuristics in Figure 4.1 include the early works, appearing in the 80s (these early approaches include tabu search, random search, or path relinking) and some other techniques with a chemical or physical inspiration. Examples of the latter are stochastic diffusion search or the electromagnetism approach, while that simulated annealing is a classical method with chemical inspiration (from metallurgy, in this case).

The metaheuristics in Figure 4.2 include those based on the principles of swarm intelligence, which will be later explained in Section 4.4, and other bio-inspired techniques. The former group include techniques as popular as ant colony optimization, particle swarm optimization, artificial immune systems, the firefly algorithm, or cuckoo search. The later group consider evolutionary bio-inspired techniques such as genetic programming, evolutionary computing, bacterial foraging, differential evolution, or grammatical evolution, to mention just a few. For the sake of completeness and aiming to help the reader to grasp about their main features, some of these techniques will be

TYPE	METHOD	COMMENTS
SWARM INTELLIGENCE	Ant Colony Optimization	Marc Dorigo (1992)
	Particle Swarm Optimization	Kennedy & Eberhart (1995)
	Artificial Immune Systems	
	Firefly Algorithm	Xin-She Yang (2009)
	Artificial Bee Colony	Karaboga & Basturk (2007)
	Bee Colony Optimization	Lučić & Teodorović (2001)
	Cuckoo Search	Xin-She Yang (2011)
BIO-INSPIRED	Genetic Algorithms	Holland (1975)
	Differential Evolution	Storn & Price (1997)
	Bio-geography optimization	Dan Simon (2008)
	Evolutionary Computing	
	Genetic Programming	Koza (1992)
	Bacterial Foraging	Passino (2002)
	Random Search	Rastrigin (1963)
	Pattern Search	Hooke & Jeeves (1961)
	Grammatical Evolution	Ryan et al (1998)

Figure 4.2: Taxonomy of metaheuristics (Part II).

briefly described throughout this chapter.

4.1.4 Why to use a metaheuristics?

There are four main reasons that explain why it is advisable to apply metaheuristic techniques to solve the curve/surface reconstruction problem:

1. *they can be used even when we have very little information about the problem.* Many alternative methods can only be applied to academic examples whose data have some kind of topological or geometric structure. This does not happen, however, in real-world reverse engineering applications, where typically little or no information about the problem is known beyond the data points.

2. *their objective function does not need to be continuous or differentiable.* In fact, metaheuristics have shown to be able to deal with optimization

problems whose underlying function is non-differentiable (even non-continuous).

3. *they can be successfully applied to multimodal and multivariate nonlinear optimization problems.* Metaheuristic techniques are able to find optimal solutions to nonlinear optimization problems in high-dimensional search spaces. Curve and surface reconstruction are two of such problems. Moreover, due to the (potential) existence of many local optima of the least-squares objective function, it is also a multimodal problem. Metaheuristic techniques are also well suited for multimodal problems.

4. *they can deal with noisy data.* This is a very important issue in many real-world applications. For instance, laser scanner systems yield an enormous amount of irregular data points. They are also known to be less accurate than their contact-based counterparts [106]. As a consequence, reconstruction methods must be robust against this measurement noise.

Other situations where it is advisable to apply metaheuristics are, for instance:

- An easy problem with very large instances
- An easy problem with hard real-time constraints
- Difficult problems with medium size and/or difficult input structures
- Optimization problems with time-consuming objective functions and/or constraints
- Non-analytical models of optimization problems: black box scenario (objective function)
- Non-deterministic complex models: uncertainty, robust optimization, dynamic, multi-objective, etc.
- Non-differentiable (even non-continuous) underlying functions of data
- Input data subjected to noise, imperfect sampling and other artifacts

4.2 Single-Particle Methods: Simulated Annealing

As indicated above, a possible criterion to classify the metaheuristics is based on the population model. Roughly speaking, we can talk about single-particle

methods and multiple-particle methods. The former group was predominant at the early stages of this field, while the latter appeared at more recent stages, when the advantages of having multiple particles cooperating together were highly outlined. In this section, we illustrate the single-particle approach by describing one of the most classical (and probably the most popular) of these approaches: the simulated annealing.

4.2.1 Background

The *simulated annealing* (SA) algorithm is a physics-inspired metaheuristic algorithm originally introduced in 1983 by Kirkpatrick *et al.* [116] in the area of combinatorial optimization. The SA algorithm is rooted in the thermodynamics field, where one studies the thermal energy of a system. In particular, the algorithm was originally motivated by the process of annealing in metallurgy and in ceramic work. The annealing technique consists of changing the internal structure (the crystals) of the material by heating and then slowly cooling it. When we heat the material its crystals evolve towards a more perfect, but also unstable, structure. Then, as the temperature decreases, the system energy also decreases to finally reach a more stable structure hopefully with better crystals than before. During the process, atoms tend to move to configurations that minimize the system energy even if during such migration certain configurations rise the system overall energy (when it stabilizes for a fixed temperature, we call it *thermal equilibrium*). Such moves are more prominent at the beginning of the process that at the end, when the particles loose thermal mobility in order to polish the system inner structure to finally produce a better metal. As a result, the metals become stronger and with better properties, specially if the process is conducted several consecutive times (a process called *re-annealing*). To better understand the connection between the SA algorithm and mechanics science we refer the reader to the seminal paper in [116].

The original SA algorithm is an advanced interpretation of the *Metropolis-Hastings algorithm* [140] to generate sample states of a thermodynamic system, showing the deep connections between statistical mechanics and combinatorial optimization. Given an initial (usually random) *state* in the solution domain, the algorithm iteratively perturbs it. Whenever a better solution is found, the change is always accepted; otherwise, it is accepted only with a certain probability. This probability is higher at the beginning (mimicking what happens in the thermodynamic process at high temperatures) that at the end. In other words, this idea of slow cooling is translated as a slow decrease of the probability of accepting such worse solutions. So essentially the system evolves from a free exploration of the search space at initial stages to a stochastic hill-climbing at latter stages.

This idea of slow cooling after heating is implemented in the SA algorithm as a slow decrease of the probability to accept worse solutions (i.e., accept movements in the search space towards less promising states hoping to escape from local optima). In its most simple version SA can be interpreted as an stochastic hill-climbing: start from a random state in the solution domain and perturb the solution; if the new state has a better fitness, the change is always accepted; otherwise, it is only accepted according to a certain probability. As simple as it can be, simulated annealing has been used with satisfactory results in very diverse fields. Since its publication the algorithm has received a lot of attention from the scientific community, with many real-world applications in the most diverse fields, ranging from the classical NP-hard combinatorial *travelling salesman* problem [136] to the minimization of highly multimodal real-valued functions [186]. See, for instance, [121, 179] for an in-depth review of several simulated annealing applications.

The SA algorithm has two pivotal strengths: on one hand, its easiness of adaptation [179], which allowed researchers and practitioners to apply it to many different real-world problems; on the other hand, some theoretical studies have obtained important results about proofs of convergence [130]. All these results provide the ground for a good choice of suitable values for the parameters of the method, one of the most critical issues for all metaheuristic techniques.

4.2.2 The algorithm

The SA algorithm was designed to compute a good approximation to the global optimum of a given function in a (usually large) search space Ω. Each point $\mathbf{s}$ of the search space Ω is a *state* of some physical system. The goal is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy (i.e., a minimization problem). The function $\mathcal{E}(\mathbf{s})$ to be minimized is the internal energy of the system in that state. This is our *fitness function* (also called *cost function*). At each step, the SA algorithm considers some neighboring state $\mathbf{s}_*$ of the current state $\mathbf{s}$, and probabilistically decides between moving the system to state $\mathbf{s}_*$ or staying in state $\mathbf{s}$. These probabilities ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state good enough for the specific application, or until the method reaches a prescribed number of iterations.

The corresponding pseudo-code of the simulated annealing algorithm used in this paper is shown in *Algorithm 1 Simulated Annealing*. Basically, it can be summarized as follows: we randomly choose an initial state $\mathbf{s}_0$ and the initial system temperature T_0. Then, at each iteration, the SA algorithm replaces the current solution $\mathbf{s}$ by a random "nearby" solution $\mathbf{s}_*$ by using

Algorithm 1 Simulated Annealing

Require: (Initial Parameters)
Initial Temperature T_0
Schedule Criteria $\mathcal{S}_c$
Stop Criteria SC
Neighbourhood function $\mathcal{N} : [0,1]^m \longrightarrow [0,1]^m$
System Energy $\mathcal{E} : [0,1]^m \subseteq \mathbb{R}^m \longrightarrow \mathbb{R}$
Cooling Schedule $\mathcal{S} : \mathbb{R} \longrightarrow \mathbb{R}$ (strictly decreasing)

$T \leftarrow T_0$
Randomly select start state $\mathbf{s}_0 \in [0,1]^m$
$\mathbf{s}^{old} \leftarrow \mathbf{s}_0$
repeat
  repeat
    $\mathbf{s}^{new} \leftarrow \mathcal{N}(\mathbf{s}^{old})$
    $\triangle \leftarrow \mathcal{E}^{new} - \mathcal{E}^{old}$
    if $\triangle < 0$ **then**
      $\mathbf{s}^{old} \leftarrow \mathbf{s}^{new}$
    else
      Randomly compute $u \in Rand(0,1)$
      if $u \leqslant \frac{-\triangle}{T}$ **then**
        $\mathbf{s}^{old} \leftarrow \mathbf{s}^{new}$
      end if
    end if
  until $\mathcal{S}_c$ == **true**
  $T \leftarrow \mathcal{S}(T)$
until SC == **true**
return $\mathbf{s}^{new}$

a neighborhood function $\mathcal{N}$. The new state is chosen with a probability $\mathcal{P}$ that depends on two factors:

- the difference between the corresponding function values at the old and the new states, $\mathcal{E}(\mathbf{s}) - \mathcal{E}(\mathbf{s}_*)$, and
- a global parameter T (the temperature), gradually decreased during the process.

One requirement for this probability function is that $\mathcal{P} > 0$ if $\mathcal{E}(\mathbf{s}) < \mathcal{E}(\mathbf{s}_*)$. This means that the system may move to the new state even when it is worse (has a higher energy) than the current one. This feature prevents the method from becoming stuck in a local minimum. Other typical conditions are that when $T \to 0$, $\mathcal{P} \to 0$ if $\mathcal{E}(\mathbf{s}) < \mathcal{E}(\mathbf{s}_*)$, and $\mathcal{P} \to \alpha > 0$ otherwise. The interpretation of these conditions is that for sufficiently small values of T,

the system will increasingly favor moves that go "downhill" (to lower energy values), and avoid those that go "uphill". In particular, when T becomes 0, the procedure will only make the move if it goes downhill, so this method reduces to a greedy search algorithm. Finally, the probability functions is usually chosen so that the probability of accepting a move decreases as the difference $\mathcal{E}(\mathbf{s}) - \mathcal{E}(\mathbf{s}_*)$ increases. In other words, small uphill moves are preferred over large ones. Under these conditions, the temperature T becomes a critical parameter in describing the evolution of the system, because the system evolution is sensitive to finer (resp. coarser) energy variations for small (resp. large) values of this parameter.

Although it is not strictly necessary, most SA implementations consider an acceptance probability that follows the Metropolis criterion, first introduced by Metropolis et al in [140] as a Monte-Carlo method to simulate the creation of new states in a thermodynamic system. It is given by the following formula:

$$\mathcal{P}(\text{Accept } \mathbf{s}_*) = \begin{cases} 1 & \text{if} \quad \mathcal{E}(\mathbf{s}_*) \leqslant \mathcal{E}(\mathbf{s}) \\ e^{-\dfrac{(\mathcal{E}(\mathbf{s}_*) - \mathcal{E}(\mathbf{s}))}{T}} & \text{otherwise} \end{cases} \tag{4.1}$$

where T represents the system temperature at the iteration where $\mathbf{s}_*$ has been generated. It is easy to see that this choice of the probability function holds all premises stated above. Note also that this function provides an adequate trade-off between exploration and exploitation: at higher temperatures the algorithm explores the search space while near the end it resembles a hill-climbing algorithm, except that now there is always the possibility to accept a worse state.

4.3 Multiple-particle Methods: Genetic Algorithms

Genetic Algorithms (GA) are search procedures based on principles of evolution and natural selection. They can be used in optimization problems where the search of optimal solutions is carried out in a space of solutions coded as finite-length strings called *chromosomes*. To do so, GA handle populations consisting of sets of potential solutions, i.e. the algorithm maintains a population of p individuals $\mathbf{Pop}(g) = \{x_1(g), \ldots, x_p(g)\}$ for each iteration g (also called *generation*), where each individual represents a potential solution of the problem.

Normally the initial population is randomly selected, but some knowledge about the specific problem can be used to include in the initial population

begin
Let g=*0* be the generation counter
Create and initialize a population, **Pop***(0)*
repeat
Evaluate the fitness, $f(x_i(g))$, of each individual x_i of **Pop***(g)*
Select individuals from **Pop***(g)*
Apply crossover with probability p_c to produce offspring
Apply mutation with probability p_m on offspring
Set population of new generation **Pop**$(g+1)$
Advance to the new generation $g = g + 1$
until stopping condition is true
end

Table 4.1: General structure of the genetic algorithm

special potential solutions in order to improve the convergence speed. The size of this initial population is one of the most important aspects to be considered and may be critical in many applications. If the size is too small, the algorithm may converge too quickly, and if it is too large the algorithm may waste computational resources. The population size is usually chosen to be constant although GA with varying population size are also possible. A study about the optimal population size can be found in [72].

Each individual in the population, i.e. each potential solution, must be represented using a genetic representation. Commonly, a binary representation is used, however other approaches are possible. Each one of the potential solutions must be evaluated by means of a fitness function; the result of this evaluation is a measure of individual adaptation.

4.3.1 The algorithm

The algorithm, shown in Table 4.1, is an iterative process in which new populations are obtained using a selection process (reproduction) based on individual adaptation and some "genetic" operators (crossover and mutation). The individuals with the best adaptation measure have more chance of reproducing and generating new individuals by crossing and muting. The reproduction operator can be implemented in several ways, such as tournament, roulette wheel, rank-based, hall of fame and others (see [71, 143]). The selection process is repeated d times and the selected individuals form a tentative new population for further genetic operator actions.

After reproduction some of the members of the new tentative population undergo transformations. A *crossover* operator creates two new individuals

(*offsprings*) by combining parts from two randomly selected individuals of the population. In GA the crossover operator is randomly applied with a specific probability, p_c. A good GA performance requires the choice of a high crossover probability. *Mutation* is a unitary transformation which creates, with certain probability, p_m, a new individual by a small change in a single individual. In this case, a good algorithm performance requires the choice of a low mutation probability (inversely proportional to the population size). The mutation operator guarantees that all the search space has a nonzero probability of being explored. Using these genetic operators, the general structure of the algorithm is described in Table 4.1.

This procedure is repeated several times (thus yielding successive generations) until a termination condition has been reached. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. The general workflow of a classical genetic algorithm is depicted in Figure 4.3.

4.4 Swarm Intelligence

Swarm Intelligence (SI) has been defined as *the property of a system whereby the collective behaviors of (unsophisticated) agents or boids interacting locally with one another and with their environment cause coherent functional global patterns to emerge* [36]. A trademark of SI is the development of a collective behavior arising from decentralized systems comprised of (generally mobile) agents which communicate with each other (either directly or indirectly). Instead of a central behavior determining the evolution of the population, these local interactions between agents lead to the emergence of a global behavior for the swarm. Agents in a SI system obey very simple rules, have limited perception or intelligence and cannot individually carry out the task it intends to. A typical example of SI is the behavior of a flock of birds when moving all together following a common tendency in their displacements. Other examples from nature include ant colonies, animal herding, fish schooling and many others.

Nowadays, swarm intelligence is attracting increasing attention from researchers and practitioners because of its potential applications in several fields [153, 180, 185]. For instance, self-organizing swarm robots can potentially accomplish complex tasks and thus replace sophisticated and expensive robots by simple inexpensive drones [173], a research subfield usually referred to as *swarm robotics*. Military and civil applications of SI have also been reported in the literature with regards to the control of unmanned vehicles [174]. Applications of swarm intelligence range from crowd simulation in computer movies and video games to ant-based routing in telecommuni-

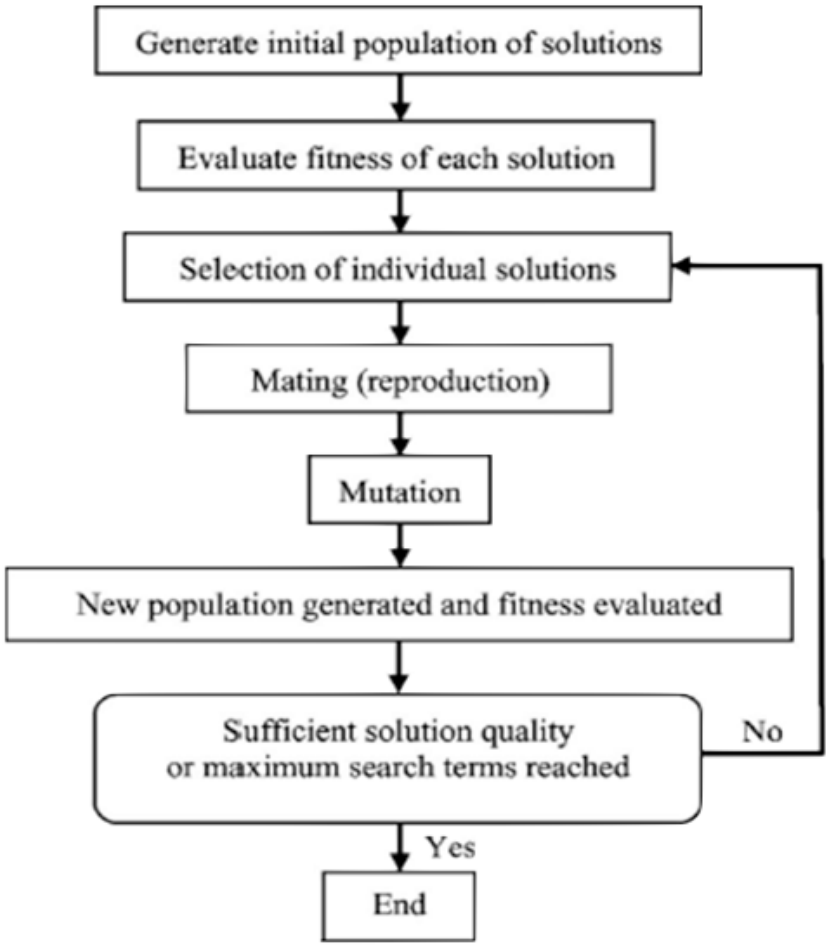

Figure 4.3: General workflow of a genetic algorithm

cation networks, and new exciting areas of research are constantly arising [18, 81, 127, 170].

The objective of SI schemes is to model the simple behaviors of individual agents as well as their interactions with both the environment and their own neighbors in order to obtain more sophisticated behaviors that can be applied to solving complex problems, for instance optimization problems [139, 193]. Classical examples of swarm intelligence approaches are particle swarm optimization (PSO), firefly algorithm (FFA), cuckoo search (CS), and the bat algorithm (the approach used in this thesis).

4.4.1 Exploration and exploitation

It has been shown that the success of bio-inspired techniques relies largely on an adequate trade-off between two opposite terms: *exploration* and *exploitation*. The former accounts for *diversification*, i.e., to generate diverse solutions so as to explore the search space on a global scale of the solution domain, while the latter accounts for *intensification*, i.e., to concentrate the effort in a local region around a current good solution searching for better solutions in its neighborhood.

Different solutions have been proposed in the literature to provide a particular method with the exploration and exploitation features. One of the solutions is to use a specific parameter for this goal. By proper tuning of this parameter, the method can shift from exploration, which is recommended at earlier stages of the method in order to fly all over the search space looking for possible optima, and exploitation at latter stages when an approximate solution has already been found and we seek to improve such a solution locally. A classical example of this strategy is the inertia coefficient for particle swarm optimization (see Section 4.6 below for details).

Other possible solution to this problem is to apply a local search procedure after the convergence of the metaheuristics is attained. In this case we talk about memetic methods. Some examples of this strategy will be discussed in detail in Chapter 9. Finally, another solution consists of applying a coupling of either two different metaheuristics (such as particle swarm optimization and genetic algorithms, for instance) or a metaheuristics and some other methods (i.e., a simplex method, a classical hill climbing, or simply a heuristics).

In general, the most sophisticated methods include some mechanism to shift from exploration to exploitation back and forth, usually in the form of modulating some parameters from end values at both sides of the spectrum. In that case, the parameter tuning plays a very important role in order to promote the most adequate behavior at each stage of the running of the algorithm.

In previous section 4.3 we talked about the genetic algorithms, one of the most classical methodologies in swarm intelligence. In next sections we will also describe some of the other most popular swarm intelligence methods. Our description includes: artificial immune systems (Section 4.5), particle swarm optimization (Section 4.6), firefly algorithm (Section 4.7), and cuckoo search (Section 4.8). Then, we will focus our attention in the bat algorithm, the swarm intelligence metaheuristics used in this thesis. It will be explained in detail in (Section 4.9).

4.5 Artificial Immune Systems

Artificial Immune Systems (AIS) are receiving increasing attention from the scientific community because of their ability to solve complex optimization problems in several fields. Roughly speaking, AIS are a group of computationally intelligent systems inspired by the principles and processes that typically happen at the level of the immune system of humans and other vertebrate.

The immune system is a complex network composed of specialized cells, tissues and organs and is responsible for protecting the organism against diseases caused by pathogenic agents. It is comprised of two distinct parts: the *innate immune system* and the *adaptive immune system*. The former is responsible for powerful immediate but non-specific defenses that prevent or limit infections by most pathogenic microorganisms. The first line of defense consists of physical barriers (such as the skin and mucous membranes), and the second line consists of cells, such as neutrophils, that recognize specific parts of pathogenic microorganisms, called *antigens* (represented by Ag onwards).

If pathogens successfully evade the innate response, there is a second layer of protection, the adaptive immune system, which is activated by the innate response. The adaptive immune system uses somatically generated antigen receptors which are clonally distributed on the two types of lymphocytes: B cells and T cells. These antigen receptors are generated by random processes and, as a consequence, the general design of the adaptive immune response is based upon the clonal selection of lymphocytes expressing receptors with particular specificities. Through this second line of defense, the immune system adapts its response during an infection to improve its recognition of the pathogen. This improved response is preserved even after the pathogen has been eliminated so that any future attack of this pathogen finds a much faster and stronger answer over the time. In other words, the adaptive immune system has some kind of immunological memory, a valuable feature that can be extended to AIS.

There are also some other properties of the immune system of potential interest for computer scientists and engineers [15]:

- *uniqueness*: each individual possesses its own immune system, with its particular features and vulnerabilities;
- *recognition of foreigners*: the molecules that are not native to the body are recognized and eliminated by the immune system;
- *anomaly detection*: the immune system can detect and react to pathogens that the body has never encountered before;

- *distributed detection*: the cells of the immune system are distributed all over the body and, most importantly, are not subject to any centralized control;
- *imperfect detection*: an absolute recognition of the pathogens is not required, hence the system is flexible;
- *reinforcement learning and memory*: the system can "learn" the structures of pathogens, so that future responses to the same pathogens are faster and stronger.

To date, no individual AIS tried to implement all features of a real immune system. Instead, there are several models in AIS, each focused on the implementation of one or, at most, a few of those features. Relevant examples of AIS models include negative selection [48], artificial immune network [107], dendritic cells [77] and clonal selection [16]. Because of its appealing features regarding the optimization of multimodal functions [17], in this section we will focus on the clonal selection theory. It is briefly described in next paragraphs.

4.5.1 The clonal selection theory

The clonal selection principle is a widely accepted theory introduced by Dr. Burnet in 1957 and used to explain the basic features of an adaptive immune response to an antigenic stimulus. When a human or animal is exposed to an Ag, its immune system responds by producing *antibodies* (represented by Ab onwards). Ab's are molecules attached primarily to the surface of B cells whose aim is to recognize and bind to Ag's. Each B cell secretes a single type of Ab, which is relatively specific for the Ag. This level of specificity is evaluated through the *affinity*, which refers to the degree of binding of the cell receptor with the antigen. Under the clonal selection theory, only those cells that recognize the antigens are selected to proliferate. The selected cells are subjected to an *affinity maturation* process, which improves their affinity to the selective Ag's.

In the immune system, the learning process involves raising the relative population size and affinity of those lymphocytes that have been valuable in recognizing a given Ag. For practical reasons, it is convenient to keep a small set of best individuals rather than using a large number of candidate solutions. A *clone* (set of cells generated by mitosis that are the progeny of a single cell) will be created temporarily and further mutated, and those progenies of B cells with low affinity will be discarded through *apoptosis* (or programmed cell death).

Typically, an organism would be expected to encounter a given Ag (or a very similar one) repeatedly during its lifetime. The initial exposure to

an Ag that stimulates an adaptive immune response is handled by a small number of low-affinity B cells, each producing an Ab type of different affinity. The effectiveness of the immune response to secondary encounters is enhanced considerably by the presence of memory cells associated with the first infection, capable of producing high-affinity Ab's just after subsequent encounters.

Ab's in a memory response have, on average, a higher affinity than those of the early primary response. This phenomenon is referred to as the *maturation* of the immune response. This maturation requires the Ag-binding sites of the Ab molecules to be structurally different from those present in the primary response. Then, random changes (*mutation*) are introduced into the genes responsible for the Ag-Ab interactions, leading occasionally to an increase in the affinity of the Ab. A rapid accumulation of mutations is necessary for a fast maturation of the immune response, even although most of the changes lead to poorer Ab's. Indeed, when a B cell recognizes an antigen, it is stimulated to divide (or proliferate). During proliferation, the B cell receptor locus undergoes an extremely high rate of somatic mutation that is at least five or six orders of magnitude greater than the normal rate of mutation across the genome. This process, referred to as *somatic hypermutation* (SHM), is regulated by the fact that cells with low-affinity receptors may be further mutated and die if they do not improve their antigenic affinity, while in cells with high-affinity Ab receptors, hypermutation may become gradually inactive.

4.5.2 Clonal selection algorithm (CSA)

The clonal selection algorithm (CSA) is an AIS scheme based on the clonal selection theory discussed in previous section. The algorithm, initially introduced by De Castro and Von Zuben in [16] and formally described in [17], is based on two basic principles: (1) only the cells recognizing the antigen are selected for growing (*mutation* and *cloning*); (2) the affinity of the selected cell to the antigen is increased by *affinity maturation* process. This algorithm has shown to be very well suited for optimization problems, having been successfully applied to problems such as character recognition and multimodal optimization of functions with very good performance. The main immune features taken into account in the algorithm are [17]:

1. maintenance of a specific memory set;
2. selection and cloning of the most stimulated Ab's;
3. death of nonstimulated Ab's;
4. affinity maturation;

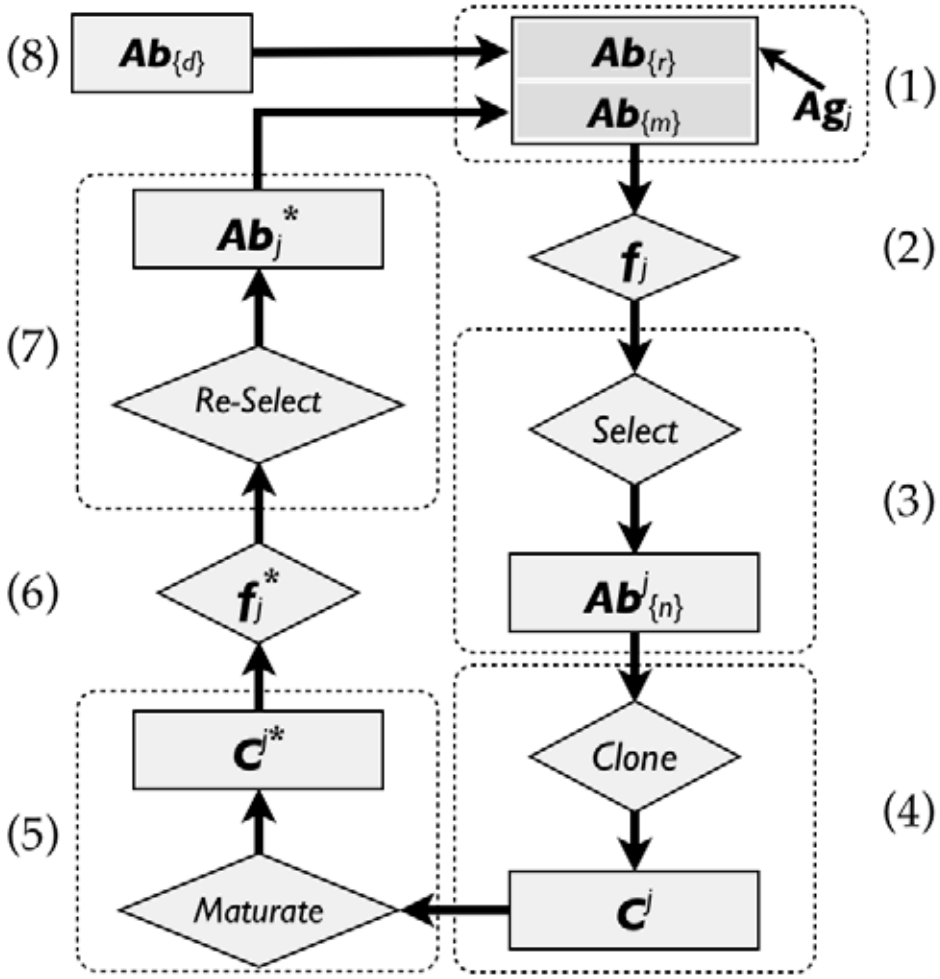

Figure 4.4: Flowchart of the clonal selection algorithm for pattern recognition purposes. Some modifications might be required for its application to unsupervised learning.

5. re-selection of the clones proportionally to their antigenic affinity; and

6. generation and maintenance of diversity.

Typically, the given problem to be solved is represented through a Ag-Ab codification (binary or real-valued) and a distance measure (called the *affinity measure*), used to calculate the degree of interaction between these molecules. Affinity between an antibody and an antigen, represented by $Af(Ab, Ag)$, can be estimated by distance measure between two arrays (or vectors) by using different methods. If the vectors representing antigens and antibodies are real-valued vectors, then Manhattan or Euclidian distance measures can be used; if they are represented by binary symbols, then the Hamming distance

is usually applied (see [14] for more details).

Figure 4.4 shows the flowchart of the clonal selection algorithm as originally proposed for pattern recognition purposes [17]. The algorithm considers two repertoires (populations): a set of antigens $\mathbf{Ag}_{\{M\}}$ and a set of antibodies $\mathbf{Ab}_{\{N\}}$. Similar to [17], cardinality is indicated by the subindexes within brackets for clarity. The set of antibodies $\mathbf{Ab}_{\{N\}}$ is further divided into two subsets: memory Ab repertoire, $\mathbf{Ab}_{\{m\}}$, and remaining Ab repertoire, $\mathbf{Ab}_{\{r\}}$, such that $m + r = N$. The algorithm also keeps track of two other sets: the set $\mathbf{Ab}_{\{n\}}$ of the n Ab's with the highest affinities to a given Ag, and the set $\mathbf{Ab}_{\{d\}}$ of the d new Ab's that will replace the low-affinity Ab's from $\mathbf{Ab}_{\{r\}}$.

The algorithm can be summarized as follows:

1. Random choice of an antigen $\mathbf{Ag}_j$. It is presented to all antibodies of $\mathbf{Ab}_{\{N\}}$.

2. Compute the vector affinity $\mathbf{f} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N)$ where $\mathbf{f}_i = Af(Ab_i, \mathbf{Ag}_j)$.

3. Select the n highest affinity components of $\mathbf{f}$ to generate $\mathbf{Ab}_{\{n\}}$.

4. Elements of $\mathbf{Ab}_{\{n\}}$ will be cloned adaptively. The number of clones is proportional to the affinity: the higher the affinity, the higher the number of clones. Such amount is given by: $N_c = \sum_{h=1}^{n} round\left(\frac{\Lambda.N}{h}\right)$ where N_c represents the number of clones, Λ is a positive number that plays the role of a multiplying factor, N is the total number of Ab's and $round(.)$ is the operator that rounds its argument toward the closest integer.

5. The clones in the set resulting from the previous step are subjected to somatic hypermutation. The affinity maturation rate is inversely proportional to the antigenic affinity: the higher the affinity, the smaller the maturation rate.

6. Compute the vector affinity of $\mathbf{Ag}_j$ with respect to the new matured clones.

7. From this set of matured clones, select the one with the highest affinity to be candidate to enter into the set $\mathbf{Ab}_{\{m\}}$. If $Af(Ab_k, \mathbf{Ag}_j) > Af(Ab_l, \mathbf{Ag}_j)$ for a given $Ab_l \in \mathbf{Ab}_{\{n\}}$, then Ab_k will replace Ab_l.

8. Replace the d Ab's with lowest affinity in $\mathbf{Ab}_{\{r\}}$ by new, randomly generated individuals in $\mathbf{Ab}_{\{d\}}$, inserted into $\mathbf{Ab}_{\{r\}}$ in order to preserve the diversity of population.

Each execution of the previous steps for $\mathbf{Ag}_j$, $j = 1, \ldots, M$, is called a generation. The algorithm is repeated for a certain number of generations, N_{gen}, a parameter that depends on the specific problem under analysis.

The algorithm described above is only suitable for supervised problems, for which an explicit $\mathbf{Ag}_{\{M\}}$ population is available for recognition. To overcome this limitation, authors in [17] proposed a modified version of CSA for multimodal optimization problems. Main changes in their modified version are:

- There is no need to maintain a separate subset of memory $\mathbf{Ab}_{\{m\}}$, since no specific $\mathbf{Ag}_j$ has to be recognized. Instead, the whole population of antibodies will compose the memory set.
- The affinity measure function corresponds to the evaluation of the least-squares function, so that each Ab represents a potential solution of the problem.
- Several Ab's with high affinity are selected in step 7 of the algorithm, rather than just the best one.
- All Ab's in the population can be selected for cloning in step 3, so no need to maintain set $\mathbf{Ab}_{\{n\}}$.
- In that case, the affinity proportionate cloning is no longer necessary. All antibodies can be cloned at the same rate (i.e. the number of clones generated for each antibody will be the same).

4.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic algorithm based on the evolution of populations for problem solving. In this methodology the particle swarm simulates the social optimization commonly found in animal communities with a high degree of organization (flocks of birds, herds, schools of fishes). For a given problem, some fitness function is needed to evaluate the proposed solution. In order to get a good one, PSO methods incorporate both a global tendency for the movement of the set of individuals and local influences from neighbors [113].

The original PSO algorithm was first reported in 1995 by Kennedy and Eberhart in [113] (see also [32] for further details). The reader is referred to the excellent book in [114]. See also [36] for a gentle analysis of PSO from a computational point of view.

Algorithm 1 Particle Swarm Optimization

{Initialization}
    $s \leftarrow 0$ /* s: time variable */
    for $i = 1$ to m **do** /* m: size of the swarm */
        Initialize vectors V_i and P_i to random values
        $P_i^b \leftarrow P_i$
    end for
    $P_g^b \leftarrow best\{P_i^b\,;\, i = 1, \ldots, m\}$

{Main Loop}
    while (not termination condition) **do**
        {Evaluation Loop}
            for $i = 1$ to m **do**
                if $f(P_i)$ is better than $f(P_i^b)$ **then** /* f: fitness function */
                    $P_i^b \leftarrow P_i$
                end if
                if $f(P_i^b)$ is better than $f(P_g^b)$ **then**
                    $P_g^b \leftarrow P_i^b$
                end if
            end for
        {Update Loop}
            for $i = 1$ to m **do**
                $V_i \leftarrow w.V_i + \gamma_1.R_1(0,1).(P_g^b - P_i) + \gamma_2.R_2(0,1).(P_i^b - P_i)$
                $P_i \leftarrow P_i + V_i$
            end for
            $s \leftarrow s + 1$
    end while

All particle swarm optimization procedures start by choosing a population (swarm) of random candidate solutions in a multidimensional space (the search space), called *particles*. Then they are displaced throughout their domain looking for an optimum taking into account global and local influences, the latest coming form the neighborhood of each particle. To this purpose, all particles have a position and a velocity and evolve "flying" all through the hyperspace according to two essential reasoning capabilities: a memory of their own best position and knowledge of the global or their neighborhood's best.

The meaning of the "best" must be understood in the context of the problem to be solved. In a minimization problem (like the problems addressed in this thesis) that means the position with the smallest value for the target

function.

The dynamics of the particle swarm is considered along successive iterations, like time instances. Each particle modifies its position P_i along the iterations, keeping track of its best position in the variables domain implied in the problem. This is made by storing for each particle the coordinates P_i^b associated with the best solution (fitness) it has achieved so far along with the corresponding fitness value, f_i^b. These values account for the *memory* of the best particle position.

In addition, members of a swarm can communicate good positions to each other, so they can adjust their own position and velocity according to this information. To this purpose, we also collect the best fitness value among all the particles in the population, f_g^b, and its position P_g^b from the initial iteration. This is a global information for modifying the position of each particle.

Finally, the evolution for each particle i is given by:

$$V_i(k+1) = w\,V_i(k) + \gamma_1 R_1[P_g^b(k) - P_i(k)] + \gamma_2 R_2[P_i^b(k) - P_i(k)] \quad (4.2)$$

$$P_i(k+1) = P_i(k) + V_i(k) \quad (4.3)$$

where $P_i(k)$ and $V_i(k)$ are the position and the velocity of particle i at time k respectively, w is called *inertia weight* and decide how much the old velocity will affect the new one and coefficients γ_1 and γ_2 are constant values called *learning factors*, which decide the degree of affection of P_g^b and P_i^b. In particular, γ_1 is a weight that accounts for the "social" component, while γ_2 represents the "cognitive" component, accounting for the memory of an individual particle along the time. The graphical meaning of all these terms is shown in Figure 4.5. Two random numbers, R_1 and R_2, with uniform distribution on $[0, 1]$ are included to enrich the searching space. Finally, a fitness function must be given to evaluate the quality of a position.

This procedure is repeated several times (thus yielding successive generations) until a termination condition is reached. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. Global PSO procedure is briefly sketched in *Algorithm 1 Particle Swarm Optimization*.

The driving force of PSO is social interaction: particles learn from each other and evolve in order to become more similar to their "better" neighbors. This evolution is strongly related to the social structure of the swarm, usually referred to as the *communication topology*. Highly connected social networks in which most individuals communicate with each other have the advantage that good information quickly disseminates throughout the swarm. This means faster convergence to a solution than in less connected networks. At their turn, sparsely connected organizations are less susceptible to be

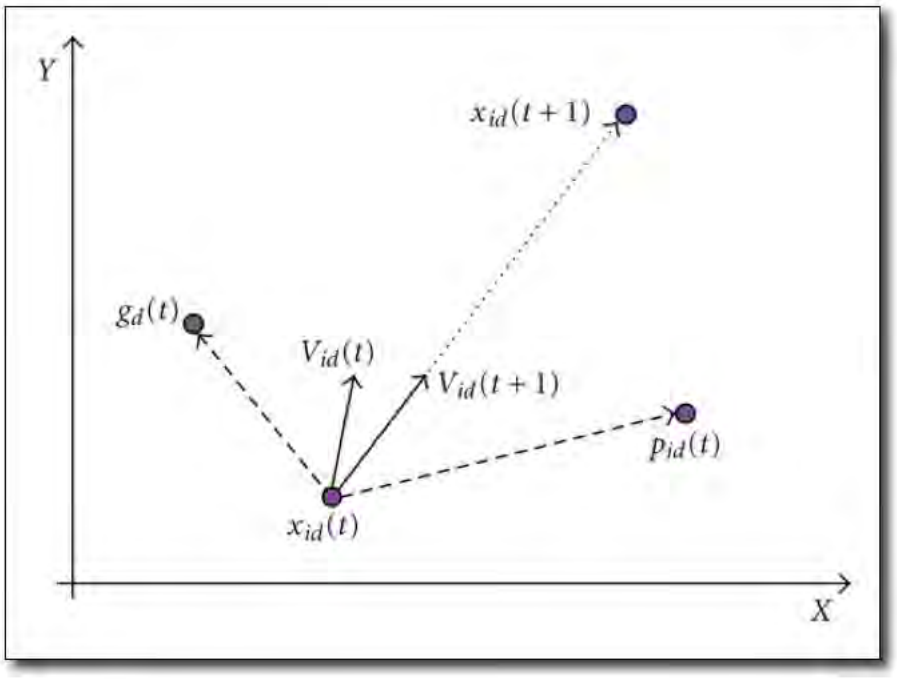

Figure 4.5: Graphical interpretation of the different terms of the evolution equations of the particle swarm optimization method (see Eqs. (4.2-4.3)).

trapped in local minima. Consequently, a trade-off between fast convergence and susceptibility to local minima must be achieved. As a result, several communication topologies have been developed.

The first implementation of PSO was based on the *star* social structure, where all particles are connected with each other. In such a case, each particle is attracted towards the best solution found by the entire swarm. The resulting algorithm is described by Eqs. (4.2)-(4.3) and is usually called *gbest* (global best). Other social structures such as ring, wheel, pyramid, four clusters, von Neumann and so on are also possible. They lead to *lbest* (local best) models where the best fitness value and position of the swarm must be replaced by those of the neighborhood (and therefore Eqs. (4.2)-(4.3) must be modified accordingly). This variation (called *local PSO*) allows parallel exploration of the search space while reducing the probability of the PSO to fall into local minima, at the price of slow convergence speed. In general, smaller neighborhoods lead to slower convergence while larger neighborhoods yield faster convergence. Because of this reason, most PSO methods consider the global approach (i.e. the entire swarm) instead of a local approach (the neighborhood of each particle). However, the best option is always problem-dependent and no universal rule can be stated in this regard.

4.7 The Firefly Algorithm

The *firefly algorithm* is a nature-inspired metaheuristic algorithm introduced in 2008 by Xin-She Yang to solve optimization problems [200, 202] (see also [182] for a recent modified version of this algorithm). The algorithm is based on the social flashing behavior of fireflies in nature.

The key ingredients of the method are the variation of light intensity and formulation of attractiveness. In general, the attractiveness of an individual is assumed to be proportional to their brightness, which in turn is associated with the encoded objective function. The reader is kindly referred to [203] for a comprehensive review of the firefly algorithm and other nature-inspired metaheuristic approaches. See also [204] for a gentle introduction to metaheuristic applications in engineering optimization.

In the firefly algorithm, there are three particular idealized rules, which are based on some of the major flashing characteristics of real fireflies [200]. They are:

1. All fireflies are unisex, so that one firefly will be attracted to other fireflies regardless of their sex;

2. The degree of attractiveness of a firefly is proportional to its brightness, which decreases as the distance from the other firefly increases due to the fact that the air absorbs light. For any two flashing fireflies, the less brighter one will move towards the brighter one. If there is not a brighter or more attractive firefly than a particular one, it will then move randomly;

3. The brightness or light intensity of a firefly is determined by the value of the objective function of a given problem. For instance, for maximization problems, the light intensity can simply be proportional to the value of the objective function.

The distance between any two fireflies i and j, at positions $\mathbf{X}_i$ and $\mathbf{X}_j$, respectively, can be defined as a Cartesian or Euclidean distance as follows:

$$r_{ij} = ||\mathbf{X}_i - \mathbf{X}_j|| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{4.4}$$

where $x_{i,k}$ is the k-th component of the spatial coordinate $\mathbf{X}_i$ of the i-th firefly and D is the number of dimensions.

In the firefly algorithm, as attractiveness function of a firefly j one should select any monotonically decreasing function of the distance to the chosen firefly, e.g., the exponential function:

$$\beta = \beta_0 e^{-\gamma r_{ij}^{\mu}} \qquad (\mu \geqslant 1) \tag{4.5}$$

where r_{ij} is the distance defined as in Eq. (4.4), β_0 is the initial attractiveness at $r = 0$, and γ is an absorption coefficient at the source which controls the decrease of the light intensity.

The movement of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is governed by the following evolution equation:

$$\mathbf{X}_i = \mathbf{X}_i + \beta_0 e^{-\gamma r_{ij}^{\mu}} (\mathbf{X}_j - \mathbf{X}_i) + \alpha \left(\sigma - \frac{1}{2} \right) \tag{4.6}$$

where the first term on the right-hand side is the current position of the firefly, the second term is used for considering the attractiveness of the firefly to light intensity seen by adjacent fireflies, and the third term is used for the random movement of a firefly in case there are not any brighter ones. The coefficient α is a randomization parameter determined by the problem of interest, while σ is a random number generator uniformly distributed in the space $[0, 1]$.

The firefly algorithm considers an initial population of N_P individuals. This population size is maintained along the iterations. In each generation, the individuals (fireflies) undergo transformation (movement) according to Eq. (4.6). The process is repeted iteratively until the maximum number of iterations is reached.

The method described in previous paragraphs corresponds to the original version of the firefly algorithm (FFA), as originally developed by its inventor. Since then, many different modifications and improvements on the original version have been developed, including the discrete FFA, multi-objective FFA, chaotic FFA, parallel FFA, elitist FFA, Lagrangian FFA, and many others, including its hybridization with other techniques. The interested reader is referred to the nice paper in [39] for a comprehensive, updated review and taxonomic classification of the firefly algorithms and all its variants and applications.

4.8 The Cuckoo Search Algorithm

Cuckoo search (CS) is a nature-inspired population-based metaheuristic algorithm originally proposed by Yang and Deb in 2009 to solve optimization problems [207]. The algorithm is inspired by the obligate interspecific brood-parasitism of some cuckoo species that lay their eggs in the nests of host birds of other species with the aim of escaping from the parental investment in raising their offspring. This strategy is also useful to minimize the risk of egg loss to other species, as the cuckoos can distributed their eggs amongst a number of different nests.

Of course, sometimes it happens that the host birds discover the alien eggs in their nests. In such cases, the host bird can take different responsive

Algorithm: Firefly Algorithm

begin
Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_D)^T$
Generate initial population of n fireflies $\mathbf{x}_i$ $(i = 1, 2, \ldots, n)$
Formulate light intensity I associated with $f(\mathbf{x})$
Define absorption coefficient γ
while $(t < MaxGeneration)$ or (stop criterion)
for $i = 1$ to n
for $j = 1$ to n
if $I(i) > I(j))$
then move firefly i towards j
end if
Vary attractiveness with distance r via $e^{-\gamma r}$
Evaluate new solutions and update light intensity
end for
end for
Rank fireflies and find the current best
end while
Post-processing the results and visualization
end

Table 4.2: Firefly Algorithm pseudocode

actions varying from throwing such eggs away to simply leaving the nest and build a new one elsewhere. However, the brood parasites have at their turn developed sophisticated strategies (such as shorter egg incubation periods, rapid nestling growth, egg coloration or pattern mimicking their hosts, and many others) to ensure that the host birds will care for the nestlings of their parasites.

This interesting and surprising breeding behavioral pattern is the metaphor of the cuckoo search metaheuristic approach for solving optimization problems. In the cuckoo search algorithm, the eggs in the nest are interpreted as a pool of candidate solutions of an optimization problem while the cuckoo egg represents a new coming solution. The ultimate goal of the method is to use these new (and potentially better) solutions associated with the parasitic cuckoo eggs to replace the current solution associated with the eggs in the nest. This replacement, carried out iteratively, will eventually lead to a very good solution of the problem.

In addition to this representation scheme, the cuckoo search algorithm is also based on three idealized rules [207, 208]:

1. Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;

2. The best nests with high quality of eggs (solutions) will be carried over to the next generations;

3. The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

For simplicity, the third assumption can be approximated by a fraction p_a of the n nests being replaced by new nests (with new random solutions at new locations). For a maximization problem, the quality or fitness of a solution can simply be proportional to the objective function. However, other (even more sophisticated) expressions for the fitness function can also be defined.

Based on these three rules, the basic steps of the cuckoo search algorithm can be summarized as shown in the pseudo-code reported in Table 4.3. Basically, the cuckoo search algorithm starts with an initial population of n host nests and it is performed iteratively. In the original proposal, the initial values of the jth component of the ith nest are determined by the expression:

$$x_i^j(0) = rand.(up_i^j - low_i^j) + low_i^j$$

where up_i^j and low_i^j represent the upper and lower bounds of that jth component, respectively, and $rand$ represents a standard uniform random number on the open interval $(0, 1)$. Note that this choice ensures that the initial values of the variables are within the search space domain. These boundary conditions are also controlled in each iteration step.

For each iteration g, a cuckoo egg i is selected randomly and new solutions $\mathbf{x}_i(g + 1)$ are generated by using the Lévy flight, a kind of random walk in which the steps are defined in terms of the step-lengths, which have a certain probability distribution, with the directions of the steps being isotropic and random. According to the original creators of the method, the strategy of using Lévy flights is preferred over other simple random walks because it leads to better overall performance of the CS. The general equation for the Lévy flight is given by:

$$\mathbf{x}_i(g + 1) = \mathbf{x}_i(g) + \alpha \oplus levy(\lambda) \tag{4.7}$$

where g indicates the number of the current generation, and $\alpha > 0$ indicates the step size, which should be related to the scale of the particular problem

Algorithm: Cuckoo Search via Lévy Flights

begin
Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_D)^T$
Generate initial population of n host nests $\mathbf{x}_i$ $(i = 1, 2, \ldots, n)$
while $(t < MaxGeneration)$ or (stop criterion)
Get a cuckoo (say, i) randomly by Lévy flights
Evaluate its fitness F_i
Choose a nest among n (say, j) randomly
if $(F_i > F_j)$
Replace j by the new solution
end
A fraction (p_a) of worse nests are abandoned and new ones are built via Lévy flights
Keep the best solutions (or nests with quality solutions)
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

Table 4.3: Cuckoo search algorithm via Lévy flights as originally proposed by Yang and Deb in [207, 208].

under study. The symbol $\oplus$ is used in Eq. (4.7) to indicate the entry-wise multiplication. Note that Eq. (4.7) is essentially a Markov chain, since next location at generation $g + 1$ only depends on the current location at generation g and a transition probability, given by the first and second terms of Eq. (4.7), respectively. This transition probability is modulated by the Lévy distribution as:

$$levy(\lambda) \sim g^{-\lambda}, \qquad (1 < \lambda \leqslant 3) \tag{4.8}$$

which has an infinite variance with an infinite mean. Here the steps essentially form a random walk process with a power-law step-length distribution with a heavy tail.

From the computational standpoint, the generation of random numbers with Lévy flights is comprised of two steps: firstly, a random direction according to a uniform distribution is chosen; then, the generation of steps following the chosen Lévy distribution is carried out. The authors suggested to use the so-called Mantegna's algorithm for symmetric distributions, where

"symmetric" means that both positive and negative steps are considered (see [203] for details). Their approach computes the factor:

$$\hat{\phi} = \left(\frac{\Gamma(1+\hat{\beta}).sin\left(\frac{\pi.\hat{\beta}}{2}\right)}{\Gamma\left(\left(\frac{1+\hat{\beta}}{2}\right).\hat{\beta}.2^{\frac{\hat{\beta}-1}{2}}\right)} \right)^{\frac{1}{\hat{\beta}}} \tag{4.9}$$

where Γ denotes the Gamma function and $\hat{\beta} = \frac{3}{2}$ in the original implementation by Yang and Deb [208]. This factor is used in Mantegna's algorithm to compute the step length s as:

$$\varsigma = \frac{u}{|v|^{\frac{1}{\hat{\beta}}}} \tag{4.10}$$

where u and v follow the normal distribution of zero mean and deviation σ_u^2 and σ_v^2, respectively, where σ_u obeys the Lévy distribution given by Eq. (4.9) and $\sigma_v = 1$. Then, the stepsize η is computed as:

$$\eta = 0.01\,\varsigma\,(\mathbf{x} - \mathbf{x}_{best}) \tag{4.11}$$

where ς is computed according to Eq. (4.10). Finally, $\mathbf{x}$ is modified as: $\mathbf{x} \leftarrow \mathbf{x} + \eta.\Upsilon$ where Υ is a random vector of the dimension of the solution $\mathbf{x}$ and that follows the normal distribution $N(0,1)$.

The cuckoo search method then evaluates the fitness of the new solution and compares it with the current one. In case the new solution brings better fitness, it replaces the current one. On the other hand, a fraction of the worse nests (according to the fitness) are abandoned and replaced by new solutions so as to increase the exploration of the search space looking for more promising solutions. The rate of replacement is given by the probability p_a, a parameter of the model that has to be tuned for better performance. Moreover, for each iteration step, all current solutions are ranked according to their fitness and the best solution reached so far is stored as the vector $\mathbf{x}_{best}$ (used, for instance, in Eq. (4.11)).

This algorithm is applied in an iterative fashion until a stopping criterion is met. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. Other stopping criteria might be however be used instead.

4.9 The Bat Algorithm

As we previously pointed out, in this thesis we propose to apply a recently introduced bio-inspired metaheuristics to the curve and surface reconstruction problems. In this section, we describe this bat algorithm in detail. We

start with the description of the main idealized rules this method is based on. Then, a detailed description of the algorithm is given.

4.9.1 Basic rules

The *bat algorithm* is a bio-inspired population-based metaheuristic algorithm originally proposed by Xin-She Yang in 2010 to solve optimization problems [201, 209]. The algorithm is based on the echolocation behavior of bats. The author focused particularly on microbats, as they use a type of sonar called echolocation, with varying pulse rates of emission and loudness, to detect prey, avoid obstacles, and locate their roosting crevices in the dark.

Despite the short time since its appearance, the bat algorithm has already been applied to several engineering and industrial problems. Simultaneously, some modifications and improvements on the original version have been developed, such as the multi-objective bat algorithm [205], directed artificial bat algorithm [167], binary bat algorithm [142], and others. The interested reader is referred to the general paper in [206] for a comprehensive, updated review and taxonomic classification of the bat algorithm and all its variants and applications.

In this thesis we consider the standard bat algorithm, as described in the original paper in [201]. According to that source, the idealization of the echolocation of microbats can be summarized as follows: (see [201] for details):

1. Bats use echolocation to sense distance and distinguish between food, prey and background barriers.

2. Each virtual bat flies randomly with a velocity $\mathbf{v}_i$ at position (solution) $\mathbf{x}_i$ with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. As it searches and finds its prey, it changes wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r, depending on the proximity of the target.

3. It is assumed that the loudness will vary from an (initially large and positive) value A_0 to a minimum constant value A_{min}.

In order to apply the bat algorithm for optimization problems more efficiently, some additional assumptions are strongly advisable. In general, we assume that the frequency f evolves on a bounded interval $[f_{min}, f_{max}]$. This means that the wavelength λ is also bounded, because f andλ are related to each other by the fact that the product $\lambda.f$ is constant. For practical reasons, it is also convenient that the largest wavelength is chosen such that it is comparable to the size of the domain of interest (the search space, for optimization problems). For simplicity, we can assume that $f_{min} = 0$, so

Algorithm 2 Bat Algorithm

Require: (Initial Parameters)
Population size: $\mathcal{P}$
Maximum number of generations: $\mathcal{G}_{max}$
Loudness: $\mathcal{A}$
Pulse rate: r
Maximum frequency: f_{max}
Dimension of the problem: d
Objective function: $\phi(\mathbf{x})$, with $\mathbf{x} = (x_1, \ldots, x_d)^T$
Random vectors: $\Theta = (\theta_1, \ldots, \theta_{\mathcal{P}}), \Psi = (\psi_1, \ldots, \psi_{\mathcal{P}})$ with $\theta_k, \psi_k \in U(0, 1)$

$g \leftarrow 0$ //g: generation index
for $i = 1$ **to** $\mathcal{P}$ **do**
  Initialize the location and velocity $\mathbf{x}_i$ and $\mathbf{v}_i$ //Initialization phase
  Define pulse frequency f_i at $\mathbf{x}_i$
  Initialize pulse rates r_i and loudness $\mathcal{A}_i$
end for
while $g \leqslant \mathcal{G}_{max}$ **do**
  for $i = 1$ **to** $\mathcal{P}$ **do**
    Generate new solutions by adjusting frequency,
    and updating locations and velocities //eqns. (4.12)-(4.14)
    if $\theta_i > r_i$ **then**
      $\mathbf{s}^{best} \leftarrow \mathbf{s}^g$ //select the current best global solution
      $\mathbf{ls}^{best} \leftarrow \mathbf{ls}^g$ //generate a local solution around $\mathbf{s}^{best}$
    end if
    Generate a new solution by local random walk //eqn. (4.15)
    if $\psi_i < \mathcal{A}_i$ *and* $\phi(\mathbf{x_i}) < \phi(\mathbf{x}^*)$ **then**
      Accept new solutions
      Increase r_i and decrease $\mathcal{A}_i$ //eqns. (4.16)-(4.17)
    end if
  end for
  $g \leftarrow g + 1$
end while
Rank the bats and find current best $\mathbf{x}^*$
return $\mathbf{x}^*$

$f \in [0, f_{max}]$. The rate of pulse can simply be in the range $r \in [0, 1]$, where 0 means no pulses at all, and 1 means the maximum rate of pulse emission. With these idealized rules indicated above, the basic pseudo-code of the bat algorithm is shown in *Algorithm 2 Bat Algorithm*. It is described in next section.

4.9.2 The algorithm

Basically, the algorithm considers an initial population of $\mathcal{P}$ individuals (bats). Each bat, representing a potential solution of the optimization problem, has a location $\mathbf{x}_i$ and velocity $\mathbf{v}_i$. The algorithm initializes these variables with random values within the search space. Then, the pulse frequency, pulse rate, and loudness are computed for each individual bat (lines 2-6). Then, the swarm evolves in a discrete way over generations (line 7), like time instances (line 21) until the maximum number of generations, $\mathcal{G}_{max}$, is reached (line 22). For each generation g and each bat (line 8), new frequency, location and velocity are computed (lines 9-10) according to the following evolution equations:

$$f_i^g = f_{min}^g + \beta(f_{max}^g - f_{min}^g) \tag{4.12}$$

$$\mathbf{v}_i^g = \mathbf{v}_i^{g-1} + \left[\mathbf{x}_i^{g-1} - \mathbf{x}^*\right] f_i^g \tag{4.13}$$

$$\mathbf{x}_i^g = \mathbf{x}_i^{g-1} + \mathbf{v}_i^g \tag{4.14}$$

where $\beta \in [0, 1]$ follows the random uniform distribution, and $\mathbf{x}^*$ represents the current global best location (solution), which is obtained through evaluation of the objective function at all bats and ranking of their fitness values. The superscript $(.)^g$ is used to denote the current generation g.

It is worthwhile to remark a certain similarity of the evolutions equations (4.13)-(4.14) to those of the velocity and position in standard particle swarm optimization (PSO), as f_i controls the pace and range of the movement of the particles of the swarm. From this viewpoint, the bat algorithm is a balanced combination of the standard PSO and the local search modulated by the loudness and pulse rate.

The best current solution and a local solution around it are probabilistically selected according to some given criteria (lines 11-14). Then, search is intensified by a local random walk (line 15). For this local search, once a solution is selected among the current best solutions, it is perturbed locally through a random walk of the form:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon \mathcal{A}^g \tag{4.15}$$

where ϵ is a random number with uniform distribution on the interval $[-1, 1]$ and $\mathcal{A}^g =< \mathcal{A}_i^g >$, is the average loudness of all the bats at generation g.

If the new solution achieved is better than the previous best one, it is probabilistically accepted depending on the value of the loudness. In that case, the algorithm increases the pulse rate and decreases the loudness (lines 16-19). This process is repeated for the given number of generations. In general, the loudness decreases once a bat finds its prey (in our analogy, once a new best solution is found), while the rate of pulse emission decreases. For

simplicity, the following values are commonly used: $\mathcal{A}_0 = 1$ and $\mathcal{A}_{min} = 0$, assuming that this latter value means that a bat has found the prey and temporarily stop emitting any sound. However, any other value within its range can also be feasible (see our discussion about parameter tuning in Chapters 5 to 9 for details).
The evolution rules for loudness and pulse rate are as follows:

$$\mathcal{A}_i^{g+1} = \alpha \mathcal{A}_i^g \tag{4.16}$$

$$r_i^{g+1} = r_i^0[1 - exp(-\gamma g)] \tag{4.17}$$

where α and γ are constants. Note that for any $0 < \alpha < 1$ an any $\gamma > 0$ we have:

$$\mathcal{A}_i^g \to 0, \quad r_i^g \to r_i^0, \quad \text{as} \;\; g \to \infty \tag{4.18}$$

In general, each bat should have different values for loudness and pulse emission rate, which can be computationally achieved by randomization. To this aim, we can take an initial loudness $\mathcal{A}_i^0 \in (0, 2)$ while the initial emission rate r_i^0 can be any value in the interval $[0, 1]$. Loudness and emission rates will be updated only if the new solutions are improved, an indication that the bats are moving towards the optimal solution. As a result, the bat algorithm applies a parameter tuning technique to control the dynamic behavior of a swarm of bats. Similarly, the balance between exploration and exploitation can be controlled by tuning algorithm-dependent parameters. This means that we do not need to apply hybridization to shift from exploration and exploitation and conversely. However, some types of local search could still enhance the convergence rate and the accuracy of the optimal solutions, as it will be discussed in Chapter 9.

Part III

CONTRIBUTIONS

Chapter 5

Curve Reconstruction with Polynomial Bézier Curves

In this chapter we address the problem of curve reconstruction by using a linear combination of global-support basis functions as the fitting function. In particular, in this chapter we consider the case of polynomial Bézier curves, fitted to data points by means of the least-squares approximation technique. In order to solve the resulting nonlinear continuous optimization problem, we apply the bat algorithm described in Section 4.9.

We would like to remark that, in spite of its valuable features for continuous optimization, to the best of our knowledge the bat algorithm was never been applied in the context of data fitting for geometric modeling or computer graphics until the research work in this thesis. In this chapter the bat algorithm is applied to obtain a very accurate fitting curve to a given set of data points. To check the performance of our approach, it has been applied to some simple yet illustrative examples of two-dimensional and three-dimensional shapes.

The structure of this chapter is as follows: some previous work in the subject of data fitting and Bézier curve parameterization is briefly reported in Section 5.2. The problem of data fitting with polynomial Bézier curves is described in Section 5.3. Then, our proposed approach to solve it is described in Section 5.4. The section also discusses the important issue of parameter tuning. Then, three illustrative examples of its application are reported in Section 5.5. The chapter closes in Section 5.6 with the main conclusions of this work.

5.1 Introduction

Obtaining an approximating curve from a given set of data points is a very important and recurrent problem in many applied and industrial domains, such

as computer-aided design and manufacturing (CAD/CAM), virtual reality, medical imaging, computer animation, and many others [6, 7, 12, 29, 38, 86, 134, 187, 199]. In some cases, data can be generated synthetically by direct application of a myriad of CAD/CAM and computer modeling and animation software programs. Very often, however, data points are acquired from real measurements of an existing geometric entity [33, 53, 56, 57]. It appears, for instance, in computer-aided design and manufacturing (CAD/CAM), a field where data points are usually obtained from real measurements of an existing geometric entity, as it typically happens in the construction of car bodies, ship hulls, airplane fuselage, and other free-form objects. This is also a common problem in the shoes industry, in archeology (reconstruction of archeological assets), in medicine (computer tomography (CT), magnetic resonance imaging), computer graphics and animation, virtual and augmented reality, and in many other fields.

Data points in these settings are acquired through many different technologies, such as 3D laser scanning, touch scanners, coordinate measuring machines, CT scanners, or convergent photogrammetry, to mention just a few. A common factor of all these techniques is that a huge number of data (often in the range of hundreds of thousands, and even millions) is usually obtained (see Section 3.1 for further details). Besides, in most cases no geometric or topological information is available beyond the data points. Since dealing with this amount of data becomes impractical, a primary goal in the field is to convert the real data from a physical object into a fully usable digital model, a process commonly called *reverse engineering* (also described in detail in Section 3.1 of this thesis). This is a typical approach in many industrial fields, such as the construction of car bodies, ship hulls, airplane fuselage and other free-form objects [33, 53, 56, 95, 94, 156, 187]. As discussed above, this conversion process will allow significant savings in terms of storage capacity and processing and manufacturing time. Furthermore, the digital models are easier and cheaper to modify than their real counterparts and are usually available anytime and anywhere.

In all those cases, it is desirable to obtain the fitting curve that approximates the set of data points optimally in the sense of least-squares. Indeed, in real-world problems, data points are usually affected by measurement noise, irregular sampling, and other artifacts [7, 152, 156]. Consequently, a good fitting of data should be generally based on approximation schemes rather than on interpolation. In this case, the approximating curve is not required to pass through all input data points, but just near to them, according to some prescribed distance criteria. Best fitting methods make commonly use of least-squares techniques [54, 134, 187].

Several approximating families of functions have been applied to this problem. Among them, the free-form parametric curves such as Bézier, B-

spline and NURBS, are widely applied in many industrial settings due to their great flexibility and the fact that they can represent smooth shapes with only a few parameters [7, 108, 125, 150, 151]. In general, the approximating curves can be classified as global-support and local-support. By global-support curves we mean curves mathematically expressed as a combination of basis functions whose support is the whole domain of the problem. As a consequence, these curves exhibit a global control, in the sense that any modification of the shape of the curve in a particular location is propagated throughout the whole curve. This is in clear contrast to the local-support approaches, which provide local control of the shape of the curve [23, 28] and have become prevalent in CAD/CAM and computer graphics. In this thesis we focus on the global-support approach.

Owing to their remarkable mathematical properties, polynomial curves are a classical choice for data fitting. In particular, free-form parametric curves such as Bézier, B-spline and NURBS, are widely applied in many industrial settings due to their great flexibility and the fact that they can represent well any smooth shape with only a few parameters.

In this thesis we focus particularly on the case of Bézier curves. In spite of the simplicity of these curves, the fitting problem is still far from being trivial; since the curve is parametric, we are confronted with the problem of obtaining a suitable parameterization of the data points [38, 57]. In fact, it is well known that the selection of an appropriate parameterization is essential for a good fitting.

Unfortunately,this is also a very hard problem. It leads to a difficult continuous, multivariate, multimodal, over-determined nonlinear optimization problem (see Section 5.3 for details). In fact, traditional mathematical techniques are not able to solve it in its generality and more powerful approaches have to be used instead. In this chapter we apply a recently proposed nature-inspired meta-heuristic technique called *bat algorithm* (described in detail in Section 4.9 of this thesis) to the curve reconstruction problem with polynomial Bézier curves.

5.2 Previous Work

The problem of data fitting with free-form parametric curves has been the subject of research for many years [7, 29, 110, 154, 157]. First approaches in the field were mostly based on numerical procedures [29, 38, 157, 168]. Some of early approaches relied on given parameterizations, such as uniform or arc-length. This is a reasonable assumption, since these parameterizations have found several important applications in industry (e.g., in computer-numerically-controlled (CNC) operations for computer manufacturing [152]).

However, a prescribed parameterization does not necessarily lead to optimal solutions in data fitting. In fact, it has been shown that better results can be obtained when the data parameters are considered free variables of the problem. Unfortunately, this leads to a difficult nonlinear optimization problem that cannot be fully solved by using standard mathematical techniques. Consequently, there has been a great interest to explore other possible approaches to this problem. Some recent approaches in this line use error bounds [150], curvature-based squared distance minimization [192], or dominant points [151]. In general, they perform well but require some particular constraints (such as high differentiability, closed curves, noiseless data) which are not so commonly met in real-world applications.

On the other hand, some papers published during the last two decades have shown that the application of artificial intelligence techniques can achieve remarkable results regarding this parameterization problem [33, 79, 95, 94, 117]. Most of these methods rely on some kind of neural networks, either standard neural networks [79], Kohonen's SOM (Self-Organizing Maps) nets [6, 86], or the Bernstein Basis Function (BBF) network [117]. In some other cases, the neural network approach is combined with partial differential equations [6] or other approaches [98, 99]. The generalization of these methods to functional networks is also analyzed in [33, 95, 94, 96]. The application of support vector machines to solve the least-squares B-spline curve fitting problem is reported in [108].

Other approaches are based on the application of nature-inspired metaheuristic techniques, which have been intensively applied to solve difficult optimization problems that cannot be tackled through traditional optimization algorithms. For instance, a paper in [52] describes the application of genetic algorithms and functional networks yielding pretty good results for both curves and surfaces. Genetic algorithms have also been applied to this problem in both the discrete version [171, 210] and the continuous version [56, 211]. Other metaheuristic approaches applied to this problem include the use of the popular particle swarm optimization technique [54, 57], artificial immune systems [60, 184], firefly algorithm [59, 61], cuckoo search [68], estimation of distribution algorithms [214], memetic algorithms [67], and hybrid techniques [58, 62, 171].

5.3 Description of the Problem

In this section we provide the reader with the mathematical description of a polynomial Bézier curve. Then, we describe the problem of data fitting with polynomial Bézier curves.

5.3.1 Mathematical background

In this section we assume that the reader is familiar with the main concepts about free-form parametric curves [38]. A *free-form parametric Bézier curve* $\mathbf{C}(t)$ *of degree* n is defined as:

$$\mathbf{C}(t) = \sum_{j=0}^{n} \mathbf{P}_j B_j^n(t) \tag{5.1}$$

where $\mathbf{P}_j$ are vector coefficients (usually referred to as the *control points*), $B_j^n(t)$ are the *Bernstein polynomials of index* j *and degree* n, given by:

$$B_j^n(t) = \binom{n}{j} t^j \ (1-t)^{n-j} \tag{5.2}$$

and t is the *curve parameter*, defined on a finite interval $[0, 1]$. Note that in this paper vectors are denoted in bold. By convention, $0! = 1$.

Different parameterizations $\{t_i\}$ can be considered for the fitting curve based on the set of input data [154]. The simplest parameterization is given by the equally spaced parameters:

$$t_0 = 0 \qquad t_m = 1 \qquad t_j = \frac{j}{m} \qquad j = 1, \ldots, m-1 \tag{5.3}$$

This is called *uniform* parameterization. In general, it is not recommended because it can produce undesirable effects (such as loops) when the data is not evenly spaced.

A more common parameterization is given by the *chord length* method as:

$$t_0 = 0 \qquad t_m = 1 \qquad t_j = t_{j-1} + \frac{|\mathbf{Q}_k - \mathbf{Q}_{k-1}|}{\sum_{k=1}^{m} |\mathbf{Q}_k - \mathbf{Q}_{k-1}|} \qquad j = 1, \ldots, m-1 \tag{5.4}$$

This parameterization is more suitable because it reflects the distribution of data points while also approximates a uniform parameterization.

A variation of the chord length parameterization is given by the *centripetal* method:

$$t_0 = 0 \qquad t_m = 1 \qquad t_j = t_{j-1} + \frac{\sqrt{|\mathbf{Q}_k - \mathbf{Q}_{k-1}|}}{\sqrt{\sum_{k=1}^{m} |\mathbf{Q}_k - \mathbf{Q}_{k-1}|}} \qquad j = 1, \ldots, m-1 \tag{5.5}$$

which performs better when the curves exhibits sharp turns.

In this thesis we are interested to obtain the optimal result. So, instead of relying on any of the previous given parameterizations, we consider the parameterization to be subjected to optimization. Consequently, the data parameters will also be variables to be computed during optimization, as explained in next section.

5.3.2 Data fitting with polynomial Bézier curves

Let us suppose now that we are given a set of data points $\{\mathbf{Q}_i\}_{i=1,...,m}$ in $\mathbb{R}^d$ (usually $d = 2$ or $d = 3$). Our goal is to obtain the free-form parametric Bézier curve $\mathbf{C}(t)$ that fits the data points better in the discrete least-squares sense. To do so, we have to compute the control points $\mathbf{P}_j$ $(j = 0, ..., n)$ of the approximating curve $\mathbf{C}(t)$ by minimizing the least-squares error, E, defined as the sum of squares of the residuals:

$$E = \sum_{i=1}^{m} \left(\mathbf{Q}_i - \sum_{j=0}^{n} \mathbf{P}_j B_j^n(t_i) \right)^2 \tag{5.6}$$

where we need a parameter value t_i to be associated with each data point $\mathbf{Q}_i$, $i = 1, \ldots, m$.

Considering the column vectors $\mathbf{B}_j = (B_j^n(t_1), \ldots, B_j^n(t_m))^T$, $j = 0, \ldots, n$, where $(.)^T$ means transposition, and $\bar{\mathbf{Q}} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_m)$, Eq. (5.6) becomes the following system of equations (called the *normal equation*):

$$\begin{pmatrix} \mathbf{B}_0^T.\mathbf{B}_0 & \ldots & \mathbf{B}_n^T.\mathbf{B}_0 \\ \vdots & \vdots & \vdots \\ \mathbf{B}_0^T.\mathbf{B}_n & \ldots & \mathbf{B}_n^T.\mathbf{B}_n \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \vdots \\ \mathbf{P}_n \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{Q}}.\mathbf{B}_0 \\ \vdots \\ \bar{\mathbf{Q}}.\mathbf{B}_n \end{pmatrix} \tag{5.7}$$

which can be compacted as:

$$\mathbf{M}\,\mathbf{P} = \mathbf{R} \tag{5.8}$$

with $\mathbf{M} = \left[\sum_{j=1}^{m} B_l^n(t_j) B_i^n(t_j) \right]$, $\mathbf{R} = \left[\sum_{j=1}^{m} \mathbf{Q}_j B_l^n(t_j) \right]$ for $i, l = 0, \ldots, n$.

Note that if values are assigned to the t_i, our problem is a classical linear least-squares minimization, with the coefficients $\{\mathbf{P}_i\}$ as unknowns. This problem can readily be solved by standard numerical techniques.

On the contrary, if the values of t_i are treated as unknowns, the problem becomes much more difficult. Indeed, since the blending functions $B_j^n(t)$ are nonlinear in t, the least-squares minimization of the errors is a nonlinear continuous optimization problem. Note also that in many practical cases the number of data points can be extremely large, meaning that we have to deal with a large number of unknowns. In other words, we are also confronted with a high-dimensional problem. It is also a multimodal problem, since

there might be arguably more than one data parameterization vector leading to the optimal solution.

It is clear that, while solving the parameterization problem is the key to obtain an optimal Bézier fitting curve of data, it also leads to a very difficult multimodal, multivariate, continuous, nonlinear optimization problem. In this work, we are interested to solve this general problem. In our approach we make no assumption about the values of data parameters; instead, we compute them at full extent.

5.4 Our approach

5.4.1 Proposed method

Our approach to solve this general problem consists of applying the bat algorithm described above to determine suitable parameter values for the least-squares minimization of functional E according to (5.6).

To this aim, each bat, representing a potential solution, corresponds to a parametric vector $\mathcal{T}_j = (t_1^j, t_2^j, \ldots, t_m^j) \in [0,1]^m$, $(j = 1, \ldots, \mathcal{P})$ and the $\{t_i^j\}_{i=1,\ldots,m}$ are strictly increasing parameters. These parametric vectors are initialized with random values and then sorted. Application of our method yields new positions and velocities of the bats representing the potential solutions of this optimization problem. The process is performed iteratively for a given number of generations, until the convergence of the minimization of the error is eventually achieved.

Regarding the implementation issues, all computations in this chapter have been performed on a 2.6 GHz. Intel Core i7 processor with 8 GB. of RAM running on Windows 8. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2013b.

In our opinion, *Matlab* is a very suitable tool for this task: it is fast and provides reliable, well-tested routines for efficient matrix manipulations. It also contains a bulk of resources regarding the solving of systems of equations. This feature proved to be very valuable in case of ill-conditioned matrices, i.e., with too large (or even infinite) condition number. This is a situation that can actually happen in practice, for instance, when one or several singular values in SVD decomposition are null or very near to zero. Advisable answer to this problem is to set reciprocals of such singular values to zero. *Matlab* command `svd` handles this situation for us.

Similarly, when LU decomposition is used instead, *Matlab* command `lu` returns a suitable matrix factorization regardless the sparsity of the matrix, although different (mostly LAPACK and UMFPACK) routines are invoked in each case. Also, *Matlab* provides us with the command `mldivide` to solve

Symbol	*Meaning*	*Range of Values*	*Chosen Value*
$\mathcal{P}$	population size	$10 - 200$	100
$\mathcal{G}_{max}$	maximum number of generations	$100 - 3000$	2000
$\mathcal{A}^0$	initial loudness	$(0, 2)$	0.5
$\mathcal{A}_{min}$	minimum loudness	$(0, 1)$	0.5
r^0	initial pulse rate	$[0, 1]$	0.5
f_{max}	maximum frequency	$[0, 10]$	2
α	multiplicative factor	$(0, 1)$	0.6
γ	exponential factor	$[0, 1]$	0.4

Table 5.1: Parameters and values used in our method.

the equation $\mathbf{A}.\mathbf{X} = \mathbf{B}$ for both squared and non-squared systems (by using Gaussian elimination with partial pivoting and least-squares techniques, respectively). Depending on the general structure of matrix $\mathbf{A}$, this command applies specialized LAPACK and BLAS routines to get the best possible solution to this system. Besides, *Matlab* provides excellent graphical options and optimized code for input/output interaction and high performance computations.

5.4.2 Parameter tuning

A critical issue when working with metaheuristic techniques is the parameter tuning. It is well-known that the performance of these methods is strongly dependent on the choice of suitable values for their parameters. Moreover, such values are problem-dependent, making it hard to determine good values in advance. Therefore, although there are some papers describing suitable values for some problems, our choice must be necessarily empirical. To this purpose, we carried out numerous computer simulations for different parameter values.

The different parameters used in our method are arranged in rows in Table 5.1. For each parameter, the table shows (in columns) its symbol, meaning, range of values, and the parameter value chosen in this paper.

The most important parameters in the bat algorithm are:

- *population size*: in general, increasing the number of individuals (bats) decreases the number of required iterations, but it also increases the number of function evaluations. Therefore, a trade-off between both situations must be achieved for better performance. In this work, we

	# data points	C^0 *points*	*Self-int. curve*	*2D/3D curve*	*Total DOFs*
Epitrochoid	300	×	✓	2D	332
Piriform	100	✓	×	2D	110
Viviani	200	×	✓	3D	214

Table 5.2: Benchmark used in this chapter along with the main features of each example.

tested populations ranging from 10 to 200 bats and found that while very low values require too many iterations, values in the range $100-200$ perform similarly, so we set this value to 100 individuals.

- *maximum number of iterations*: we tested our method for values of this parameter in the range $100-3000$ and found that the method converged in less than 1500 iterations in all our executions. We finally set this parameter in 2000 iterations, but to prevent wasting computation time without any improvement, also set an additional termination criterion: the method stops if no further improvement of the solution is reached after 20 consecutive iterations, even although the total number of iterations is less than 2000. With this additional criterion, the computation times improved significantly without penalizing the quality of the final solution.

- *initial and minimum loudness and parameter* α: they are set to 0.5, 0.5, and 0.6, respectively. However, from our computer experiments we noticed that our results do not change significantly for values of the initial loudness in the whole range $(0, 2)$, meaning that this parameter is very robust against variations on that interval. Empirically, we found $\alpha = 0.6$ to be a suitable value for this problem.

- *initial pulse rate and parameter* γ: of these two parameters, the initial pulse rate is the most relevant. In fact, parameter γ only affects the very early iterations. We set the initial pulse rate to 0.5, meaning that the selection has an equal probability of change in the long term. We also set $\gamma = 0.4$.

With this choice of parameters, the bat algorithm is run iteratively for the given number of generations. Final positions and velocities of the bats are computed and ranked according to our fitness function E. The position of the global best at the last iteration is taken as the final solution of our minimization problem.

Curve	*Fitting error*
Epitrochoid	**4.065316e-4** (mean E) **7.254377e-5** (best E) **1.164089e-3** (mean RMSE) **4.917444e-4** (best RMSE)
Piriform	3.206772*e*-5 (mean E) 1.378186*e*-6 (best E) 5.662837*e*-4 (mean RMSE) 1.173962*e*-4 (best RMSE)
Viviani	2.192432*e*-4 (mean E) **3.527668e-5** (best E) 1.047003*e*-3 (mean RMSE) **4.199802e-4** (best RMSE)

Table 5.3: Fitting errors for the examples used in this paper (in rows): E error and $RMSE$ for the mean and best results from 50 executions for our bat algorithm-based method (in columns). Best results in our comparative work are highlighted in bold (see our discussion in the main body text for further details).

5.5 Experimental Results

The method described in previous section has been applied to some examples chosen by the authors. Unfortunately, the field lacks a standardized benchmark for further analysis and comparative purposes. We still think, however, that the examples presented here are good enough and suficiently useful to determine the good applicability of our method to this problem. To keep this thesis in manageable size, in this section we describe only three of them, corresponding to 2D and 3D curves.

The benchmark in this paper is shown in Table 5.2, where the three examples selected are arranged in rows. For each example, the table shows (in columns) the number of data points used in our experiments along with some other interesting features: whether or not the curve includes any non-differentiable point (such as cusps) or self-intersections, which usually represent challenging features for data fitting techniques. It also reports the dimensionality of both the curve (2D/3D) and the optimization problem, the latter given by the number of degrees of freedom, *DOF*s for short (i.e., the number of variables to be minimized). These examples have been primarily chosen to reflect the diversity of situations our method can be applied to. First example corresponds to a planar closed curve called epitrochoid, which has several self-intersections; second example shows a curve called piriform,

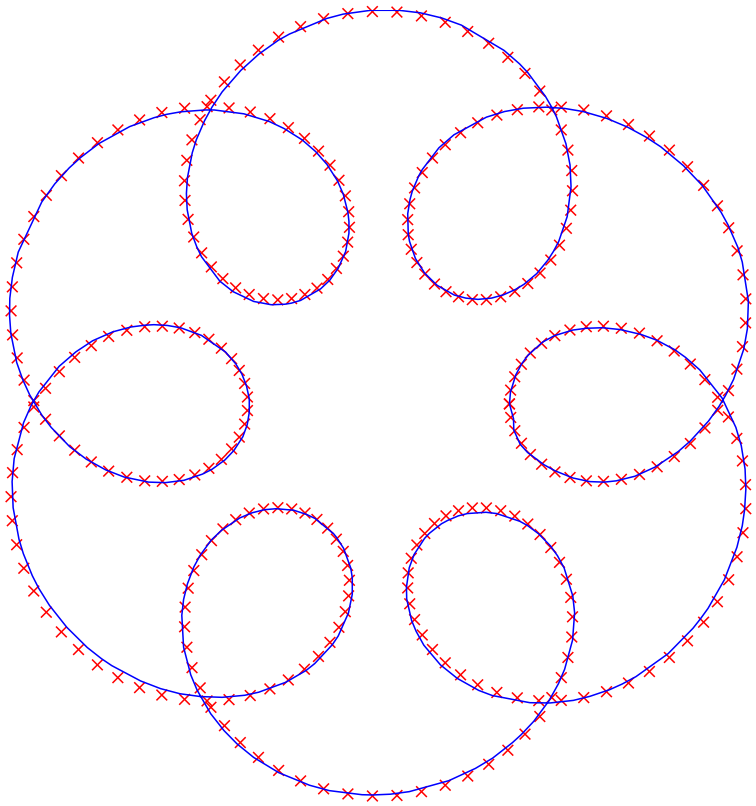

Figure 5.1: Application of our data fitting method to the epitrochoid curve. Original data points are displayed as red $\times$ symbols and the reconstructed curve is displayed as a blue solid line.

a planar closed curve with a cusp; and last example corresponds to a 3D closed curve called the Viviani curve.

The experimental results for the three examples in this paper are shown in Figures 5.1, 5.2, and 5.3, respectively. In all cases, the original data points are displayed as red $\times$ symbols while the reconstructed curve is displayed as a blue solid line.

The reader will notice the good visual fitting between the original data points and the reconstructed curve for the three examples. They show that our method is able to reconstruct the underlying shape of data points even in presence of challenging features, such as several self-intersections or cusps.

This good visual behavior is confirmed by our numerical results, reported in Table 5.3. Once again, the three examples are arranged in rows in first

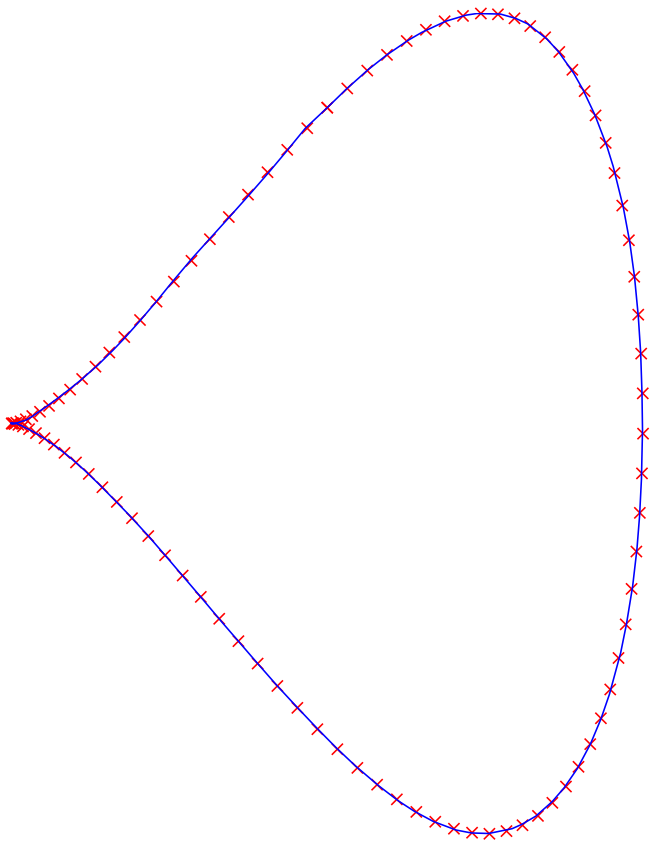

Figure 5.2: Application of our data fitting method to the piriform curve. Original data points are displayed as red $\times$ symbols and the reconstructed curve is displayed as a blue solid line.

column. For each example, the second column of the table reports respectively:

- the average and best error of the functional E according to Eq. (5.6);
- the average and best value of the root-mean square error, given by:

$$RMSE = \sqrt{\frac{E}{m}}.$$

The average error has been obtained as the mean value from 50 independent executions of the algorithm, while the best value corresponds to the

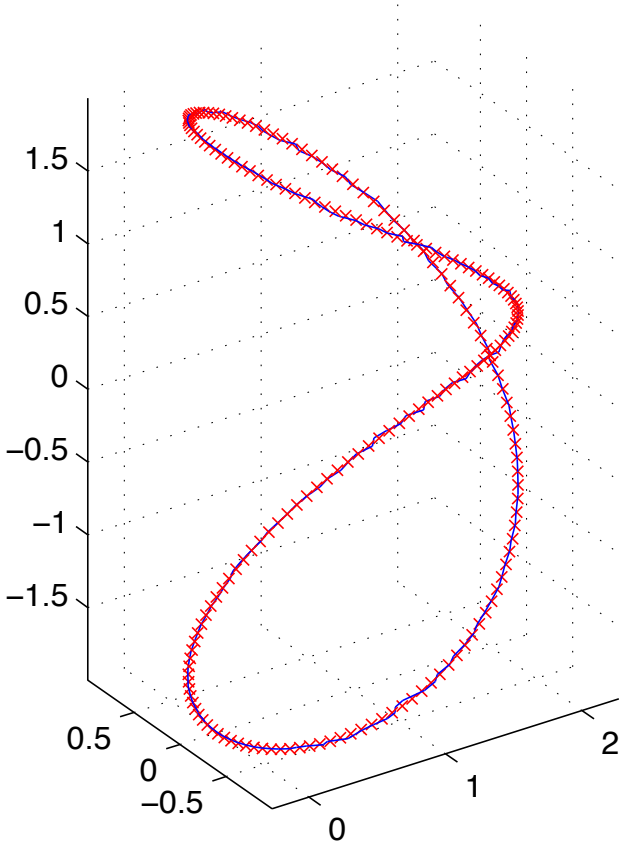

Figure 5.3: Application of our data fitting method to the 3D Viviani curve. Original data points are displayed as red $\times$ symbols and the reconstructed curve is displayed as a blue solid line.

results of the best execution from the 50 runs. As the reader can see, average errors are of order $10^{-4} \sim 10^{-5}$ (mean) to $10^{-5} \sim 10^{-6}$ (best) for the E error, with RMSE of order $10^{-3} \sim 10^{-4}$ (mean) to 10^{-4} (best) for the three examples in our benchmark. From the data in Table 5.3 we can conclude that the method performs very well, being able to replicate the original shape with high accuracy for all instances in our benchmark.

5.5.1 Comparison with other approaches

We have also compared our method with two other alternative parameterizations, obtained through the arc-length and the firefly algorithm. They are

Curve	*Fitting error (arc-length)*	*Fitting error (firefly algorithm)*
Epitrochoid	5.342768e-3 (mean E) 9.017334e-4 (best E) 4.220097e-3 (mean RMSE) 1.733371e-3 (best RMSE)	4.526912e-4 (mean E) 1.034348e-4 (best E) 1.228401e-3 (mean RMSE) 5.871819e-4 (best RMSE)
Piriform	5.532432e-4 (mean E) 5.113864e-5 (best E) 2.352112e-3 (mean RMSE) 7.151128e-3 (best RMSE)	**3.187544e-5** (mean E) **1.283742e-6** (best E) **5.645833e-4** (mean RMSE) **1.113302e-4** (best RMSE)
Viviani	8.174659e-4 (mean E) 9.364587e-5 (best E) 2.021714e-3 (mean RMSE) 6.842728e-4 (best RMSE)	**2.186931e-4** (mean E) 4.624328e-5 (best E) **1.045689e-3** (mean RMSE) 4.808496e-4 (best RMSE)

Table 5.4: Fitting errors for the examples used in this paper (in rows): E error and $RMSE$ for the mean and best results from 50 executions for the arc-length, firefly algorithm, and our method (in columns). Best results in our comparative work are highlighted in bold (see our discussion in the main body text for further details).

also reported in the last two columns of Table 5.4. Best results from the comparison between our results with the bat algorithm method (reported in Table 5.3) and the results with the arc-length parameterization and that obtained with the firefly algorithm (reported in Table 5.4) are highlighted in bold in both tables for easier identification.

As the reader can see from both tables, our method outperforms the arc-length parameterization significantly for all instances in our benchmark. On the other hand, our results are comparable with those of the firefly algorithm, which has already been reported to be an excellent method for the curve parameterization issue [59, 61, 64, 67]. These results confirm the good performance of our approach and its applicability to data fitting with polynomial Bézier curves.

5.6 Main Conclusions

This chapter described a new method for data fitting with polynomial Bézier curves. This problem arises in a number of areas, such as computer-aided design and manufacturing (CAD/CAM), virtual reality, medical imaging, computer graphics, computer animation, and many others. Unfortunately, it is also a highly nonlinear, over-determined, multivariate continuous opti-

mization problem. As a consequence, classical mathematical methods cannot solve it in its generality.

Our approach is based on a powerful nature-inspired optimization method called bat algorithm, which has been recently introduced to solve hard continuous optimization problems. In spite of these remarkable features for optimization, the bat algorithm has never been applied in the context of data fitting for geometric modeling or computer graphics.

To analyze the performance of this approach, it has been applied to some simple yet illustrative examples of Bézier curves with satisfactory results. Our experimental results show that our method based on the bat algorithm is very competitive to obtain a good parameterization for the fitting curve to the cloud of data points. In particular, the method outperforms the classical arc-length parameterization for all instances in our benchmark.

On the other hand, the results with the bat algorithm method are comparable with those with the firefly algorithm, which has already been reported to be an excellent method for the curve parameterization. These facts validate the applicability and good performance of our bat algorithm approach.

Chapter 6

Curve Reconstruction with Rational Bézier Curves

In previous chapter we applied the bat algorithm to obtain an optimal parameterization of the polynomial Bézier curve that best fits a given set of data points. In this chapter we are interested to extend this methodology to the case of rational Bézier curves. As it will be discussed in this chapter, this extension is not natural in the sense that the rational case is much more complicated than the polynomial one. However, the method can indeed be extended with proper care. Furthermore, we can still obtain pretty good results.

The structure of this chapter is as follows: some previous work in the subject of data fitting with rational parametric curves is briefly reported in Section 6.2. The problem of curve approximation with rational Bézier curves is discussed in Section 6.3. Our proposed approach to solve this problem is described in Section 6.4. The section also discusses the important issue of parameter tuning. Then, three illustrative examples of its application are reported in Section 6.5. The paper closes with the main conclusions of this contribution in Section 6.6.

6.1 Introduction

In previous chapter we applied the bat algorithm to obtain an optimal parameterization of the polynomial Bézier curve that best fits a given set of data points. Some previous papers also addressed this problem by using Bézier curves [68, 131]. In general, they obtained good results for a number of shapes, as was presented, for instance, in previous chapter for bat algorithm and in [59, 61, 67, 68, 131] for the firefly algorithm, the cuckoo search and for the simulated annealing. However, this polynomial approach is still limited, as it cannot adequately describe some particular shapes such as the

conics (e.g., circles, ellipses, hyperbolas). As a consequence, there is still a need for more powerful blending functions.

An interesting extension in this regard is given by the rational basis functions, which are mathematically described as the quotient of two polynomials. A remarkable advantage of this rational scheme is that the conics can be canonically described as rational functions. In this chapter, we take advantage of this valuable feature to solve the curve approximation problem by using rational Bézier curves.

Unfortunately, this rational approach becomes more difficult than the polynomial one, since new parameters (the weights) are now introduced into the problem. Consequently, we are confronted with the challenge of obtaining optimal values for many parameters, namely, data parameters, poles, and weights that are qualitatively different in nature. To make things worse, all these sets of parameters are not independent but strongly related to each other in an intricate and highly nonlinear way. As a consequence, we are facing an extremely difficult over-determined high-dimensional continuous nonlinear optimization problem.

Our approach to solve this issue consists of extending the bat algorithm method developed in previous chapter and designed for polynomial Bézier curves to the rational case. This issue will be discussed throughout this chapter.

6.2 Previous Work

As mentioned in previous chapter, there are some approaches to data fitting with free-form parametric shapes based on the application of nature-inspired metaheuristic techniques. Some examples include genetic algorithms [56, 171, 211], particle swarm optimization technique [54, 57], artificial immune systems [60, 69], firefly algorithm [61, 62], cuckoo search [68], simulated annealing [131], estimation of distribution algorithms [214], memetic algorithms [67], and hybrid techniques [60, 171].

In general, all these methods yield good results for shapes that can be properly described in polynomial terms, but more complicated shapes are still elusive. That is the reason behind our proposal to use rational functions as an extension of the polynomial case. In particular, none of these approaches is directly applicable to rational curves. This fact is not accidental, but a clear indication of the difficulty to deal with rational shapes for curve reconstruction. It is also a clear indication about the originality of the proposal in this chapter.

6.3 Description of the Problem

In this section we provide the reader with the mathematical description of a rational Bézier curve. Then, we describe the problem of data fitting with rational Bézier curves.

6.3.1 Mathematical background

In Section 5.3.1 we introduced the definition of a polynomial Bézier curve. Now, we proceed in a similar way with the rational case. Mathematically, a *free-form rational Bézier curve* $\mathbf{\Phi}(\tau)$ *of degree* η is defined as:

$$\mathbf{\Phi}(\tau) = \frac{\sum_{j=0}^{\eta} \omega_j \mathbf{\Lambda}_j \phi_j^{\eta}(\tau)}{\sum_{j=0}^{\eta} \omega_j \phi_j^{\eta}(\tau)} \tag{6.1}$$

where $\mathbf{\Lambda}_j$ are vector coefficients called the *poles*, ω_j are their scalar weights, $\phi_j^{\eta}(\tau)$ are the *Bernstein polynomials of index* j *and degree* η, given by:

$$\phi_j^{\eta}(\tau) = \binom{\eta}{j} \tau^j \ (1-\tau)^{\eta - j}$$

and τ is the *curve parameter*, defined on the finite interval $[0, 1]$. By convention, $0! = 1$.

Note that considering the rational Bernstein basis function, mathematically expressed as:

$$\varphi_j^{\eta}(\tau) = \frac{\omega_j \phi_j^{\eta}(\tau)}{\sum_{k=0}^{\eta} \omega_k \phi_k^{\eta}(\tau)} \tag{6.2}$$

Eq. (6.1) becomes:

$$\mathbf{\Phi}(\tau) = \sum_{j=0}^{\eta} \mathbf{\Lambda}_j \varphi_j^{\eta}(\tau) \tag{6.3}$$

In other words, the rational curve can also be expressed as a linear combination of basis functions, but such basis functions are at their turn rational functions.

6.3.2 Data fitting with polynomial Bézier curves

Suppose now that we are given a set of data points $\{\mathbf{\Delta}_i\}_{i=1,\ldots,\kappa}$ in $\mathbb{R}^\nu$ (usually $\nu = 2$ or $\nu = 3$). Our goal is to obtain the rational Bézier curve $\mathbf{\Phi}(\tau)$ performing discrete approximation of the data points $\{\mathbf{\Delta}_i\}_i$.

To that purpose, we have to compute all parameters (i.e. poles $\mathbf{\Lambda}_j$, weights ω_j, and parameters τ_i associated with data points $\mathbf{\Delta}_i$, for $i = 1, \ldots, \kappa$, $j = 0, \ldots, \eta$) of the approximating curve $\mathbf{\Phi}(\tau)$ by minimizing the least-squares error, Υ, defined as the sum of squares of the residuals:

$$\Upsilon = \underset{\substack{\{\tau_i\}_i \\ \{\mathbf{\Lambda}_j\}_j \\ \{\omega_j\}_j}}{\text{minimize}} \left[\sum_{i=1}^{\kappa} \left(\mathbf{\Delta}_i - \frac{\sum_{j=0}^{\eta} \omega_j \mathbf{\Lambda}_j \phi_j^\eta(\tau_i)}{\sum_{j=0}^{\eta} \omega_j \phi_j^\eta(\tau_i)} \right)^2 \right]. \tag{6.4}$$

Considering the rational Bernstein basis functions in Eq. (6.2), Eq. (6.4) becomes:

$$\Upsilon = \underset{\substack{\{\tau_i\}_i \\ \{\mathbf{\Lambda}_j\}_j \\ \{\omega_j\}_j}}{\text{minimize}} \left[\sum_{i=1}^{\kappa} \left(\mathbf{\Delta}_i - \sum_{j=0}^{\eta} \mathbf{\Lambda}_j \varphi_j^\eta(\tau) \right)^2 \right], \tag{6.5}$$

which can be rewritten in matrix form as:

$$\mathbf{\Omega}.\mathbf{\Lambda} = \mathbf{\Xi} \tag{6.6}$$

called the *normal equation*, where:

$$\mathbf{\Omega} = [\Omega_{i,j}] = \left[\left(\sum_{k=1}^{\kappa} \varphi_i^\eta(\tau_k) \varphi_j^\eta(\tau_k) \right)_{i,j} \right]$$

$$\mathbf{\Xi} = [\Xi_j] = \left[\left(\sum_{k=1}^{\kappa} \mathbf{\Delta}_k \varphi_j^\eta(\tau_k) \right)_j \right]$$

$$\mathbf{\Lambda} = (\mathbf{\Lambda}_0, \ldots, \mathbf{\Lambda}_\eta)^T$$

for $i, j = 0, \ldots, \eta$, and $(.)^T$ means the transposition of a vector or a matrix.

In general, $\kappa >> \eta$ meaning that the system (6.6) is over-determined. If values are assigned to the τ_i, our problem can be solved as a classical linear least-squares minimization, with the coefficients $\{\mathbf{\Lambda}_i\}_{i=0,\ldots,\eta}$ as unknowns. This problem can readily be solved by standard numerical techniques.

On the contrary, if the values of τ_i are treated as unknowns, the problem becomes much more difficult. Indeed, since the polynomial blending

functions $\phi_j^\eta(\tau)$ are nonlinear in τ and so are the rational blending functions $\varphi_j^\eta(\tau)$, the least-squares minimization of the errors is a nonlinear continuous optimization problem.

Note also that in many practical cases the number of data points can be extremely large, meaning that we have to deal with a large number of unknowns. In other words, we are also confronted with a high-dimensional problem. It is also a multimodal problem, since there might be arguably more than one set of parameter values leading to the optimal solution.

In conclusion, it is clear that the interplay among all sets of unknowns (data parameters, poles, and weights) leads to a very difficult over-determined, multimodal, multivariate, continuous, nonlinear optimization problem. In this chapter, we are interested to solve this general problem. This means that we make no assumption about the values of the free parameters; instead, we compute them at full extent.

6.4 Our approach

6.4.1 Proposed method

Our approach to solve this general problem consists of applying the bat algorithm described in Section 4.9 of this thesis to determine suitable parameter values for the least-squares minimization of functional Υ according to (6.4). To this aim, each bat, representing a potential solution, corresponds to a parametric vector $\mathcal{S}_j$ of length $\kappa + \eta + 1$ given by $\mathcal{S}_j = [\mathcal{T}_j; \mathcal{W}_j]$, where $\mathcal{T}_j = (\tau_1^j, \tau_2^j, \ldots, \tau_\kappa^j) \in [0,1]^\kappa$ with the $\{\tau_i^j\}_{i=1,\ldots,\kappa}$ strictly increasing parameters, and $\mathcal{W}_j = (w_0^j, \ldots, w_\eta^j)$ for $j = 1, \ldots, \mathcal{P}$. These parametric vectors are initialized with random values according to the uniform distribution on the unit interval. Once initialized, the coordinates of $\mathcal{T}_j$ are sorted to reproduce the ordered structure of the data points.

Regarding $\mathcal{W}_j$, in this paper we consider values for the weights within the range $(0, 100)$, as values larger than 100 do not modify the shape of the curve noticeably. Application of the bat algorithm described above yields new positions and velocities of the bats representing the potential solutions of our optimization problem.

As shown in the previous section, the introduction of the rational Bernstein basis functions allows us to rewrite the expression of the rational Bézier curve as a linear combination of basis functions, leading to the normal equations in Eq. (6.6). This implies that the numerical routines required to solve the problem in Eq. (6.6) are basically the same used to solve Eq. (5.8) for the polynomial case. Since these numerical routines have already been described in Section 5.4.1, they will be omitted here to avoid unnecessary duplication of material.

Symbol	*Meaning*	*Range of Values*	*Chosen Value*
$\mathcal{P}$	population size	$10 - 200$	100
g	maximum number of generations	$100 - 3000$	2000
$\mathcal{A}^0$	initial loudness	$(0, 2)$	0.5
$\mathcal{A}_{min}$	minimum loudness	$[0, 1]$	0
r^0	initial pulse rate	$[0, 1]$	0.5
f_{max}	maximum frequency	$[0, 10]$	2
α	multiplicative factor	$(0, 1)$	0.6
γ	exponential factor	$[0, 1]$	0.4

Table 6.1: Parameters and values used in our method.

6.4.2 Parameter tuning

The issue of parameter tuning for the bat algorithm was already discussed in Section 5.4.2, where the most relevant parameters of the method were explained in detail. Therefore, there is no need to repeat the same discussion here. However, we remark that the most suitable choice for values for such parameters was derived for the polynomial case, meaning that it might not be directly applicable to the rational case. Once again, such values are problem-dependent, and therefore our choice must be necessarily empirical.

To this purpose, we carried out numerous computer simulations for different parameter values. Similar to the polynomial case, the different parameters used in our method are arranged in rows in Table 6.1. For each parameter, the table shows (in columns) its symbol, meaning, range of values, and the parameter value chosen in this paper.

In general, the values obtained for the polynomial case are still valid for the rational one with little variations. The most significant one refers to the minimum loudness parameter, which is set to 0 in this case, with the effect of accelerating the convergence by allowing lower values than the initial choice $\mathcal{A}^0$. Anyway, the reduction of computation time is not very dramatic actually, at least in our computational experiments described in next section.

With this choice of parameters, the bat algorithm is run iteratively for a given number of generations or until the convergence of the minimization of the error is eventually achieved. Final positions and velocities of the bats are computed and ranked according to our fitness function Υ. The position of the global best at the final iteration is taken as the final solution of our minimization problem.

Benchmark example	*# data points*	*Open/closed curve*	*Non-differentiable points*	*2D/3D curve*	*degrees of freedom*
Hypocycloid	200	closed	✓	2D	124
Spiral	100	open	×	2D	136
Helix	100	open	×	3D	242

Table 6.2: Benchmark used in this paper along with the main features of each example.

Curve	$Av_\mu(mean)$	$Av_\mu(best)$	Υ (mean)	Υ (best)	*RMSE* (mean)	*RMSE* (best)
Hypocycloid	x : 6.390788*e*-4 y : 7.138176*e*-4	x : 3.294507*e*-5 y : 4.051993*e*-5	3.475935*e*-4	3.075698*e*-5	1.318322*e*-3	3.921542*e*-4
Spiral	x : 6.998090*e*-3 y : 7.189277*e*-3	x : 2.532214*e*-4 y : 2.985567*e*-4	8.589174*e*-4	1.867033*e*-5	2.930729*e*-3	4.320918*e*-4
Helix	x : 1.050018*e*-2 y : 1.102567*e*-2 z : 9.711053*e*-4	x : 3.277831*e*-3 y : 2.689946*e*-3 y : 3.091722*e*-4	3.261139*e*-4	1.860439*e*-5	1.217693*e*-3	4.313281*e*-4

Table 6.3: Fitting errors for the examples used in this paper (arranged in rows): coordinate error, Υ error and $RMSE$ for the mean and best results from 50 executions (in columns).

6.5 Experimental Results

The method described in previous section has been applied to several examples chosen by the authors. In this chapter we describe only three of them, corresponding to 2D and 3D curves as described in Table 6.2. In this table, the three examples are reported in rows. For each example, the table shows (in columns) the number of data points used in our experiments along with some other interesting features: if the curve is open or closed, planar or 3D, or whether it includes any non-differentiable point (such as cusps or discontinuities). It also reports the number of degrees of freedom (DOFs) of the optimization problem (i.e. the number of variables to be minimized).

These examples have been primarily chosen to reflect the diversity of situations our method can be applied to. First example corresponds to a planar closed curve called hypocycloid, which has several cusps; second example shows a spiral, a planar open curve with a smooth yet challenging shape; and last example corresponds to a helix, a 3D open curve with several DOFs.

The fitting results we obtained for the three examples in our benchmark are displayed in Figures 6.1, 6.2, and 6.3, respectively. In all cases, the original data points are displayed as red empty circles while the reconstructed

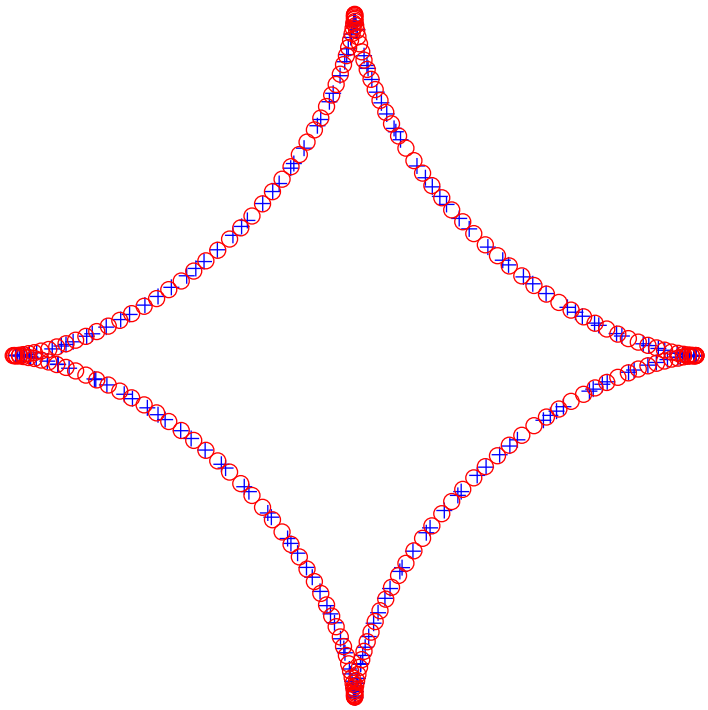

Figure 6.1: Examples of application of the bat algorithm for data approximation with rational Bézier curves to the hypocycloid curve. Original points are displayed as red empty circles and the reconstructed points as blue + symbols.

points are displayed as blue + symbols. Note the very good matching between both sets of points for the three examples.

This good visual behavior is confirmed by our numerical results, reported in Table 6.3. The three examples are arranged in rows. For each example, the table reports (in columns):

- the average and the best error of the coordinates, according to the equation: $Av_\mu = \sum_{i=1}^{\kappa} |\delta_j^\mu - \bar{\delta}_j^\mu|$, where δ_j^μ and $\bar{\delta}_j^\mu$ represent respectively the μ coordinate of the input data and reconstructed data, with $\mu = x, y$ for 2D curves ($\mu = x, y, z$ for 3D curves);

- the average and best error of the functional Υ according to Eq. (6.5);

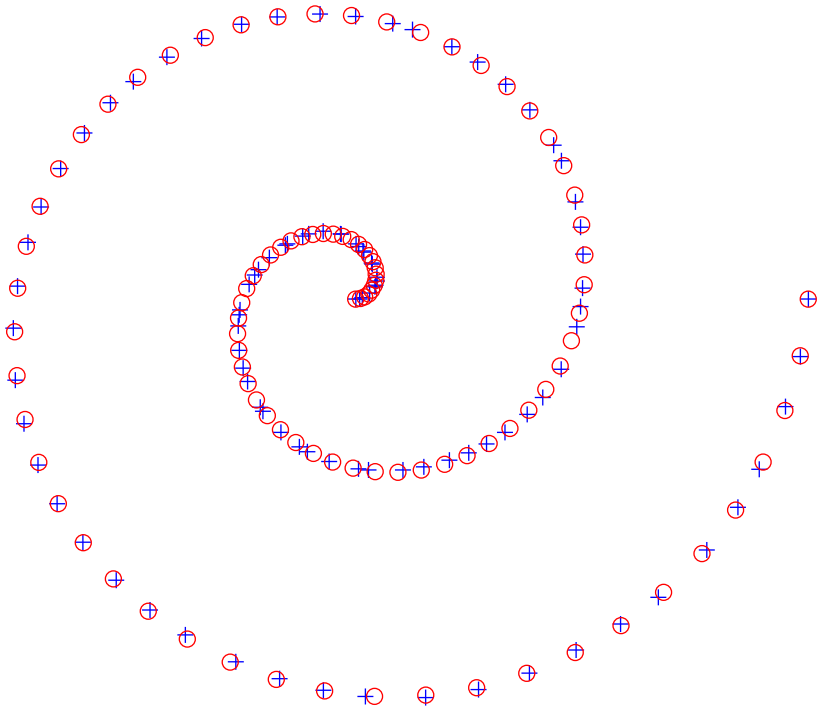

Figure 6.2: Examples of application of the bat algorithm for data approximation with rational Bézier curves to the spiral curve. Original points are displayed as red empty circles and the reconstructed points as blue + symbols.

- the average and best error of the root-mean square error, given by:

$$RMSE = \sqrt{\frac{\Upsilon}{\kappa}}.$$

The average error has been obtained as the mean value from 50 independent executions of the algorithm, while the best value corresponds to the results of the best execution from the 50 runs. As the reader can see, average errors are of order 10^{-4} (average) to 10^{-5} (best) for the Υ error, with RMSE of order 10^{-4} (average) to 10^{-5} (best) for the three examples in our benchmark. Coordinate errors show more noticeable variations for the third example due to the scale factor of the vertical component.

From the data in Table 6.3 we can conclude that the method performs very well, being able to replicate the original shape with high accuracy for all instances in our benchmark.

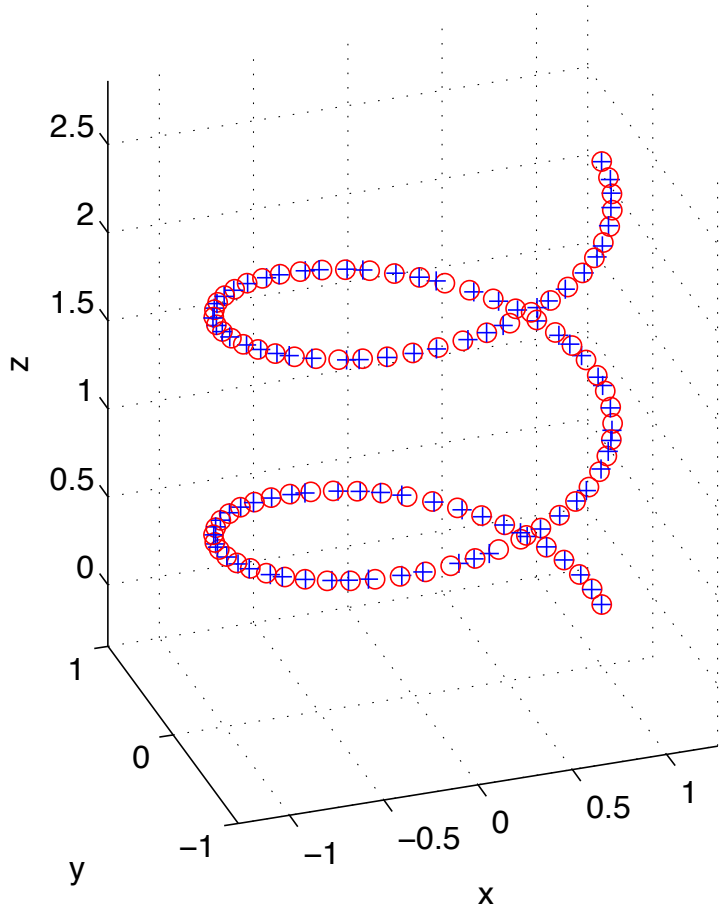

Figure 6.3: Examples of application of the bat algorithm for data approximation with rational Bézier curves to the helical curve. Original points are displayed as red empty circles and the reconstructed points as blue + symbols.

6.6 Main Conclusions

This chapter introduced a new method for discrete approximation of data points with rational Bézier curves. Given a set of data points, the method computes all relevant parameters (poles, weights, and data parameters) of the rational Bézier fitting curve as the solution of a challenging over-determined nonlinear multimodal multivariate continuous optimization problem. The method is based on the bat algorithm and is an extension of the method we developed for polynomial Bézier curves in the previous chapter. However, this extension is not trivial at all. There are several reasons to explain this: on one

hand, the problem is more complicated because new parameters (the weights) are now introduced into the problem. On the other hand, all these sets of parameters (data parameters, control points, weights) are quite different between them and are not independent but strongly related to each other in an intricate and highly nonlinear way. As a consequence, we are facing an extremely difficult over-determined high-dimensional continuous nonlinear optimization problem.

In spite of these difficulties, the bat algorithm performs well in obtaining suitable values for all these parameters. To check the performance of our approach, it has been applied to some illustrative examples of 2D and 3D curves. Our results show that the method performs pretty well, being able to yield a satisfactory approximating curve with a high degree of accuracy. This approach generalizes the previous method for data fitting based on polynomial basis functions to rational blending functions, thus expanding the potential range of applications to include more difficult shapes.

Chapter 7

Surface Reconstruction with Polynomial Bézier Surfaces

The previous two chapters in this thesis were devoted to solve the problem of curve reconstruction. In this chapter and the next one we shift our focus to the problem of surface reconstruction. In some ways, they are very similar problems. Both they gravitate around the idea of fitting a given cloud of points by using a mathematical entity (be a curve or a surface) that more or less replicate the inherent shape of the cloud of data points.

However, mathematically they are not exactly the same problem, and not only because the surface is by definition a 3D entity, while curves can be 2D as well. It is not just a matter of dimension. It is neither because the surfaces require more parameters and typically more dense clouds to be accurately outlined. Although it can be argued that the free-form parametric surfaces are somehow an extension of the free-form parametric curves (which, by the way, is true – at least, to a certain extent – for the polynomial case, but not for the rational case anymore), the experience shows that they are different problems at their own. Hopefully, this observation will become much clear throughout the following two chapters, devoted to surface reconstruction with polynomial Bézier surfaces and rational Bézier surfaces, respectively.

This chapter is aimed at solving the surface reconstruction problem by using polynomial Bézier surfaces. And once again, this problem becomes much more complicated if no parameterization *a priori* is assumed. In fact, the case where the parameterization is given is mathematically of little interest, since it can be solved by standard numerical tools such as linear least-squares and other suitable routines. However, as soon as we assume no parameterization, the problem becomes highly nonlinear, over-determined and multivariate. It is also multimodal for many instances, as it will be discussed later on. Therefore, no analytical solution can be expected; even worse, the standard mathematical optimization techniques are not powerful enough to solve this

problem. Once again, a different approach is needed. And the metaheuristics described in Chapter 4 (particularly, the bat algorithm) come again to our rescue.

Our proposal in this chapter applies the bat algorithm described in Section 4.9 to solve the data reconstruction problem with polynomial Bézier surfaces. To analyze the performance of our method, it will applied to three illustrative examples. The computational experiments show that the method performs well and can reconstruct an approximating surface with high accuracy.

This chapter is organized as follows: Section 7.1 introduces the problem to be solved. Then, Section 7.2 briefly reports some previous work in the subject. The fundamentals about parametric surfaces as well as the data fitting problem with polynomial Bézier surfaces are discussed in Section 7.3. Then, Section 7.4 describes our approach to solve this problem. Some computational results of our method as applied to three illustrative examples are described in Section 7.5. Finally, Section 7.6 reports the main conclusions of this work.

7.1 Introduction

As discussed in Sections 3.1 and 3.3 of this thesis, obtaining a surface that fits a given cloud of data points (a research topic commonly known as *surface reconstruction*) is a classical problem in several scientific and technological domains such as computer-aided design (CAD), computer aided manufacturing (CAM), virtual reality, computer graphics, computer animation, medical imaging, scientific visualization, and many others [12, 29, 45, 86, 154]. In real-world applications, data points arise from measurements of an already constructed geometric workpiece, as it often occurs in the automotive industry (car bodies), aerospatial (airplane fuselage), ship hull building, shoes industry, medical imaging (computer tomography, magnetic resonance imaging), and so on [33, 53, 57, 95, 94]. Usually, data points are captured by devices such as 3D laser scanning, coordinate measuring machines, computer tomography scanners, magnetic resonance, and so on. The amount of data captured through these technologies can be very huge, typically ranging from hundreds of thousands to billions of data points (see for instance, our discussion about the *Michelangelo* project in Section 3.3.2, and the illustrative examples in Figures 3.2 and 3.3).

Dealing with these very large sets of data points is impractical and computationally expensive. Therefore, a major task in this context is to transform the data from the real object into a digital model, the basis of the process known as *reverse engineering*. The primary goal of this process is to get a mathematical surface approximating the data accurately, thus replacing the

massive cloud of points by a geometric entity with only a few parameters. Advantages of this process have already been discussed in enough detail in Chapter 3.

In real-world settings, measurement data are affected by irregular sampling, measurement noise, or other artifacts [152]. As a result, approximation is chosen over interpolation, generally expressed as a least-squares fitting problem [54]. This is the approach followed in this chapter. In this work we will focus particularly on the case of irregularly sampled data points, which is by far the most common (and most difficult) case in real-world applications.

Because of their remarkable mathematical properties, polynomial functions (in particular, free-form parametric functions) are a classical choice for surface reconstruction. In this chapter we focus on the particular case of polynomial Bézier surfaces, a kind of free-form splines very popular in fields such as computer graphics and geometric modeling.

In this case, a major problem is to determine an appropriate data parameterization [154]. In fact, the selection of a suitable parameterization is very important for a good fitting. At the same time, it is a very hard problem. As it is commonly known, it requires to solve a very difficult overdetermined continuous nonlinear optimization problem. In addition, it is multivariate: a huge number of data points implies a lot of unknown variables. Finally, owing to the potential existence of several (global or local) optima of the objective function, it is usually a multimodal problem as well. As a result, traditional mathematical optimization techniques fail to solve it. Clearly, more powerful approaches are needed instead.

7.2 Previous Work

The problem of data fitting through free-form parametric surfaces has been an important issue of research for many years [29, 154, 157]. One of the most important problems regarding this issue is the surface parameterization. Two very popular choices in this regard are the uniform and the arc-length parameterizations. The former is widely used because of its simplicity but it is limited to smooth surfaces and regular sampling.

In many practical situations, it is convenient to apply the arc-length parameterization. In this case, constant steps on the parametric domain translate into constant distances along an arc-length parameterized curve on the surface. That is, constant parameter intervals on the domain induce regular spacing of points of the curve on the surface. This property has been traditionally applied in metrology for design and manufacturing, to collect measurement data from industrial parts of the designed and manufactured products. Other example arises in CNC (computer numerically-controlled) milling operations for manufacturing, where the tool-path of the manufac-

turing machine has to be parametrized in such a way that the cutter neither slows down nor speeds up along its path [152]. The only way to guarantee this property is using the arc-length parameterization for the tool-path.

However, a prescribed parameterization does not necessarily lead to optimal solutions in data fitting. In fact, it has been shown that better results can be obtained when the data parameters are considered free unknowns of the problem. Unfortunately, this poses a difficult nonlinear optimization problem that cannot be fully solved by using standard mathematical techniques. Consequently, there has been a great interest to explore other possible approaches to this problem.

Some artificial intelligence techniques have achieved relevant results on this parameterization problem [33, 79, 95, 94, 117]. Many of these methods are based on different neural nets, such as classical neural nets [79], Kohonen's self-organizing maps (SOM) nets [86], Bernstein basis function nets [117], or the like. Neural nets are also replaced by functional networks (which extend the neural networks allowing the scalar weights to be functions) [33, 95, 94, 96] or other approaches [99]. The application of support vector machines (SVM) to solve the least-squares curve fitting problem is reported in [108].

Other approaches rely on popular nature-inspired optimization techniques. For instance, genetic algorithms have also been applied to this problem [57, 211]. Other metaheuristic approaches applied to this problem include the popular particle swarm optimization technique [54, 56], firefly algorithm [59, 61], cuckoo search [68], memetic algorithms [67, 100], AIS [60], simulated annealing [131], hybrid techniques [58, 62] and so on.

7.3 Description of the Problem

In this section we provide the reader with the mathematical description of a polynomial Bézier surface. Then, we describe the problem of data fitting with polynomial Bézier surfaces.

7.3.1 Mathematical background

A *free-form polynomial parametric surface* is defined as (see [38, 154] for further details):

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{ij}\phi_i(u)\varphi_j(v) \tag{7.1}$$

where $\{\mathbf{P}_{i,j}\}_{i=0,\ldots,m;j=0,\ldots,n}$ are vector coefficients in $\mathbb{R}^3$ called the *control points*, $\{\phi_i(u)\}_i$ and $\{\varphi_j(v)\}_j$ are two sets of *basis functions*, and (u,v) are the surface parameters, usually defined on a bounded rectangular domain $[\alpha_u,\beta_u]\times[\alpha_v,\beta_v]\subset\mathbb{R}^2$. Note that vectors are denoted in bold.

In this chapter we will focus on the particular case of free-form polynomial Bézier surfaces. In this case, Eq. (7.1) becomes:

$$\mathbf{B}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{i,j}\Psi_i^m(u)\Psi_j^n(v) \tag{7.2}$$

where the blending functions $\Psi_k^d(\omega)$ are given by:

$$\Psi_k^d(\omega) = \frac{d!}{k!\,(d-k)!}\,\omega^k\,(1-\omega)^{d-k}$$

and the surface parameters (u, v) are defined on the unit square $[0,1]\times[0,1]$. Note that, by convention, $0! = 1$.

7.3.2 Data fitting with polynomial Bézier surfaces

Suppose now that we are given a set of 3D data points $\{\mathbf{Q}_{k,l}\}_{k=1,\ldots,p;l=1,\ldots,q}$. Our goal is to obtain the best fitting surface $\mathbf{B}(u,v)$ in the discrete least-squares sense. To do so, we have to minimize the least-squares error, E:

$$E = \sum_{k=1}^{p}\sum_{l=1}^{q}\left(\mathbf{Q}_{k,l} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_k)\Psi_j^n(v_l)\right)^2 \tag{7.3}$$

Minimization of (7.3) leads to the system of equations:

$$< \mathbf{Q} >=< \mathbf{P} > .\boldsymbol{\Xi} \tag{7.4}$$

where $< \mathbf{Q} >$, $< \mathbf{P} >$ correspond to the vectorization of the sets $\{\mathbf{Q}_{k,l}\}$ and $\{\mathbf{P}_{i,j}\}$, respectively, and $\boldsymbol{\Xi}$ is a matrix given by:

$$\boldsymbol{\Xi}_{i,j} = \boldsymbol{\Psi}^n(v_j) \odot \Psi_0^m(\mathbf{u})$$

with

$$\boldsymbol{\Psi}^d(\omega_k) = (\Psi_0^d(\omega_k), \ldots, \Psi_D^d(\omega_k))$$

$$\Psi_k^d(\boldsymbol{\Theta}) = (\Psi_k^d(\theta_1), \ldots, \Psi_k^d(\theta_K))$$

for any $\boldsymbol{\Theta} = (\theta_1, \ldots, \theta_K)$, and $\odot$ represents the tensor product of vectors. The indices in Eq. (7.4) vary in the ranges of values indicated throughout the section.

It is important to remark that the problem in Eq. (7.4) is over-determined, because we usually expect to fit the cloud of data points with many fewer

parameters than the number of data points (otherwise, we could simply interpolate all data points and hence, the fitting error would vanish). However, if we are provided with a given parameterization, the problem is not really a very serious issue, as it can readily be solved by standard numerical tools such as linear least-squares and other suitable routines, such as those described for the case of curves in Chapter 5 (actually, they operate properly on the multi-dimensional case, so they can be extended to the case of surfaces with little modification – if any).

In other words, the case where the parameterization is given is mathematically of little interest, since we can find its solution in a straightforward way. Actually, the real deal arises when no parameterization *a priori* is assumed. In this case, we have firstly to solve the parameterization problem, which is the real challenge of the surface reconstruction problem with polynomial parametric surfaces. It is at this critical step that the bat algorithm-based method proposed in this thesis is applied.

Note that since the blending functions $\{\Psi_i^m(u)\}_i$ and $\{\Psi_j^n(v)\}_j$ are nonlinear, minimization of E is a nonlinear continuous optimization problem. In many practical cases the number of data can be extremely large, meaning that we are also confronted with a high-dimensional problem. It is also multimodal, since there might be arguably more than one parameterization vector leading to the optimal solution. In summary, we have to deal with a difficult multimodal, high-dimensional, continuous, over-determined nonlinear optimization problem.

7.4 Our approach

In this section we describe the proposed method for solving the surface reconstruction problem indicated in the previous section. Firstly, a general overview of the method and the corresponding workflow are presented. Then, each step of the method is discussed in detail. Finally, the issue of parameter tuning is also discussed.

7.4.1 Overview of the method

The graphical workflow in Figure 7.1 summarizes the main steps of our method. The initial input consists of a set of irregularly sampled noisy 3D data points assumed to lie on an unknown surface. The goal is to obtain the polynomial Bézier surface that approximates these data points optimally. To this purpose, we need to solve two important sub-problems: data parameterization and surface approximation.

To address data parameterization, we apply the bat algorithm on a population of bats representing the potential solutions of the parameterization

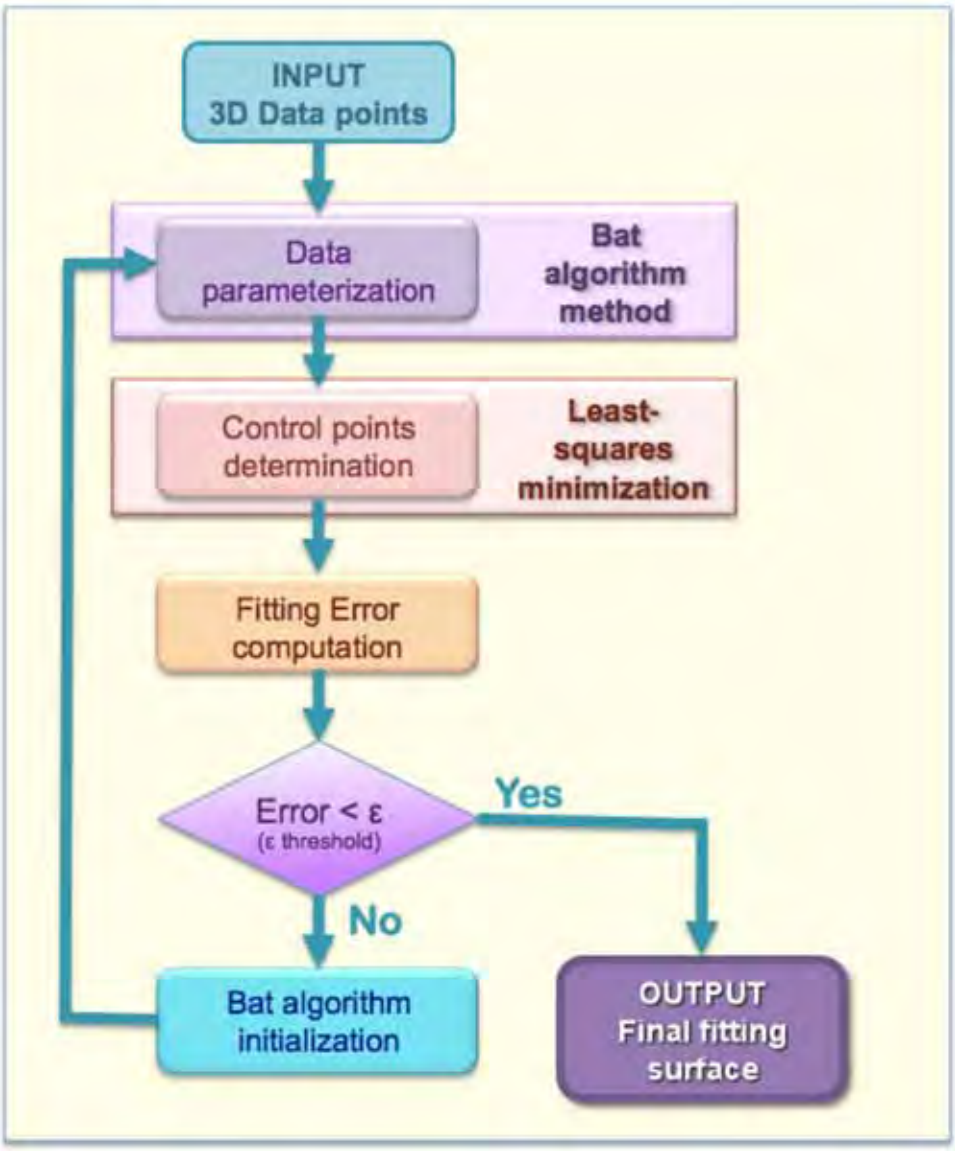

Figure 7.1: Graphical workflow of the proposed method.

problems for both parametric variables. Then, we perform least-squares minimization in order to compute the control points of the approximating polynomial Bézier surface. Finally, the fitting error is computed and compared against a threshold value. When our fitting error gets better than the threshold value, the method finishes and the approximating surface is returned. Otherwise, the bat algorithm is re-initializaed with random values and the procedure is re-started again.

7.4.2 Data parameterization

The parameterization step consists of establishing the relationships among the data points in the surface parametric domain. This process is essential for a good fitting of data points. Some standard procedures are given by the uniform, chord-length and centripetal parameterizations, already explained in Section 5.3.1. However, these methods are only suitable for data points distributed in a uniform grid, and tend to fail for unorganized, irregularly sampled data.

An alternative procedure is based on the idea of projecting the data points onto an additional surface, usually called base surface, reflecting the distribution of data points and then computing a parameterization by using the projected 2D points. The simplest case of this approach consists of using a parametric plane [93], usually orthogonal to the main viewing direction of the digitizing device. A better alternative is to use a suitable 3D surface for data projection [134], usually a coarse approximation of final fitting surface, which is expected to be modified by successive improvements of this initial surface.

Our approach to solve this parameterization problem consists of applying the bat algorithm described above to solve the least-squares minimization of functional E according to (7.3). To this aim, each bat, representing a potential solution, is given by: $\mathcal{B}_\zeta = (u_0^\zeta, u_1^\zeta, \ldots, u_m^\zeta; v_0^\zeta, v_1^\zeta, \ldots, v_n^\zeta) \in [0,1]^{m+1} \times [0,1]^{n+1}$ $(\zeta = 1, \ldots, \mathcal{P})$. In other words, the surface parameterization problem is subdivided into the individual sub-problems of obtaining a proper parameterization for each parametric variable. Note that this strategy is only possible because of the tensor product of the polynomial Béizer surface, meaning that it will not be available for the rational case addressed in next chapter. Both sublists are initialized with random values and then sorted, to reflect the ordered structure of the parametric values. The final output of this step is an optimal parameterization of data points.

7.4.3 Least-squares minimization

Using the data parameterization obtained in the previous step, we now perform least-squares minimization of the functional E according to (7.3). The output of the first step is a near-to-optimal parameterization of data points. Let $\mathbf{u} = (u_0, \ldots, u_m)$ denote such a parameterization for the variable u (a similar discussion can be done for the variable v; it is omitted here to avoid duplication of very similar material). Then, Eq. (7.4) can be rewritten as:

$$\begin{bmatrix} \boldsymbol{\Psi}_0^T \boldsymbol{\Psi}_0 & \cdots & \boldsymbol{\Psi}_n^T \boldsymbol{\Psi}_0 \\ \vdots & \vdots & \vdots \\ \mathbf{B}_0^T \mathbf{B}_n & \cdots & \boldsymbol{\Psi}_n^T \boldsymbol{\Psi}_n \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} = \begin{bmatrix} \mathbf{Q}\boldsymbol{\Psi}_0 \\ \vdots \\ \mathbf{Q}\boldsymbol{\Psi}_n \end{bmatrix} \tag{7.5}$$

Symbol	*Meaning*	*Range of Values*	*Chosen Value*
$\mathcal{P}$	population size	$10 - 200$	100
$\mathcal{G}_{max}$	maximum number of generations	$100 - 3000$	2000
$\mathcal{A}^0$	initial loudness	$(0, 2)$	0.5
$\mathcal{A}_{min}$	minimum loudness	$(0, 1)$	0.5
r^0	initial pulse rate	$[0, 1]$	0.5
f_{max}	maximum frequency	$[0, 10]$	2
α	multiplicative factor	$(0, 1)$	0.3
γ	exponential factor	$[0, 1]$	0.3

Table 7.1: Parameters and values used in our method.

Surface Example	*# data points*	*RMSE fitting error*	*E fitting error*
1	2499	$1.116440e-3$ (mean) $2.752097e-4$ (best)	$3.388225e-3$ (mean) $1.892753e-4$ (best)
2	1050	$2.833084e-3$ (mean) $7.200351e-4$ (best)	$8.427686e-3$ (mean) $5.443732e-4$ (best)
3	812	$3.229615e-3$ (mean) $9.731133e-4$ (best)	$8.469497e-3$ (mean) $7.689231e-4$ (best)

Table 7.2: The three examples in this paper with their number of data points and the mean and best $RMSE$ and E errors.

where $\mathbf{\Psi}_j = (\Psi_j^n(u_1), \ldots, \Psi_j^n(u_m))^T$ represents a column vector, $\mathbf{b}_i$ represents the control points of the surface along the parametric direction u, $\mathbf{Q} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_m)$ a row vector of the data points along a parametric direction, and $(.)^T$ represents the transpose of a vector or matrix.

System (7.5) is overdetermined, so no analytical solution can be obtained. Instead, we obtain an approximated solution through least-squares minimization. If $\mathbf{\Psi}^+$ denote the *generalized inverse* (also known as *Moore-Penrose pseudo-inverse*) of $\mathbf{\Psi} = (\Psi_i)^T$, $\mathbf{P} = \mathbf{\Psi}^+\mathbf{Q}$ is the solution of our problem in the least-squares sense.

7.4.4 Parameter tuning

Regarding the issue of the parameter tuning, we already know that the choice of suitable values for the parameters of the bat algorithm method is strongly

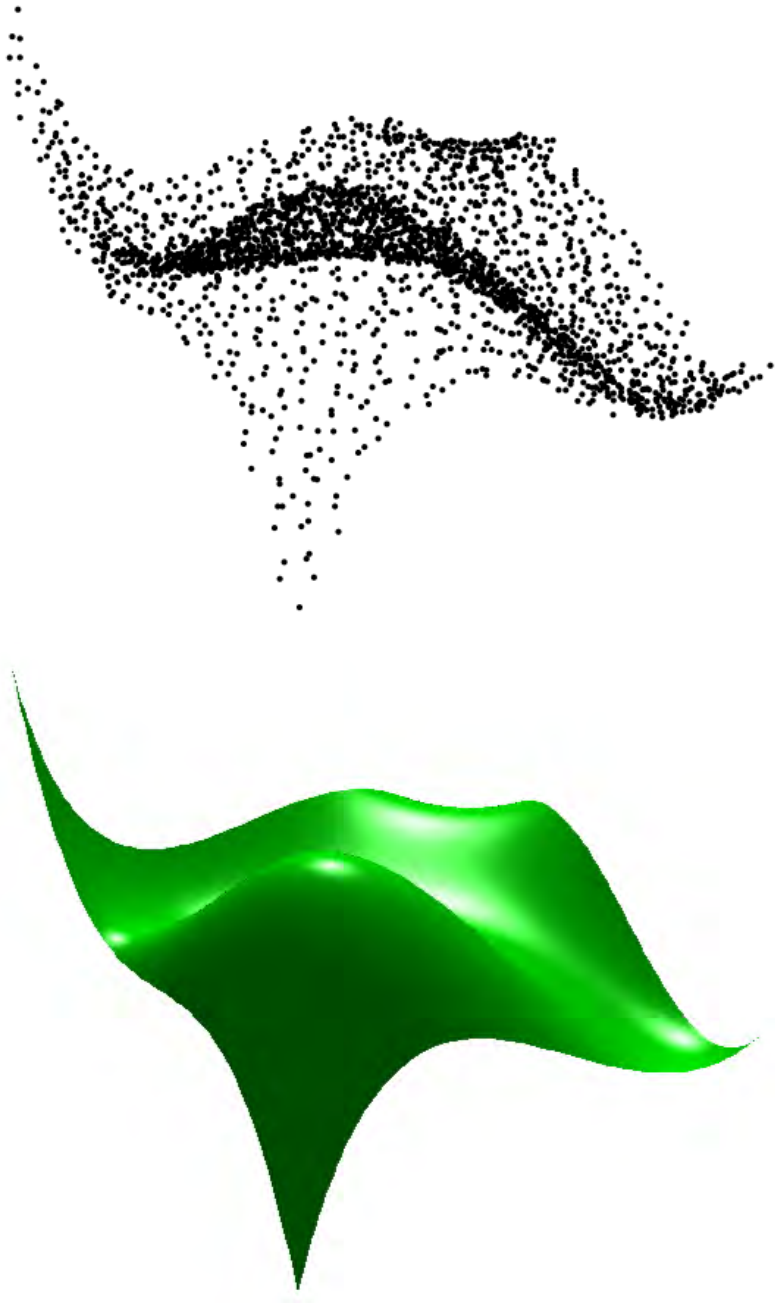

Figure 7.2: Application of our method to the first example: (top) original cloud of irregularly sampled data points; (bottom) best polynomial Bézier fitting surface.

dependent on the particular problem at hand, making it hard to determine good values in advance. Therefore, our choice here is fully empirical. To

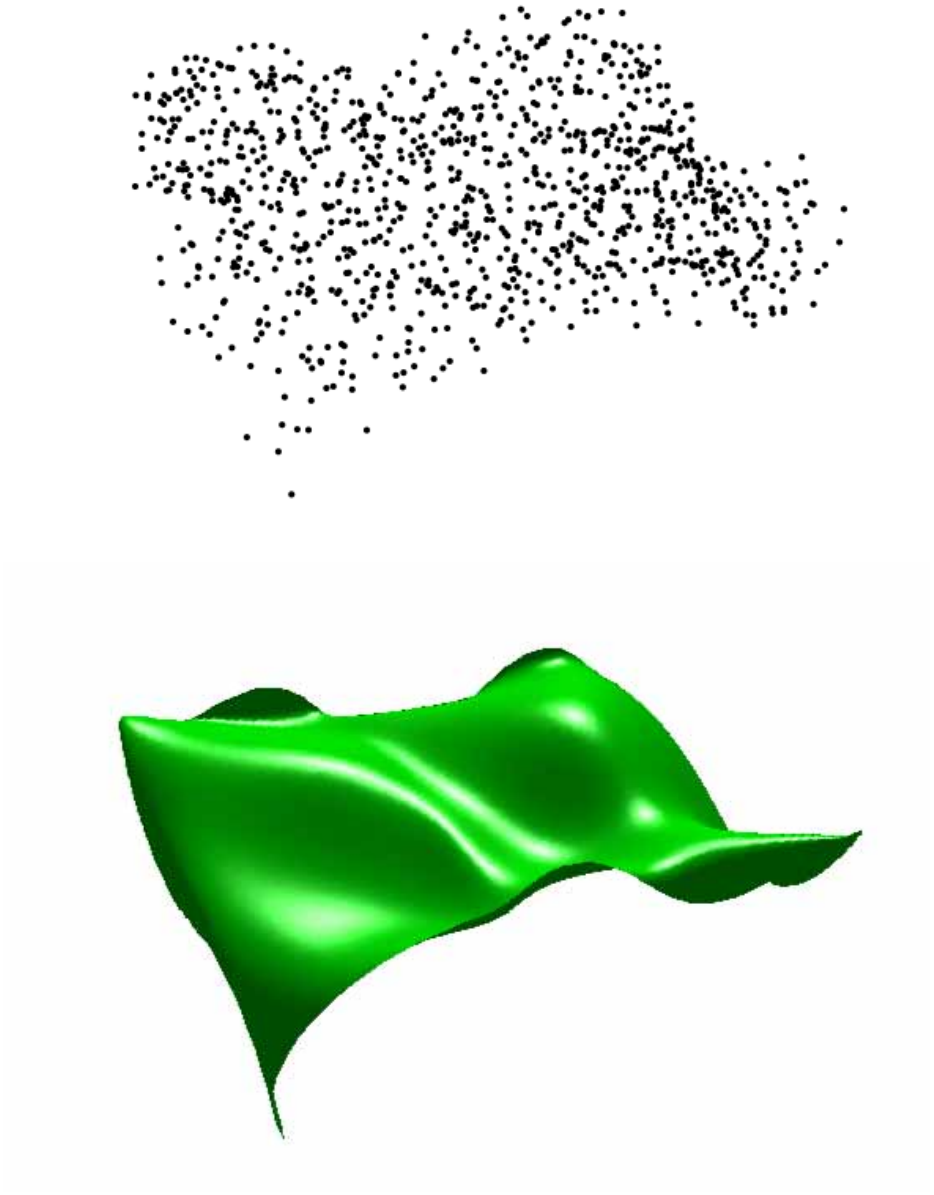

Figure 7.3: Application of our method to the second example: (top) original cloud of irregularly sampled data points; (bottom) best polynomial Bézier fitting surface.

this purpose, we carried out numerous computer simulations for different parameter values, finally reaching suitable values for our method.

The different parameters used in our method are arranged in rows in Table 7.1. For each parameter, the table shows (in columns) its symbol,

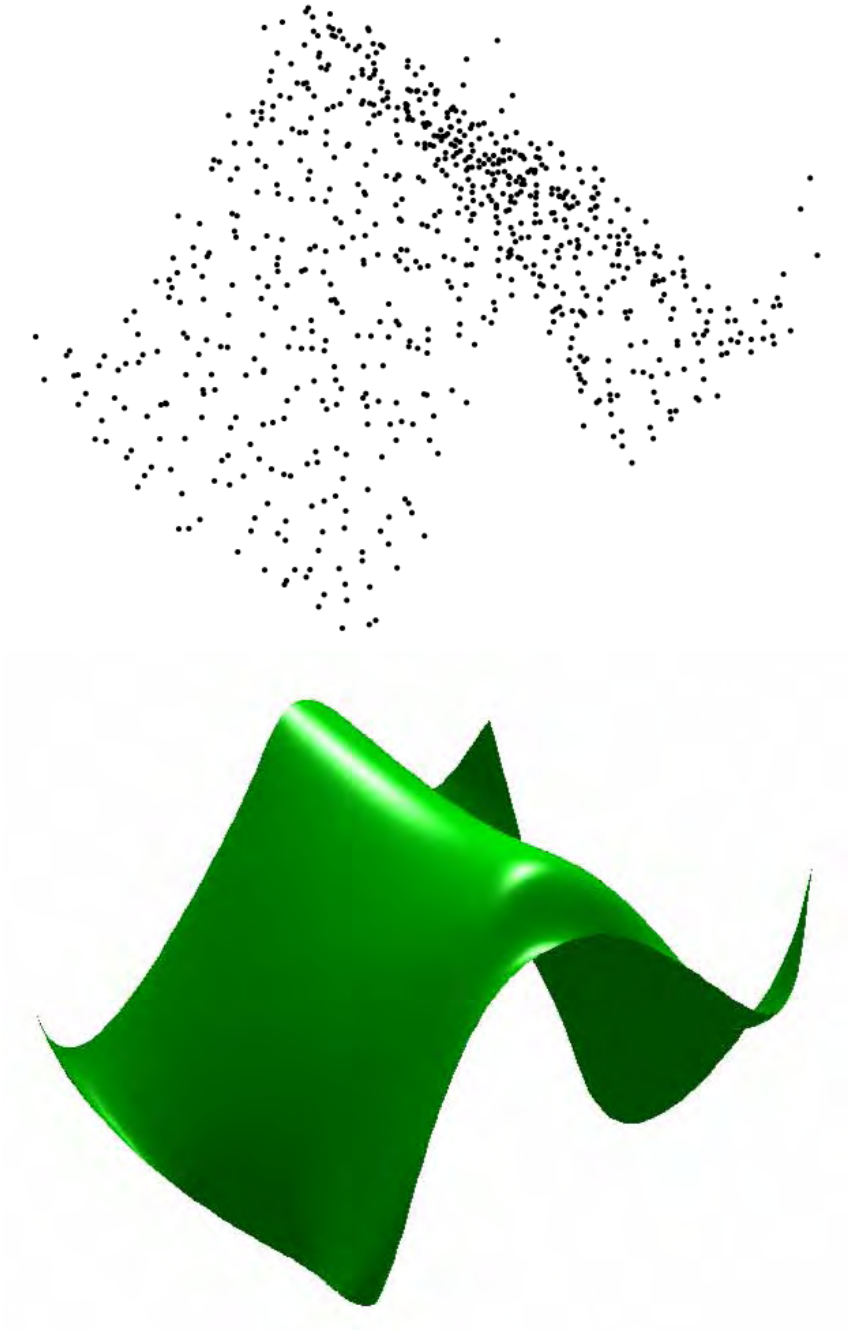

Figure 7.4: Application of our method to the third example: (top) original cloud of irregularly sampled data points; (bottom) best polynomial Bézier fitting surface.

meaning, range of values, and the parameter value chosen in this paper.

With this choice of parameters, the bat algorithm is executed iteratively for a previously chosen number of generations, or until convergence is eventually reached. Final positions and velocities of the bats are computed and ranked according to our fitness function E. The position of the particle with the global best is then chosen as the winner (i.e., the final solution of our minimization problem).

7.5 Experimental Results

Our method has been applied to several examples. In this section, we focus on three of them. They are arranged in rows in Table 7.2. For each example, the table shows (in columns) the number of data points used in our experiments along with the mean and the best fitting errors for the $RMSE$ (root-mean square error) on the left and for the for E (given by Eq. (**??**)) on the right. The $RMSE$ of our problem is given by:

$$RMSE = \sqrt{\frac{E}{(m+1) \times (n+1)}}$$

The mean error has been obtained from 50 independent executions of the algorithm (but with the same parameter values for the method), while the best value corresponds to the results of the best execution from the 50 runs.

As the reader can see, mean errors are of order 10^{-3} for both the $RMSE$ and E fitting errors, while the best results are of order 10^{-4} for the three examples in our benchmark. From these data we conclude that the method works pretty well, as it is able to replicate the original shape with very good accuracy for all examples in this paper.

These good numerical results are visually confirmed in Figures 7.2, 7.3 and 7.4, which represent graphically the three examples in our benchmark. For each example, the figure above displays the cloud of original data points displayed as black circles, while the best fitting surface is shown below. Each surface in the bottom is displayed in green and then shaded with different light sources the help the reader to identify the different features of the surface, such as the hills and valleys and enhance the general visualization of the fitting surface.

Notice that the clouds of points are very irregularly sampled, making it hard (or impossible) to figure out the shape of the surface. Note also, however, the good visual fitting between the original points and the approximating surface for the three examples. They show that our method is able to capture the underlying shape of data, even in the case of clouds with a severe irregular pattern.

7.6 Main Conclusions

This chapter addresses the problem of surface reconstruction problem by using polynomial Bézier surfaces. This problem is strongly related to that of surface parameterization, because the data fitting problem is only really challenging when no parameterization is given, but considered to be a subset of the unknowns of the corresponding optimization problem. In this sense, our proposal to solve the surface reconstruction problem with polynomial Bézier surfaces can be basically converted into that of solving the polynomial Bézier surface parameterization from clouds of irregularly sampled data points. This problem itself has many applications in several applied and industrial fields. It is also a very difficult mathematical issue; in fact, it leads to a high-dimensional, nonlinear, overdetermined, continuous optimization problem.

Our method for surface reconstruction is based in two main steps: surface parameterization and surface fitting. The fist step is based on the application of the bat algorithm to obtain a suitable parameterization of the fitting surface from the data points. Then, surface fitting is carried out by means of the least-squares methods for optimization of the values of the control points of the approximating surface.

These two steps are repeated until the fitting error satisfies a prescribed threshold error. Any time this condition does not hold, the procedure is repeated by applying a re-initialization of the bat algorithm in order to promote diversity and improve the convergence rate.

To analyze the goodness of our scheme, it was applied to three examples of Bézier surfaces. Other examples (omitted here due to keep the chapter in manageable size) have also been tested with satisfactory results. Our numerical and visual results confirm the good performance of this proposal even for clouds of strong irregular patterns, a very challenging issue for many optimization techniques.

In fact, the possibility to apply our method to noisy clouds of data points is an important contribution with respect to some previous works in the field. Many previous approaches tend to restrict themselves to clouds of points that follow a strong quadrilateral structure. Although at first sight this is not an important feature, it is very relevant in practical terms since it means that we already have an internal topological structure that can be advantageously used to determine a suitable surface parameterization without the difficulties typically found in noisy clouds of data. Furthermore, such topological structure is often used to skip the parameterization step by simply considering rather simple (even naive) strategies like projecting the data points onto a reference plane, as the the cloud of points was always associated to a height map. When the data points are affected by noise, these projection-based

strategies tend to fail, because they do no longer maintain the squared structure. As a consequence, more powerful methods are required to solve the parameterization issue. In this sense, the presented method is a significant contribution to solve the general surface reconstruction problem, with potential applications to real-world problems where the noise in data is the norm rather than the exception.

Chapter 8

Surface Reconstruction with Rational Bézier Surfaces

In the previous chapter we addressed the problem of surface reconstruction from clouds of noisy data points with polynomial Bézier surfaces. We showed that when no parameterization is assumed in the process, this issue leads to a difficult nonlinear continuous optimization problem. Our proposal to tackle this issue was based on two steps: in the first one, the bat algorithm was applied to obtain a suitable parameterization of the approximating surface; in the second step, we performed least-squares optimization to compute the control points of the approximating polynomial surface and achieve surface fitting.

The aim of the present chapter is to extend our previous method for polynomial surfaces to the (more challenging) case of surface reconstructions with rational Bézier surfaces. This problem is more difficult than it might seem at first sight. The main reasons are:

- The free-form rational Bézier surfaces depend on many different sets of variables (data parameters, poles, weights, surface degree) that are qualitatively different regarding their behavior and their effect on the fitting surface.
- Furthermore, all these sets of variables are strongly related to each other in a highly complex and intertwined way, leading to a strongly *nonlinear continuous optimization problem*.
- It is also a *multivariate* problem, as it typically involves a large number of unknown variables for a large number of data points, the most common case in real-world applications.
- Finally, the problem is also *multimodal*, i.e., the least-squares objective function can exhibit many local optima, meaning that the problem

might have several (global and/or local) good solutions.

Note that although our approach in this paper also relies on the bat algorithm, the current problem is much more difficult and requires a more sophisticated approach. In particular, in this work we propose a new method where the bat algorithm is sequentially applied to compute the data parameters and the weights. This process is performed iteratively by injecting the output of each bat algorithm as the input of the next one, and so on. Finally, the poles are computed through classical least-squares optimization.

The structure of this paper is as follows: Section 8.1 introduces the problem to be solved. Section 8.2 describes some previous work in the field. Then, the fundamentals about rational surfaces as well as the data fitting problem with rational Bézier surfaces are discussed in Section 8.3. Section 8.4 describes our approach to solve this surface reconstruction problem with rational surfaces. The computational results obtained by application of our method to some illustrative examples are described in Section 8.5. Finally, Section 8.6 reports the main conclusions of this work.

8.1 Introduction

As discussed above, the previous chapter addressed the problem of surface reconstruction with free-form polynomial Bézier surfaces. In that case, the problem consists of determining a suitable data parameterization, a very important factor for a good fitting. The proposal to solve that problem was based on the bat algorithm, to solve the parameterization step, while least-squares optimization was applied to compute the control points.

In spite of these good results of that approach for several examples (see our previous discussion in Section 7.5), the polynomial surfaces are still strongly limited as they cannot adequately describe some particular shapes, such as the quadrics. As a consequence, there is still a need for more powerful and more general blending functions.

An interesting extension in this regard is given by the rational basis functions, which are mathematically described as the quotient of two polynomials. A remarkable advantage of this rational scheme is that the quadrics and other shapes can be canonically described as rational functions. Unfortunately, this rational approach becomes more difficult than the polynomial one, since new parameters are now introduced into the problem. Consequently, we are confronted with the challenge of obtaining optimal values for many (qualitatively different) parameters, namely, data parameters, poles, and weights. This leads to an even more difficult optimization problem, as it will described in this chapter.

8.2 Previous Work

The problem of surface reconstruction has been an important issue of research for many years [29, 154, 157]. In many practical cases the only available information about the problem is a (typically dense and often unorganized) cloud of noisy 3D data points obtained by using some sort of digitizing devices. In that case, the reconstructed surface can be represented in three different levels of accuracy:

- The coarsest representation is given by the polygonal mesh. in this model, data points as used as vertices connected by lines (edges) that work together to create a 3D model, comprised of vertices, edges and faces. This representation is the first choice for many computer graphics tasks, since it is very flexible and quicker to render and well suited for dealing with current graphical cards. However, *polygonal meshes are never a truly faithful representation of a smooth surface.* They are merely linear approximations of curved shapes; as such, they only provide a coarse representation of real objects: in a polygonal mesh, curves are approximated by linear segments, while surfaces are approximated by triangular or quadrilateral flat polygons. In this sense, polygonal meshes provide the lowest degree of accuracy, being mostly used for coarse geometry and fast rendering. Surface reconstruction methods with polygonal meshes can be found, for instance, in [46, 90, 91, 124, 148, 158] and references therein.

- The next level is given by the constructive solid geometry models (CSG models). in this model, elementary geometries (such as spheres, boxes, cylinders or cones) are combined in order to produce more elaborated shapes by applying some simple (Boolean) operators: union, intersection, difference. Although often CSG (Constructive Solid Geometry) presents a model or surface that appears visually complex, it is actually little more than a clever combination of simple objects by means of rather simple operations. As such, it is also limited in terms of the kind of objects it can represent: multi-branched, self-intersecting or high-genus objects are very hard (if not impossible) to be constructed with this technology. CSG models are barely applied to surface reconstruction because their extreme simplicity is a limiting factor in order to recover non-trivial shapes.

- The most sophisticated and most accurate level is described in terms of the mathematical equation of the fitting surface, in particular, by the *free-form parametric surfaces.* They provide the most sophisticated and most accurate representation of smooth real-world objects. Among

them, the rational surfaces are the most powerful free-form parametric surfaces because of their flexibility, versatility, and the fact that they represent well a wide variety of shapes in a very compact and intuitive mathematical form. This is the preferred form for detailed surfaces, since they faithfully represent real objects: using rational surfaces, a sphere can be faithfully represented as a true mathematical sphere, not a simple collection of flat polygons resembling a sphere.

Several papers in the literature addressed this problem through different mathematical representations: subdivision surfaces [176], function reconstruction [47, 178], implicit surfaces [128], algebraic surfaces [159], hierarchical splines [49], and so on.

However, in spite of all these references to previous work in the field, none of the existing papers in the literature reports a method solving this problem for rational Bézier surfaces. This fact clearly indicates the originality and opportunity of this research work.

8.3 Description of the Problem

In this section we provide the reader with the mathematical description of a rational Bézier surface. Then, we describe the problem of data fitting with rational Bézier surfaces.

8.3.1 Mathematical background

We recall the reader that a *free-form polynomial Bézier surface* $\mathbf{\Phi}(t,s)$ *of degree* (η,σ) in $\mathbb{R}^d$ is given by:

$$\mathbf{\Phi}(t,s) = \sum_{i=0}^{\eta}\sum_{j=0}^{\sigma} \mathbf{\Lambda}_{i,j}\phi_i^{\eta}(t)\phi_j^{\sigma}(s) \tag{8.1}$$

where $\{\mathbf{\Lambda}_{i,j}\}_{i=0,\ldots,\eta;j=0,\ldots,\sigma}$ are vector coefficients called the *poles*, and the functions $\phi_r^{\rho}(\upsilon)$ are the *Bernstein polynomials of index* r *and degree* ρ, given by:

$$\phi_r^{\rho}(\upsilon) = \binom{\rho}{r}\upsilon^r(1-\upsilon)^{\rho-r} \quad \text{with} \quad \binom{\rho}{r} = \frac{\rho!}{r!(\rho-r)!} \quad \text{for} \quad r=0,\ldots,\rho \tag{8.2}$$

and υ is the function parameter, defined on the unit interval $[0,1]$.

The polynomial representation in Eqs. (8.1)-(8.2) is not powerful enough to represent a variety of shapes, particularly the *quadratic surfaces* or *quadrics*, such as cones, cylinders, ellipsoids, paraboloids, hyperboloids, spheres, and spheroids, which are very important in many different fields.

One way to overcome this limitation is to use homogeneous coordinates (see [38, 154] for details). The basic idea is to consider the projection of the standard polynomial Bézier surface in $\mathbb{R}^{d+1}$, with new poles $\mathbf{\Lambda}^h_{i,j}$. The resulting surface in $\mathbb{R}^d$ is called a rational Bézier surface. Mathematically, this surface can be described as a quotient of two bivariate polynomial functions.

In particular, a *free-form rational Bézier surface* $\mathbf{\Psi}(t,s)$ *of degree* (η,σ) in $\mathbb{R}^d$ is given by:

$$\mathbf{\Psi}(t,s) = \frac{\displaystyle\sum_{i=0}^{\eta}\sum_{j=0}^{\sigma} \omega_{i,j}\mathbf{\Lambda}_{i,j}\phi_i^{\eta}(t)\phi_j^{\sigma}(s)}{\displaystyle\sum_{i=0}^{\eta}\sum_{j=0}^{\sigma} \omega_{i,j}\phi_i^{\eta}(t)\phi_j^{\sigma}(s)} \tag{8.3}$$

where $\omega_{i,j}$ are positive scalar weights associated with the poles $\mathbf{\Lambda}_{i,j}$. Considering the *rational bivariate Bernstein basis functions*:

$$\mathbf{\Xi}^{\eta,\sigma}_{k,l}(t,s) = \frac{\omega_{k,l}\phi_k^{\eta}(t)\phi_l^{\sigma}(s)}{\displaystyle\sum_{i=0}^{\eta}\sum_{j=0}^{\sigma} \omega_{i,j}\phi_i^{\eta}(t)\phi_j^{\sigma}(s)} \qquad (k=0,\ldots,\eta; l=0,\ldots,\sigma) \tag{8.4}$$

expression (8.3) can be rewritten as:

$$\mathbf{\Psi}(t,s) = \sum_{k=0}^{\eta}\sum_{l=0}^{\sigma} \mathbf{\Lambda}_{k,l}\,\mathbf{\Xi}^{\eta,\sigma}_{k,l}(t,s). \tag{8.5}$$

Note that weights $w_{k,l}$ are the last coordinates of the homogeneous poles $\mathbf{\Lambda}^h_{k,l}$. This new set of parameters provides us with additional degrees of freedom for better shape approximation. However, they also increase the model complexity because we introduce a new set of parameters that are also to be computed.

8.3.2 The surface fitting problem

Let us suppose now that we are given a set of organized 3D data points $\{\mathbf{\Delta}_{p,q}\}_{p=1,\ldots,m;q=1,\ldots,n}$. In this paper we assume that the data points are always affected by measurement noise of low/medium intensity.

In this case, the problem consists of obtaining the rational Bézier surface, $\mathbf{\Psi}(t,s)$, of a certain degree (η,σ) providing the best least-squares fitting of the data points. This leads to a minimization problem of the least-squares error functional Υ, related to the weighted sum of squares of the residuals:

$$\Upsilon = \underset{\substack{\{(t_p,s_q)\}_{p,q}\\ \{\boldsymbol{\Lambda}_{i,j}\}_{i,j}\\ \{\omega_{i,j}\}_{i,j}}}{\text{minimize}} \left[\sum_{p=1}^{m} \sum_{q=1}^{n} \left(\boldsymbol{\Delta}_{p,q} - \frac{\sum_{i=0}^{\eta} \sum_{j=0}^{\sigma} \omega_{i,j} \boldsymbol{\Lambda}_{i,j} \phi_i^{\eta}(t_p) \phi_j^{\sigma}(s_q)}{\sum_{i=0}^{\eta} \sum_{j=0}^{\sigma} \omega_{i,j} \phi_i^{\eta}(t_p) \phi_j^{\sigma}(s_q)} \right)^2 \right]. \tag{8.6}$$

The case of unorganized data points can be handled in a similar way. Let now $\{\boldsymbol{\Delta}_{\xi}\}_{\xi=1,\ldots,\kappa}$ be a set of unorganized 3D data points in $\mathbb{R}^d$. In this case, the minimization problem of the least-squares error functional $\tilde{\Upsilon}$ becomes:

$$\tilde{\Upsilon} = \underset{\substack{\{t_{\xi}\}_{\xi}\\ \{s_{\xi}\}_{\xi}\\ \{\boldsymbol{\Lambda}_{i,j}\}_{i,j}\\ \{\omega_{i,j}\}_{i,j}}}{\text{minimize}} \left[\sum_{\xi=1}^{\kappa} \left(\boldsymbol{\Delta}_{\xi} - \frac{\sum_{i=0}^{\eta} \sum_{j=0}^{\sigma} \omega_{i,j} \boldsymbol{\Lambda}_{i,j} \phi_i^{\eta}(t_{\xi}) \phi_j^{\sigma}(s_{\xi})}{\sum_{i=0}^{\eta} \sum_{j=0}^{\sigma} \omega_{i,j} \phi_i^{\eta}(t_{\xi}) \phi_j^{\sigma}(s_{\xi})} \right)^2 \right]. \tag{8.7}$$

Note that solving either (8.6) or (8.7) (for organized or unorganized data points, respectively), requires to compute all parameters, i.e., poles $\{\boldsymbol{\Lambda}_{i,j}\}_{i,j}$, weights $\{\omega_{i,j}\}_{i,j}$, and parameters $\{(t_p, s_q)\}_{p,q}$ or $\{t_{\xi}\}_{\xi}$ and $\{s_{\xi}\}_{\xi}$ (associated with data points $\{\boldsymbol{\Delta}_{p,q}\}_{p,q}$ or $\{\boldsymbol{\Delta}_{\xi}\}_{\xi}$, respectively) of the approximating surface.

It is obvious that, since each blending function in (8.2) and (8.4) is non-linear in t and s, the problem (8.6) or (8.7) becomes highly non-linear. It is also a continuous problem, since all parameters are real-valued. In many practical cases the number of data can be extremely large, meaning that we are also confronted with a high-dimensional problem. The problem is also multimodal, since there might be arguably several optima of the target function.

In summary, we have to deal once again with a difficult multimodal, high-dimensional, continuous, nonlinear optimization problem. Unfortunately, the classical optimization techniques cannot solve this problem in all its generality. Instead, only partial solutions have been reported in the literature so far. Clearly, more powerful strategies are needed to tackle this issue. The research work reported in this chapter aims at filling this gap with the approach described in next section.

8.4 The Proposed Method

As discussed above, our problem consists of reconstructing the underlying shape of a cloud of noisy 3D data points by using a rational Bézier sur-

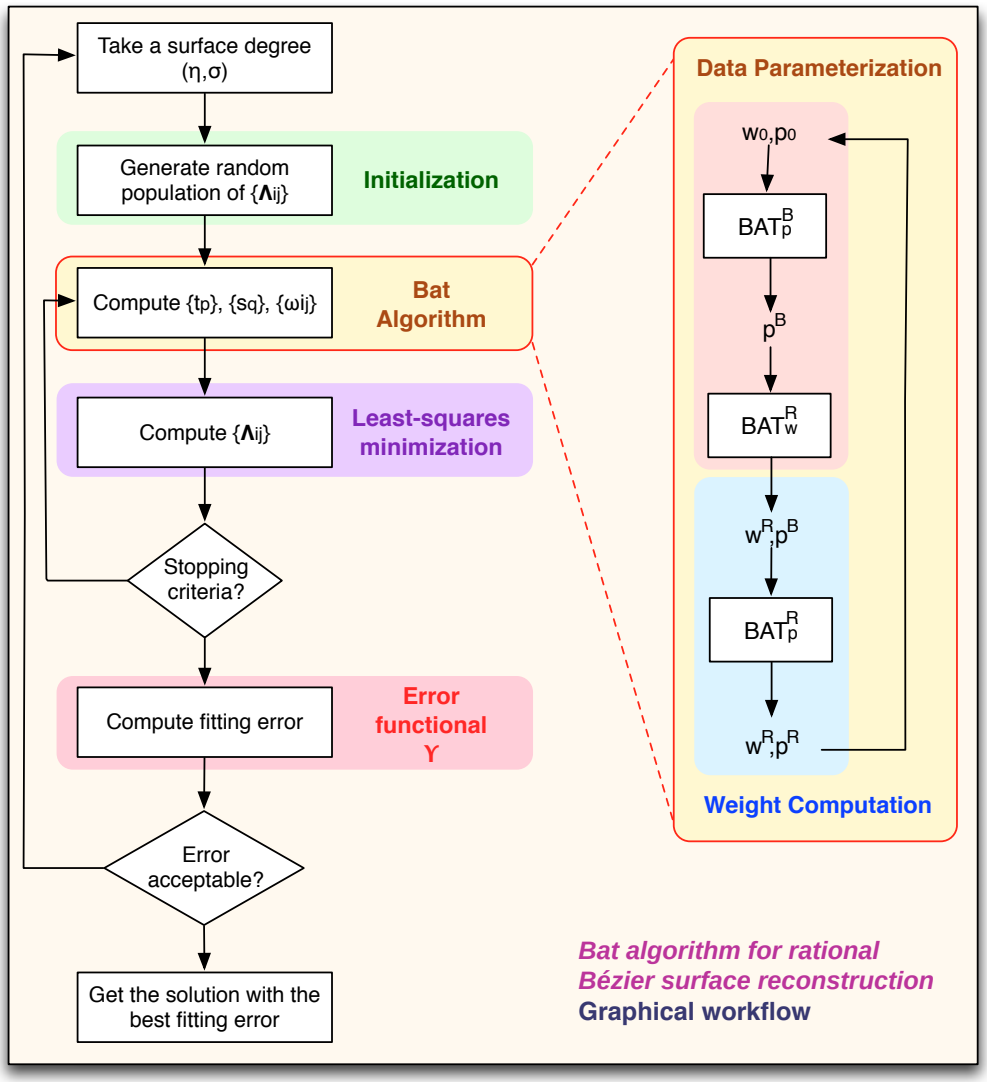

Figure 8.1: Graphical workflow of the proposed method for rational Bézier surface reconstruction.

face, leading either to functional (8.6) or (8.7). Solving this problem requires to compute three different sets of unknowns: data parameters, poles, and weights. The proposed method to tackle this issue is a hybrid strategy combining the bat algorithm described in Section 4.9 and the least-squares minimization. As it will be discussed later on, it consists of three major steps: data parametrization, weight computation, and data fitting.

Figure 8.1 shows the graphical workflow of our proposal. The different steps of the method are enclosed in areas of different background color. The

name of the different phases is also displayed in different colors for easier visual identification. In general, the text color is similar to the background color of the corresponding area, but with a darker tone. The arrows from top to bottom show the flow of work for the different steps in the method.

The method can be summarized as follows: assuming a certain surface degree (η, σ), the method starts with the initialization process, where the data parameters $\{(t_p, s_q)\}_{p,q}$, the poles $\{\mathbf{\Lambda}_{i,j}\}_{i,j}$ and the weights $\{\omega_{i,j}\}_{i,j}$ are all initialized with random uniform values over their respective domains. Then, we apply the bat algorithm iteratively to perform data parameterization and weight computation, as described in Section 8.4.1. Then, data fitting is performed via least-squares to compute the poles of the surface, as described in Section 8.4.2. Some further improvements on this scheme are discussed in Section 8.4.3. Finally, the issue of parameter tuning for the bat algorithm is discussed in detail in Section 8.4.4.

8.4.1 Data parameterization and weight computation

First two steps of our method are data parameterization and weight computation. They are displayed on the right part of Figure 8.1 in background colors light brown and cyan, respectively. In the figure, the different data parameters are represented by a single symbol $\mathbf{p}_i$ for simplicity. Similarly, the weights are represented by symbol $\mathbf{w}_j$. The super-script B is used to mean that the surface is a standard (i.e., polynomial) Bézier surface, while super-script R means the surface is rational.

The goal of data parameterization is to obtain an association between the set of parameters $\{t_p, s_q\}_{p,q}$ and the data points $\{\mathbf{\Delta}_{p,q}\}_{p,q}$. At its turn, weight computation seeks to compute the best values for the weights $\{\omega_{i,j}\}_{i,j}$. Note that although both sets of variables are quite different in nature, they are not actually independent of each other, because the parameterization affects the values of the weights, and conversely. This means that they cannot be computed independently; instead, both steps are to be carried out simultaneously.

In our method, data parameters and weights are computed by using an iterative sequential procedure where at each iteration one set of variables is initially computed and their values are then subsequently used as input values to obtain the remaining set of variables. This sequence is performed iteratively until a prescribed stopping criterion is reached.

Let us now explain our method in detail. The initial input of our method is the surface degree (η, σ) along with the cloud of data points, $\{\mathbf{\Delta}_{p,q}\}_{p,q}$. To apply the bat algorithm, the population of bats, representing the candidate solutions of our minimization problem, corresponds to the sets of unknown variables. Since we have to deal with two sets of parameters and owing to

the matrix structure of the data parameters, we consider three populations of bats, namely, $\mathcal{T} = \{\mathbf{t}_c\}_c$, $\mathcal{S} = \{\mathbf{s}_d\}_d$, and $\mathcal{W} = \left\langle (\{\omega_{i,j}\}_{i,j})^T \right\rangle$, where $\mathbf{t}_c = \{t_p^c\}_p$, $\mathbf{s}_d = \{s_q^d\}_q$, $\langle . \rangle$ denotes the vectorization of a matrix (the linear transformation that converts the matrix into a column vector by stacking its columns on top of one another) and $(.)^T$ denotes the transpose of a vector or matrix.

All data parameters in $\mathbf{t}_c$ (resp. $\mathbf{s}_d$) are initialized with random numbers within the hypercube $[0,1]^m$ (resp. $[0,1]^n$) and then sorted to replicate the structure of data points. The weights in $\mathcal{W}$ are randomly initialized within the search domain $(0,100]^{(\eta+1)\times(\sigma+1)}$.

We also set the value of parameters of the bat algorithm (see Section 8.4.4 for details). Regarding our stopping criterion for the bat algorithm, we run this procedure until no further improvement is reached after 30 consecutive generations.

The sequential procedure begins with data parameterization for the polynomial Bézier surface according to the procedure described in our previous Chapter 7. That procedure applies the bat algorithm with the initial random population in $\mathcal{T}$ and $\mathcal{S}$ to obtain an initial data parameterization for the polynomial surface, denoted by $\mathbf{t}_*^0$ and $\mathbf{s}_*^0$, where the superscript is used to indicate the iteration of our iterative sequential process[1]. This parameterization is used as an initial seed to compute a set of weights for the rational Bézier surface by applying the bat algorithm with the random set $\mathcal{W}$.

Let us now call the resulting optimal weight vector $\mathbf{w}_*^0$. Now, we start the iterative process of the sequence *data parameterization–weight computation*. For every iteration δ, we consider the following procedure:

(S1) *data parameterization*: run the bat algorithm for the data parameters by using the populations $\mathcal{T}^\delta$ and $\mathcal{S}^\delta$ for the optimal vector $\mathbf{w}_*^{\delta-1}$. This step yields the new optimal vectors $\mathbf{t}_*^\delta$ and $\mathbf{s}_*^\delta$.

(S2) *weight computation*: run the bat algorithm for the weights by using the population $\mathcal{W}^\delta$ for the optimal vectors $\mathbf{t}_*^\delta$ and $\mathbf{s}_*^\delta$ obtained in step (S1). This step yields the new optimal vector $\mathbf{w}_*^\delta$.

(S3) Repeat the sequence (S1)-(S2) until there is no further improvement of the resulting solution.

As a consequence, this procedure returns the optimal values for the data parameters and the weights, denoted by $\mathbf{t}_*$, $\mathbf{s}_*$, and $\mathbf{w}_*$, respectively. They are used to compute the poles, as described in next section.

[1]The reader is kindly warned about the difference between the number of iterations of the bat algorithm, called generations in this chapter, and the number of iterations in our iterative procedure of the sequence: *data parameterization–weight computation*.

8.4.2 Data fitting

In this step, we calculate the surface poles $\{\mathbf{\Lambda}_{i,j}\}_{i,j}$ using the $\mathbf{t}_*$, $\mathbf{s}_*$, and $\mathbf{w}_*$ obtained in previous section. Using (8.5), Eq. (8.6) can be rewritten as:

$$\mathbf{\Delta} = \mathbf{\Xi}.\mathbf{\Lambda} \tag{8.8}$$

where $\mathbf{\Delta} = \left\langle (\{\mathbf{\Delta}_{p,q}\}_{p,q})^T \right\rangle$, $\mathbf{\Lambda} = \left\langle (\{\mathbf{\Lambda}_{i,j}\}_{i,j})^T \right\rangle$, and where the matrix $\mathbf{\Xi}$ represents the vector of all rational basis functions given by Eq. (8.4) at the best parameter values, given by:

$$\mathbf{\Xi} = \left\langle \left(\left\{ \left\langle \left(\{ (\mathbf{\Xi}_{k,l}^{\eta,\sigma}((t_p)_*, (s_q)_*))_* \}_{p,q} \right)^T \right\rangle \right\}_{k,l} \right)^T \right\rangle$$

Note that vector $\mathbf{\Delta}$ has a length $m \times n$ while vector $\mathbf{\Lambda}$ has length $(\eta+1) \times (\sigma+1)$, so the system (8.8) is over-determined, meaning that no analytical solution can be obtained. Pre-multiplication of both sides by $\mathbf{\Xi}^T$ gives:

$$\mathbf{\Xi}^T.\mathbf{\Delta} = \mathbf{\Xi}^T.\mathbf{\Xi}.\mathbf{\Lambda} \tag{8.9}$$

which can now be solved numerically by a classical linear least-squares minimization. From a computational point of view, it can be obtained by either LU decomposition or singular value decomposition (SVD). In this work, we choose SVD because it returns the best answer of this least-squares problem. To this purpose, SVD computes the generalized inverse (also known as Moore-Penrose pseudo-inverse) of $\mathbf{\Xi}$, denoted by $\mathbf{\Xi}^+$. Then, $\mathbf{\Lambda} = \mathbf{\Xi}^+.\mathbf{\Delta}$ is the least-squares solution of this data fitting problem.

8.4.3 Further improvements

We introduce some modifications on the original procedure described above to further enhance the performance of our method:

- we improve the memory capacities of the method through *elitism*: the best state from the current generation is always preserved unaltered for the next generation. The effect of this new feature is to improve the convergence rate with respect to the standard (non-elitist) version of the bat algorithm.

- we add a new operator related to the domain of the problem. This extra functionality checks whether a new generated solution at generation g goes outside the search domain of the problem and sends it back into the search space whenever it goes away. To do it, we consider a uniform random convex combination of the solution at generation $g-1$ (assumed

Symbol	*Meaning*	*Range of Values*	*Chosen Value*
$\mathcal{P}$	population size	$10 - 200$	100
$\mathcal{G}_{max}$	maximum number of generations	$100 - 10000$	7000
$\mathcal{A}^0$	initial loudness	$(0, 2)$	0.5
$\mathcal{A}_{min}$	minimum loudness	$(0, 1)$	0.1
r^0	initial pulse rate	$[0, 1]$	0.5
f_{max}	maximum frequency	$[0, 10]$	2
α	multiplicative factor	$(0, 1)$	0.3
γ	exponential factor	$[0, 1]$	0.3

Table 8.1: Parameters and values used in our method.

to be inside the search domain) and the projection of the new solution at generation g outside the search domain on the domain boundary. Clearly, the resulting solution of this convex combination is inside the search domain and does not modify the probability of the boundary at all.

- finally, we speed up the bat algorithm execution by considering two stopping criteria working simultaneously: the bat algorithm procedure stops when it reaches a maximum number of generations or when no further improvement is reached after 30 consecutive generations, whatever comes first.

These new features improved the performance of our approach significantly in terms of computational time and quality of results.

8.4.4 Parameter tuning

The different parameters used in our method are arranged in rows in Table 8.1. For each parameter, the table shows (in columns) its symbol, meaning, range of values, and the parameter value chosen in this chapter.

It is worthwhile to introduce a comment about the maximum number of generations. We tested our method for values of this parameter in the range $100 - 10000$ and found that the method converged in less than 5000 generations in all our executions. We finally set this parameter in 7000 generations, but to prevent wasting computation time without any improvement, we also set an additional termination criterion: the method stops if no further improvement of the solution is reached after 30 consecutive generations, even although the total number of generations is less than 7000. With this

Surface Example	*# data points*	*RMSE fitting error*	Υ *fitting error*
height-map	30×35	1.010108E-2 (mean) 7.896673E-3 (best)	1.072931E-1 (mean) 6.547533E-2 (best)
twisted shape	70×70	3.024972E-3 (mean) 1.400709E-3 (best)	4.483726E-2 (mean) 9.613742E-3 (best)
apple	40×40	7.0242041E-3 (mean) 4.701495E-3 (best)	7.894311E-2 (mean) 3.536649E-2 (best)

Table 8.2: The three examples in this chapter with their number of data points and the mean and best $RMSE$ and Υ errors.

additional criterion, the computation times improved significantly without penalizing the quality of the final solution.

With this choice of parameters, the bat algorithm is run iteratively for a given number of generations or until the convergence of the minimization of the error is eventually achieved. Final positions and velocities of the bats are computed and ranked according to our fitness function Υ. The position of the global best $(\mathbf{t}_*, \mathbf{s}_*, \mathbf{w}_*, \mathbf{\Lambda}_*)$ is taken as the final solution of our minimization problem.

8.5 Experimental Results

The method described in previous section has been applied to several examples. In this section we describe only three of them to keep the paper in manageable size. We think, however, that the examples reported here will be useful to readers to determine the good applicability of our method to this problem.

The examples correspond to three quite different free-form shapes: a height-map surface, a complicated twisted shape, and a closed organic shape. They have been primarily chosen because they reflect the great variety of shapes to which our method can be applied. First example exhibits several oscillations and changes of curvature, while the second and third examples show very complicated shapes which are not well suited for polynomial reconstruction. In this sense, these examples are a good benchmark to check the performance of our rational reconstruction approach. In all cases, data points are affected by noise of medium intensity and irregular sampling, so they actually replicate the usual conditions of real-world applications at full extent.

The three examples in our benchmark are arranged in rows in Table 8.2.

Figure 8.2: Application of our method to a height-map example: (top) cloud of data points; (middle) best fitting rational Bézier surface; (bottom) combination of the cloud of data points and their best fitting surface for better visualization.

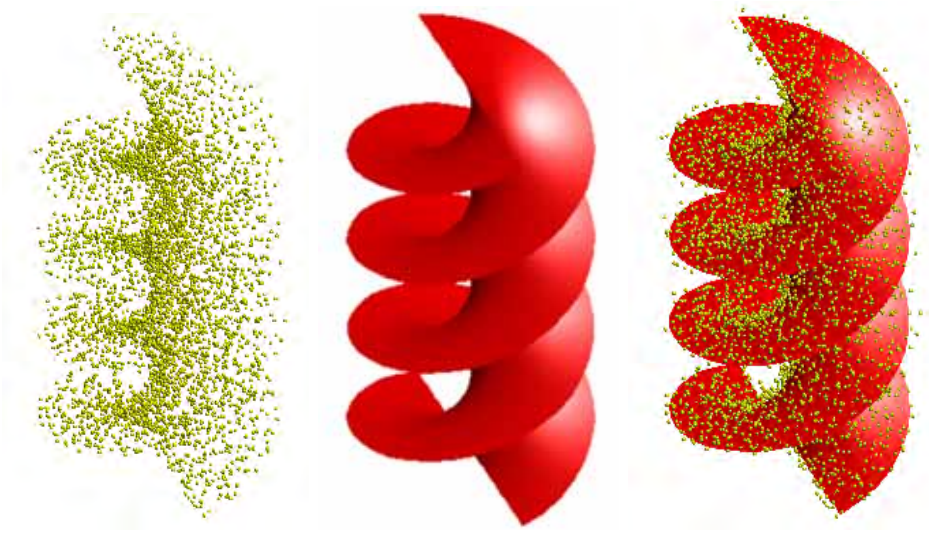

Figure 8.3: Application of our method to a twisted shape example: (left) cloud of data points; (middle) best fitting rational Bézier surface; (right) combination of the cloud of data points and their best fitting surface for better visualization.

For each example, the table shows (in columns) the number of data points used in our experiments along with the mean and the best fitting errors for Υ (given by Eq. (8.6)) and $RMSE$ (root-mean square error), given by: $RMSE = \sqrt{\frac{\Upsilon}{m\,n}}$. The mean error has been obtained from 50 independent executions of the algorithm, while the best value corresponds to the results of the best execution from the 50 runs.

First example corresponds to an academic shape representing a height-map surface. As such, it is a relatively simple shape that can arguably be reconstructed with a polynomial fitting surface. Yet, the shape is more difficult than it may seem at first sight, because it exhibits several oscillations and changes of curvature and it is strongly affected by measurement noise and irregular sampling.

These challenging features are clearly visible in Figure 8.2(top), where the noisy cloud of data points is displayed. The best fitting rational Bézier surface we obtained is displayed in Fig. 8.2(middle). We also combine both pictures in Fig. 8.2(bottom) for the sake of comparison. As the reader can see, the surface fits the cloud of data points quite well, even although the cloud of points of this example is strongly affected by measurement noise and irregular sampling.

Second example corresponds to a complicated open twisted shape, repre-

sented in Figure 8.3 (the description of the pictures in this figure is similar to the previous example - but placed from left to right - and will be omitted here to avoid redundant material). This example is very difficult to reconstruct by using a strictly polynomial surface. In this example, we consider a set of 70×70 data points, which are affected by a medium-intensity measurement noise. The best fitting surface is displayed in Figure 8.3(middle) and on the right along with the data points. As the reader can see from Table 8.2, we obtained values of order $10^{-2} \sim 10^{-3}$ for the mean error and of order 10^{-3} for the best error of the $RMSE$ and Υ fitting errors.

Third example corresponds to the closed shape of an apple, represented in Figure 8.4. In this example, we consider a set of 40×40 data points affected by noise of medium intensity. This example is particularly challenging because it contains a number of very difficult features. On one hand, it is a closed surface in vertical and horizontal directions, so it cannot be reconstructed by using only polynomial surfaces. In addition, it contains some turning points, where the surface is no longer differentiable. It is also greatly affected by irregular sampling. In fact, as the reader can see in Figure 8.4(top), the data points are not regularly distributed at all.

In spite of all these challenging features, the method performs very well, being able to replicate the original shape with high accuracy. Note, for instance, the good visual matching between the original data points and the approximating rational Bézier surface in Figure 8.4(bottom).

From the analysis of our numerical and graphical results we conclude that the method works pretty well, as it is able to replicate the original shape with very good accuracy for the examples in this paper. The numerical fitting errors are of order $10^{-1} \sim 10^{-3}$ for the three examples in our benchmark. These results are also confirmed visually: in all cases, there is a good visual fitting between the original points and the best approximating rational surface, even for clouds with a severe irregular pattern.

8.6 Main Conclusions

This method introduced in this chapter for surface reconstruction with rational Bézier surfaces is an extension of the method introduced in previous Chapter for the case of polynomial Bézier surfaces. In that chapter, we introduced a new method for polynomial Bézier surface parameterization from clouds of irregularly sampled data points. The approach was based on the application of the bat algorithm to compute a suitable parameterization for the optimal fitting surface. In this chapter, the previous method has been substantially extended to solve the problem of surface reconstruction with rational surfaces. The motivation for this work is that there are many important shapes (such as the quadrics) that cannot be adequately reconstructed

through polynomial surfaces. This problem can be solved by using rational surfaces, which provide a canonical representation of such shapes. However, this rational case becomes much more complicated because new variables (the weights) have also to be calculated in addition to the parameterization problem. In fact, the rational Bézier surfaces have not been usually applied to surface reconstruction so far. This is one of the major contributions of this chapter.

To address this issue, this chapter proposed a new approach in which the bat algorithm is sequentially applied to compute the data parameters and the weights. This process is performed iteratively by using the output of each bat algorithm as the input of the next one, and so on. Finally, the poles are computed through classical least-squares optimization.

Our method has been applied to a benchmark of three illustrative examples exhibiting challenging features. Our experimental results show that the method performs very well, and we can recover the underlying shape of very difficult surfaces with very good accuracy even for clouds of data points affected by strong measurement noise and irregular sampling, two very challenging features for many optimization techniques.

Figure 8.4: Application of our method to a closed organic shape: (top) cloud of data points; (middle) best fitting rational Bézier surface; (bottom) combination of the cloud of data points and their best fitting surface for better visualization.

Chapter 9

Memetic Approaches for Curve Reconstruction with Polynomial Bézier Curves

In the previous Chapter 5 we addressed the problem of curve parameterization for data fitting with polynomial Bézier curves by applying the bat algorithm. Although we obtained pretty good results on a number of examples, the method can still be further improved by considering a memetic approach, in which the global search bat algorithm is hybridized with a local search procedure to enhance the exploitation phase of the minimization process (see our discussion about the much-needed trade-off between the opposing features of exploration and exploitation for metaheuristic algorithms in Chapter 4 for details).

In this chapter we extend the research work carried out in Chapter 5 by considering two local search strategies: Luus-Jaakola and adaptive step size random search (ASSRS for short). In both cases, the adaptive and self-adaptive versions are considered, leading to four different memetic schemes. Then, a comparative analysis of our results on a benchmark for these four memetic schemes and our previous method is carried out. To allow fair comparison, we consider the same examples of the benchmark in previous Chapter 5. Our computer simulations show that the memetic approaches improve the efficiency of the bat algorithm method at different extent for all instances in our benchmark.

The structure of this chapter is as follows: in Section 9.1 we describe the local search strategies used in this chapter along with their adaptive and self-adaptive versions. The proposed memetic approaches introduced in this thesis are described in Section 9.2. Then, a comparative analysis of the four memetic approaches introduced in this chapter and the previous method in Chapter 5 is reported in Section 9.3. The paper closes with the main

conclusions of this contribution, reported in Section 9.4.

9.1 Local Search Procedures

Many real-world problems require to perform nonlinear optimization of very complicated objective functions. Furthermore, no analytical solution for these objective functions is commonly available, so we have to rely on numerical methods instead. In such cases, we usually apply a search algorithm, in which the objective function is first evaluated on a trial solution, even if it is far from the optimal solution; then, the quality of the solution is sequentially improved by perturbation of the initial solution according to some mathematical and/or stochastic procedures, formulas, rules, or heuristics until convergence (or a sufficiently good value) is attained.

We all know a myriad of methods to carry out this task, such as Newton and quasi-Newton methods, conjugate gradient methods, steepest descent, perturbation stochastic approximation, interior point methods, and so on. However, most of them impose strong constraints on the class of feasible functions, such as differentiability, continuity, and the like. Although they perform pretty well in many cases, such constraints are not generally met for some given problems. In such cases, we need to apply alternatives strategies for optimization, such as the *direct search* methods. The trademark of search direct methods is that no derivatives of the objective function are involved in the optimization process. As a consequence, they can be applied to unsupervised learning, where no initial information about the optimal solution is available. Clearly, the data fitting addressed in this thesis is one of such problems.

It is well-known that the success of bio-inspired techniques such as the bat algorithm relies largely on an adequate trade-off between two opposite terms: *exploration* and *exploitation*. The former accounts for *diversification*, i.e., to generate diverse solutions so as to explore the search space on a global scale of the solution domain, while the latter accounts for *intensification*, i.e., to concentrate the effort in a local region around a current good solution searching for better solutions in its neighborhood.

In our previous work in Chapter 5 for data fitting with polynomial Bézier curves we propose to apply the bat algorithm to solve the global search optimization problem. In other words, we focused on the exploration phase, while the exploitation can still be further improved with the addition of a local search procedure. This is the main contribution of the research work in this chapter.

There are many local search procedures available in the literature such as hill climbing, random optimization, random search, pattern search, iterated local search, and many others (see, for instance, the nice book on the

subject in [89]). Obviously, the choice of the local search method depends on the particular problem under study. There are, however, some local search procedures especially suited for continuous optimization problems. In this paper, we hybridize our bat algorithm method for global optimization introduced in Chapter 5 with two direct search methods, namely, Luus-Jaakola and ASSRS. They are described in next paragraphs.

9.1.1 Luus-Jaakola local search method

The Luus-Jaakola local search method (LJLS) is a heuristic for optimization of real-valued functions. This heuristics, firstly proposed by Luus and Jaakola in 1973 to solve nonlinear programming problems [132], starts with an initialization step, where random uniform values are chosen within the search space. To this aim, we compute the upper and lower bounds for each dimension. Then, a random uniform value in-between is sampled for each component. This value is added to the current position of the potential solution to generate a new candidate solution, which replaces the current one if and only if the fitness at the new position is better than that at the previous position. Otherwise, the sampling space is multiplicatively decreased by a factor, assumed to be constant in many instances. In practice, however, we found that it is better to consider an adaptive size for this factor, with the effect of speeding up the convergence to the steady state. This process is repeated iteratively. With each iteration, the neighborhood of the point decreases, so the procedure eventually collapses to a point.

This heuristics has been successfully applied in many fields, particularly in the domains of chemical engineering and optimal control. It can be proved that if the objective function of the optimization problem is C^2, the Luus-Jaakola method generates a sequence that has a convergent sub-sequence [75]. More interesting for our discussion is the fact that this heuristics is very well suited for optimization problems whose underlying functions are neither convex nor differentiable. Consequently, it can be successfully applied to our optimization problem without any limitation.

9.1.2 Adaptive step size random search (ASSRS)

The adaptive step size random search (ASSRS) is a member of the class of random search methods, a family of direct search methods based on sampling from a hypersphere surrounding the current position and then evolving iteratively to better positions in the search space [164]. The basic idea of random search is to initialize a random position in the search space, then sampling from the hypersphere of a given radius surrounding the current position to generate a new position. The value of the fitness function at this new posi-

tion is compared with the value at the current one. Whenever a better value is obtained, the new position is accepted; otherwise, the current one prevails and the process is repeated again. This strategy is applied iteratively until a steady-state (convergence or near-convergence) is attained.

Random search is a set of methods rather than a single method itself. Among the multiple variations of random search, the ASSRS has become very popular because of its ability to tune the hypershpere radius adaptively [177]. Instead of considering just one new position, two new potential positions are generated, one with the current nominal step size and another with a larger step size. The larger step size will become the new nominal step size should it lead to further improvement. If for a prescribed number of iterations neither of the steps leads to any improvement, the nominal step size is reduced according to a given factor.

A relevant issue is how to generate the sampling from the hypersphere so that it follows a uniform distribution. This issue is not trivial as the simple application of uniform random distributions on the variables in spherical coordinates returns a set of points bunched near the poles. Other popular methods for uniform sampling in the 3D sphere such as the Marsaglia method [137] can only be generalized to up to the 4-dimensional sphere but not beyond.

A feasible solution to this problem is the re-parameterization of the spherical coordinate variables. However, this method is difficult to achieve in high-dimensional spaces and it is also computationally expensive. Also, a Monte Carlo method can be applied, but its computational time increases drastically for large dimensions.

In this chapter, we applied a very efficient method consisting of the use of Gaussian random distributions (as many as the dimension of the hyperspace) modulated by the square root of the sum of the squares of the hyperspace coordinates. This method generates a uniform distribution over the hypersphere [137, 147]. It also provides the best performance in terms of accuracy for high dimensional systems while simultaneously exhibiting linear computational time.

9.1.3 Adaptive and self-adaptive variants for local search

As disussed in previous paragraphs, the two local search methods considered in this paper depend on some control parameters (mainly, the step size, the total number of iterations and the number of iterations for a fixed step size). In their simplest version, we can merely assume they take constant values and run the algorithm. However, this strategy is far from optimal in many cases.

In general, the choice of suitable values for the control parameters is

strongly problem-dependent. Therefore, the performance of local search strategies can improve dramatically when some kind of adaptive step size is considered. In this chapter, we consider two different options for the choice of control parameter values: adaptive tuning and self-adaptive (SA) tuning. In the former case, the parameter values are applied according to prescribed scaling factors. In the latter, we apply some kind of self-adaptive procedure for better parameter tuning. The combination of the two local search strategies with their adaptive and self-adaptive versions leads to four different memetic approaches: the adaptive and self-adaptive Luus-Jaakola methods (tagged as ALJLS and SALJLS, respectively), and the adaptive and self-adaptive step size random search (tagged as ASSRS and SASSRS, respectively).

In this chapter, the following settings for the adaptive versions are used: the decreasing multiplicative factor ζ for the Luus-Jaakola method is set to 0.95, while the larger radius R and smaller radius r for the ASSRS method are set to 1.05ρ and 0.95ρ respectively, where ρ denotes the current radius value. In both methods, the total number of iterations for the local search method is denoted by τ, and the number of iterations for any fixed value of the control parameter ζ or r (for Luus-Jaakola and ASSRS respectively) is denoted by δ. For this work, the following values are taken: $\tau = 100$, $\delta = 25$. This choice is fully empirical as it worked very well in our computer simulations.

Regarding the self-adaptive versions, the control parameters are tuned according to some evolution equations. For the Luus-Jaakola method, we consider a perturbation law driven by:

$$\zeta(t) = \zeta(t-1)\left(1 - \frac{t-1}{\tau}\right)^{0.1}$$

with $\zeta(1) = 0.99$, where t denotes the current iteration. The meaning of this formula is that we start with a high multiplicative value, e.g. $\zeta = 0.99$, which corresponds to a system where the search space is barely reduced, leading to extensive exploration, and gradually reduce its size to lower values near to 0, where the system performs intensive exploitation around the local optima. This process is repeated iteratively. With each iteration, the neighborhood of the point decreases, so the procedure eventually collapses to a point. Note that many other functions can be used instead of this proposal as long as they keep a similar behavior pattern.

For the SASSRS, we consider two perturbation rules for the larger, R, and the smaller radius, r, as follows:

$$R(t) = R(t-1)\left(1 + \frac{\tau - t}{50\tau}\right)^{1/4}$$

and

$$r(t) = r(t-1)\left(1 - \frac{t-1}{2\tau}\right)^{1/3}$$

respectively, with $R(1) = r(1) = \rho$. The meaning of these formulas is that, starting with two initial values equal to the nominal value, R (resp., r) is iteratively increased (decreased) at relatively fast pace at the beginning and then at slower pace as time increases to promote exploration (exploitation) of the search space over the iterations. Once again, these formulas can be replaced by others following a similar pattern.

9.2 The Proposed Method

As an extension of our previous method described in Chapter 5, our approach here is comprised of two main stages. In next paragraphs we describe each of these steps in detail.

9.2.1 First stage: global search of the optimal solution

The first stage of our method is focused on the global optimization step. In it, we apply the bat algorithm to determine suitable parameter values for the least-squares minimization of functional E according to (5.6). To this aim, each bat, representing a potential solution, corresponds to a parametric vector $\mathcal{T}_j = (t_1^j, t_2^j, \ldots, t_m^j) \in [0,1]^m$, $(j = 1, \ldots, \mathcal{P})$ and the $\{t_i^j\}_{i=1,\ldots,m}$ are strictly increasing parameters. These parametric vectors are initialized with random values and then sorted.

We then apply our bat algorithm method with the same parameters values described in Chapter 5. These parameter values are fully reported in Table 5.1 and their choice is discussed in Section 5.4.2. This procedure yields new positions and velocities of the bats representing the potential solutions of this optimization problem. The process is performed iteratively for a given number of generations, until the convergence of the minimization of the error is eventually achieved.

9.2.2 Second stage: local search refinement

In the second stage, the best solution obtained from the first stage is iteratively refined by application of the local search procedures described in Section 9.1. But instead of generating just a new perturbed solution from the current one, we consider a collection of $\mu.\eta$ new solutions, where η represents the number of free variables of the optimization problem and μ is a positive real multiplicative factor with the meaning of increasing ($\mu > 1$) or

decreasing ($\mu < 1$) the number of clones of the best solution attained so far and subjected to local perturbation.

Note that increasing the value of μ also increases the computational complexity of every iteration, but could arguable reduce the number of iterations required for convergence, while decreasing this value has exactly the opposite effect. Therefore, an adequate trade-off between both cases is generally required. However, the determination of an optimal value for this parameter is once again problem-dependent and must be performed empirically.

In this work, we take $\mu = 4$ as it provided a reasonable balance between these two competing factors, although other values close to this one might also be acceptable. The local search procedures are then applied for the number of iterations indicated in Section 9.1.3. At each iteration step, the pool of solutions is ranked according to our fitness function and the best one is preserved unaltered for the next iteration. Once the iteration loop is over, the best global solution achieved so far is selected as the final solution of our problem.

9.3 Experimental Results

The four memetic approaches described in previous section have been applied to some illustrative examples corresponding to 2D and 3D curves. Then, these results have been compared to those obtained with other alternative methods. To allow a fair comparison, in this discussion we consider the same examples of the benchmark in the previous paper, shown in Table 9.1, where the three examples selected are arranged in rows. For each example, the table shows (in columns) the number of data points used in our experiments along with some other interesting features: whether or not the curve includes any non-differentiable point (such as cusps) or self-intersections, which usually represent challenging features for data fitting techniques. It also reports the dimensionality of both the curve (2D/3D) and the optimization problem, the latter given by the number of degrees of freedom, DOFs for short (i.e. the number of variables to be minimized).

These examples have been primarily chosen to reflect the diversity of situations our method can be applied to. First example corresponds to a planar closed curve called epitrochoid, which has several self-intersections; second example shows a curve called piriform, a planar closed curve with a cusp; and last example corresponds to a 3D closed curve called the Viviani curve.

Table 9.2 reports our numerical results. Once again, the three examples are arranged in rows in first column. For each example and each method applied, we compute:

Example number:	*# data points*	*C^0 points*	*Self-int. curve*	*2D/3D curve*	*Total DOFs*
#1: Epitrochoid	300	×	✓	2D	332
#2: Piriform	100	✓	×	2D	110
#3: Viviani	200	×	✓	3D	214

Table 9.1: Benchmark used in this paper along with the main features of each example.

- the mean and best error of the functional E according to Eq. (**??**), represented by E^m and E^b, respectively, and
- the mean and best value of the root-mean square error, given by:

$$\Lambda = \sqrt{\frac{E}{m}}$$

  represented by Λ^m and Λ^b, respectively.

The mean error has been obtained as the mean fitting error value from 50 independent executions of the algorithm, while the best value corresponds to the results of the best execution from the 50 runs. The different methods considered in this comparison are arranged in columns. The methods are (from third to ninth column respectively): the arc-length parameterization (denoted by *AL*), the firefly algorithm (FFA), the original bat algorithm applied in Chapter 5, and the four bat algorithm memetic approaches, denoted by bat-ALJLS, bat-SALJLS, bat-ASSRS, and bat-SASSRS, respectively.

A simple visual inspection of Table 9.2 shows that the four memetic approaches improve the previous bat algorithm for all instances in our benchmark. This is not very surprising, since the local search has been designed to refine the solution obtained from the single bat algorithm.

We also remark, however, that the improvement is not dramatic, a clear indication that the previous best solution was already pretty good. In fact, we noticed that the new best solutions are virtually indistinguishable with respect to the previous ones from a graphical point of view. This is the reason we did not include any graphical output in this paper. As a consequence, we have to rely exclusively on our numerical results for this comparative work.

From Table 9.2, we obtained the following conclusions:

1. In general, the new methods perform very well, with average fitting errors of order $10^{-4} \sim 10^{-5}$ (mean) to $10^{-5} \sim 10^{-6}$ (best) for the E error, with Λ of order $10^{-3} \sim 10^{-4}$ (mean) to 10^{-4} (best) for the three examples in our benchmark,

2. the four memetic algorithms improve previous results obtained with the single bat algorithm,

3. Of the four memetic approaches, the self-adaptive versions perform better than the adaptive ones. This shows the ability of the self-adaptive schemes to effectively capture the inherent dynamical behavior of the optimization problem and select suitable values accordingly.

4. At its turn, the ASSRS method outperforms the Luus-Jaakola method for this particular problem.

In conclusion, the memetic methods discussed in this paper outperform previous approaches for this optimization problem, but not dramatically. However, they are simple to apply, easy to compute and do not add much extra time to our previous computations. Based on these appealing features, it is advisable to include them in future works in the field.

9.4 Conclusions

In this paper we extended our previous method developed for our curve reconstruction problem described in Chapter 5 of this thesis to address the problem of computing the fitting curve from a cloud of data points. The previous approach, based on the application of the bat algorithm to compute an optimal parameterization of the Bézier curve that fits the data points better in the least-squares sense, has been improved by hybridizing it with local search methods for better performance. The rationale of this approach is that the bat algorithm performs global search, which can be further refined with a local search procedure to enhance the exploitation phase of the optimization process.

In this chapter two local search strategies have been considered: Luus-Jaakola and adaptive step size random search (ASSRS). In both cases, the adaptive and self-adaptive versions are considered, leading to four different memetic schemes. A comparative analysis of our results on the previous benchmark for these four memetic schemes and our previous method has been carried out.

Our experimental results on the chosen benchmark show that the memetic approaches improve the efficiency of the previous method at different extent for all instances in our benchmark. In this regard, the main observations from our computer experiments are that the self-adaptive version outperforms the adaptive one in all cases, and that the ASSRS version outperforms the Luus-Jaakola approach on our benchmark.

Finally, based on the good properties of these local search methods (simple to apply, easy to compute, and do not require too much time to be

computed), we conclude that it is advisable to include them in the future developments in the field.

Curve:	*Error:*	*AL*	*FFA*	*Bat alg.*
#1	E^m	5.342768e-3	4.526912e-4	4.065316e-4
	E^b	9.017334e-4	1.034348e-4	7.254377e-5
	Λ^m	4.220097e-3	1.228401e-3	1.164089e-3
	Λ^b	1.733371e-3	5.871819e-4	4.917444e-4
#2	E^m	5.532432e-4	3.187544e-5	3.206772e-5
	E^b	5.113864e-5	1.283742e-6	1.378186e-6
	Λ^m	2.352112e-3	5.645833e-4	5.662837e-4
	Λ^b	7.151128e-3	1.113302e-4	1.173962e-4
#3	E^m	8.174659e-4	2.186931e-4	2.192432e-4
	E^b	9.364587e-5	4.624328e-5	3.527668e-5
	Λ^m	2.021714e-3	1.045689e-3	1.047003e-3
	Λ^b	6.842728e-4	4.808496e-4	4.199802e-4

Curve:	*Error:*	*Bat-ALJLS*	*Bat-SALJLS*	*Bat-ASSRS*	*Bat-SASSRS*
#1	E^m	4.013983e-4	3.914077e-4	3.974936e-4	3.859365e-4
	E^b	7.198663e-5	7.142691e-5	7.173112e-5	7.085943e-5
	Λ^m	1.156717e-3	1.142231e-3	1.151077e-3	1.134219e-3
	Λ^b	4.898524e-4	4.879443e-4	4.889823e-4	4.860021e-4
#2	E^m	3.191537e-5	3.185641e-5	3.189336e-5	3.164710e-5
	E^b	1.385526e-6	1.354716e-6	1.381075e-6	1.239567e-6
	Λ^m	5.649368e-4	5.644148e-4	5.647420e-4	5.625575e-4
	Λ^b	1.177083e-4	1.163922e-4	1.175191e-4	1.111335e-4
#3	E^m	2.189953e-4	2.186433e-4	2.190122e-4	2.171532e-4
	E^b	3.495816e-5	3.526701e-5	3.511887e-5	3.131206e-5
	Λ^m	1.046411e-3	1.045570e-3	1.046451e-3	1.042001e-3
	Λ^b	4.180798e-4	4.199226e-4	4.190397e-4	3.956677e-4

Table 9.2: Fitting errors for the examples used in this paper (in rows): E error and Λ for the mean and best results from 50 executions for the arc-length, firefly algorithm, bat algorithm and the four bat algorithm memetic approaches (bat-ALJLS, bat-SALJLS, bat-ASSRS, bat-SASSRS) in this chapter (in columns).

Part IV

CONCLUSIONS AND FUTURE WORK

Chapter 10

Conclusions

The aim of this chapter is twofold: on one hand, it summarizes the main contributions of the research work carried out for this doctoral thesis. They are described in Section 10.1. The chapter also describes briefly the publications that this thesis has produced so far along with some papers that are currently under review in some scientific journals. These publications are reported in Section 10.2.

10.1 Main Contributions of the Thesis

This section outlines some of the most important contributions of this doctoral thesis.

The main contribution of this thesis is the application of a powerful metaheuristic, the bat algorithm, to solve the problem of free-form shape reconstruction by using global-support functions. In the research work of this thesis, we have developed a methodology to reconstruct the underlying shape (either as a curve or as a surface) of a cloud of noisy data points. Such a methodology has been applied to derive suitable methods to address the cases of polynomial and rational Bézier curves and surfaces, described in Chapters 5 to 9 of this thesis.

The methodology developed in this thesis has been tested on four different benchmarks associated with the different types of curves and surfaces considered in this work. The computational results that we have obtained on the examples of each benchmark confirm that our approach works very

well, as we have been able to capture the inherent shape of the clouds of data points with high accuracy.

We have extended our results for polynomial shapes to the rational case. Most of the research works on shape reconstruction focus on the polynomial case only, since this is the most standard case in the literature. However, there are interesting shapes (such as the conics for curves and the quadrics for surfaces) that cannot be represented faithfully by using the polynomial formalism. In this thesis, we have extended our methods to the rational case for both curves and surfaces.

Such extension is also valuable because the methods for polynomial shapes cannot be extended to the rational case in a natural way. The rational shapes are more complicated not only because we have some additional variables to deal with (the weights) but mainly because the different sets of unknowns (data parameters, control points, and weights) are related each other in a highly nonlinear and complicated way. For instance, the locations of the control points affect the values of the weights, the parameterization affects the control points and weights, they at their turn affect the parameterization, and so on. Solving this problem is not just a matter of applying the bat algorithm on each set of unknowns as if they were completely independent. Instead, we have to develop sophisticated strategies to compute the different parameters taking into account their complex intertwined relationships. The iterative strategy developed for the rational Bézier surfaces in Chapter 8 is an illustrative examples of this complex approach.

A very important feature of our approach is its good applicability to the case of clouds of data points affected by measurement noise, irregular sampling and other artifacts. All examples in our benchmarks are affected by these problems, which typically appear in real-world scenarios. Our methods perform well even under these very unfavourable conditions. This means that they can be applied to real-world problems without any further improvement or modification.

This is a very remarkable feature of our approach. Many papers intro-

duce new methods that only work properly for clouds of data points that are not affected by these problems. Although still valuable, it is clear that those methods can be very limited, as they can only be applied to academic examples in theoretical works, but they fail when applied to real-world situations. On the contrary, our methods have been tested under these problematic conditions. In this sense, our methods are more general that many previous approaches limited to ideal conditions that never happen in practice.

This thesis reports the first case of application of the bat algorithm to shape reconstruction. To the best of our knowledge, no previous work has applied the bat algorithm so far in the context of geometric modeling or computer graphics, a clear evidence of the originality of this research work.

As we mentioned in some parts of the thesis, this fact is not accidental at all; the main reason is that we are dealing with a very difficult problem. In fact, this problem has remained open for several decades. In spite of the high relevance of this problem in many applied domains (see our discussion in Chapter 3), the problem is still unsolved with the exception of some particular cases.

Our methods not only solve the shape reconstruction problem for curves and surfaces. They can also be applied to other tasks such as curve and surface parameterization. In fact, we showed in Chapters 5 and 7 that the curves and surface reconstruction for polynomial shapes can be transformed into the problem of parameterization.

Indeed, although the emphasis of this thesis is on shape reconstruction, the methodology described here can also be applied to many other different problems. This fact can be better understood when we consider that we are applying a metaheuristics (the bat algorithm) designed to solve difficult continuous optimization problems. As expected, the same methodology can also be applied to other optimization problems, such as, for instance, curve and surface parameterization. This is one of the most promising features of

our method, as it opens the door to other future developments in the field.

Another contribution of this thesis is the development of some memetic approaches combining the bat algorithm methodology with some local search methods for further efficiency and higher accuracy. In Chapter 9 of this thesis we introduced four memetic approaches based on the hybridization of the bat algorithm with two local search methods (Luus-Jaakola and adaptive step size random search). We showed that this new feature improves the performance of our proposal on the examples of the benchmark.

In fact, the experimental results have been obtained on the same benchmark used in Chapter 5 to test the original method for polynomial Bézier curves. This means that these memetic approaches effectively improve the performance by addition of the local search on top of the global optimization scheme.

The developed methodology is very general. In our methods, we do not make any assumption on the properties of the fitting curve or surface (such as continuity, differentiability, or the like). Actually, we do not assume any knowledge about the problem beyond the data points.

This feature is very important, because it means that the method can be applied to any cloud of points without any limitation. Similarly, it can be applied to many different problems arising in industrial and applied environments in many different fields (such as those described in Chapter 3, but not exclusively).

10.2 Publications of the Thesis

The contributions indicated in the previous section have led to some international publications. We found this process of publishing our results very useful because it allowed, on one hand, to improve our work by incorporating the comments and suggestions from other experts in the field into our work, and, on the other hand, to have a better feedback to determine the level or

originality of our proposal as well as its technical quality.

In this section, we provide a brief reference about every publication. For the sake of clarity, each publication will be organized in the corresponding chapter related to its technical content.

Chapter 5:

Iglesias, A., Gálvez, A., Collantes, M.: *"Bat Algorithm for Curve Parameterization in Data Fitting with Polynomial Bézier Curves"*. *Proc. of Int. Conference on Cyberworlds, CW 2015*, Visby (Sweden). IEEE Computer Society Press, Los Alamitos CA (2015) 107-114.

This conference is very prestigious in the fields of computer graphics and geometric modeling. The conference has been sponsored by the two most important scientific societies in computer graphics *ACM Siggraph* and *Eurographics*, and the prestigious computer science society *IFIP - Technical Group on Computer Graphics*. The edition of 2015 was held in Visby (Sweden), in October 2015. The paper was reviewed by three experts and received very good comments. Furthermore, the paper was selected for one of the special issues of the conference (the publication associated with Chapter 9 below).

The proceedings of *Cyberworlds'2015* conference are published by *IEEE Computer Society Press*. *Cyberworlds'2015* is a **ERA-B conference**.

Chapter 6:

Iglesias, A., Gálvez, A., Collantes, M.: *"Global-Support Rational Curve Method for Data Approximation with Bat Algorithm"*. *Proc. of Int. Conference Artificial Intelligence and Applications, AIAI'2015*, Bayonne (France). IFIP Advances in Information and Communication Technology, **458** (2015) 191-205.

This conference is very prestigious in the field of artificial intelligence. The conference is sponsored by *IFIP - Technical Group on Artificial Intelligence* The edition of 2015 was held in Bayonne (France), in September 2015. The paper was reviewed by three experts and received very good comments.

The proceedings of *CW'2015* conference are published by *Springer-Verlag*, in its *IFIP Advances in Information and Communication Technology* series.

Chapter 7:

Iglesias, A., Gálvez, A., Collantes, M.: *"A Bat Algorithm for Polynomial Bézier Surface Parameterization from Clouds of Irregularly Sampled Data Points". Proc. of Int. Conference Natural Computation 2015, ICNC'2015*, Bayonne (France). Zhangjiajie (China). IEEE Computer Society Press, Los Alamitos CA (2015) 1034-1039.

This conference is a classical one in the field of artificial intelligence and soft computing. The conference is sponsored by *IEEE Computer Society* and *IEEE Computational Intelligence*. The edition of 2015 was held in Zhangjiajie (China), in August 2015.

The paper was reviewed by three experts and received very positive comments. Furthermore, the paper was selected for one of the special issues of the conference (the publication associated with Chapter 8 below).

The proceedings of *ICNC'2015* conference are published by *IEEE Computer Society Press*.

Chapter 8:

Iglesias, A., Gálvez, A., Collantes, M.: *"Iterative Sequential Bat Algorithm for Free-Form Rational Polynomial Bézier Surface Reconstruction". Int. Journal of Parallel Programming*, (special issue of ICNC'2015 conference). (*Submitted on Nov. 30th 2015, currently under review*).

This paper corresponds to an extended version of the paper accepted for ICNC'2015 conference indicated above. After the conference, the best papers were selected for a special issue in this journal. The papers were then subjected to a first review round in January, and will be subjected to a second (final) review of the modifications made on the previous version of the paper. Notification is expected for June 2016.

The *Int. Journal of Parallel Programming* (Springer) is a JCR-indexed journal.

Chapter 9:

Iglesias, A., Gálvez, A., Collantes, M.: *"Four Adaptive Memetic Bat Algorithm Schemes for Bézier Curve Parameterization"*. *Transactions on Computational Science*, (special issue of CW'2015 conference). (*Submitted on Dec. 31st. 2015, currently under review*).

This paper corresponds to an extended version of the paper accepted for CW'2015 conference indicated above. After the conference, the best papers were selected for a special issue in this journal. The papers will be subjected to a new review round. Notification is expected for July 2016.

Chapter 11

Future Work

In spite of all research work carried out so far and reported in previous chapters, the curve and surface reconstruction problems are still far from being solved in all their generality. The primary reason is that we are facing two very difficult optimization problems. We should not expect them to be solved in just a shot. Although the field has witnessed significant progresses during the last decades, there is still a long way to walk.

In this chapter we discuss some ideas for further improvement of the research work described here. They can arguably open the door for new lines of research in the field for coming years and help other researchers to continue this task.

11.1 Emerging Trends

One of the major issues in the field of metaheurisitcs for optimization is the determination of the best metaheuristic method to be applied to this problem. The "no free lunch" theorem taught us that we should not expect to find a universal "best method" for all optimization problems, as any two methods perform equivalently on average for all problems [198].

However, this property is not as limiting as it could appear, because it talks about the average. In this sense, we might arguably be able to identify a method that outperforms any other for a specific given problem. Clearly, it would be very helpful to be able to classify different metaheuristics in terms of their performance for this reconstruction problem. A sub-product of this approach would be to determine the "best metaheuristics" for this particular problem, provided that such a best method really exists. In our opinion, the application of these ideas to the fields of curve and surface reconstruction is one of the emerging trends for the next few years.

To that purpose, a first valuable step would be the creation of a reliable, standardized benchmark for the field. The primary goal of this benchmark

is to facilitate a comparative analysis among the different metaheuristics, something that is not currently available to researchers and practitioners. Obviously, the second step in this process is the comparative analysis itself. This is not a trivial task, as it requires to carry out many different tasks, including the difficult one of parameter tuning.

Indeed, another challenging issue when working with metaheuristic algorithms is their inherent problem-dependent nature. It means that the parameter setting required for a method to work optimally is not universal either, but specialized for each particular problem. In this sense, the other factors being the same, the parameter setting leading to optimal performance for a given problem might be completely inadequate for other problems. At this moment, it is not even clear which are the best sets of parameters for a given metaheuristics to perform optimally for this problem.

Other exciting future line of research is the extension of this research to other families of curves and surfaces not addressed yet in the literature. Although there are very strong reasons to deal with free-form parametric functions, they are not the only ones with interest in applied and industrial fields. However, little has been done so far in this regard. We anticipate an increasing interest in this area for coming years.

Finally, there is still a promising field in the possible applications of these methods in many areas. With the popularization of sensor and capturing technologies such as 3D laser scanners and the wide availability of affordable 3D printers, we can envision a future of mass customization of products as a growing trend. As the complexity of shapes of customer products is increasing, more sophisticated methods for shape reconstruction will be required. At that time, we can expect a new golden era in the development of these reconstruction techniques.

11.2 Some Future Lines of Research

Focusing specifically in the research topics included in this thesis, this research work can be extended in many different ways.

11.2.1 Future work for curve reconstruction

Regarding the curve reconstruction problem, our future work includes:

- The extension of our methodology to other families of curves with interest in design and manufacturing, such as the B-spline curves and NURBS.
- We are also interested to analyze its application to some industrial processes and other interesting real-world problems in different fields.

- the definition of a reliable, standardized benchmark for the field. This benchmark should facilitate the comparative analysis among the different metaheuristics applied to this particular optimization problem.
- A comparative analysis with other alternative approaches on a standardized benchmark (when available) is also part of our future goals.
- A theoretical analysis about the convergence of our methods for polynomial and rational curves would also be very welcome.
- Finally, some other local search methods could also be arguably considered for our memetic approach described in Chapter 9.

11.2.2 Future work for surface reconstruction

Regarding the surface reconstruction problem, our future work includes:

- The extension of our methodology to other families of surfaces with interest in design and manufacturing, such as the B-spline surfaces and NURBS.
- We are also interested to analyze its application to some industrial processes and other interesting real-world problems in different fields.
- A comparative work of our methods with other alternatives that could appear in the literature of the field in coming years.

Part V

APPENDIX

Addenda

This apendix includes four addenda about the conventions for names, the general notation of this thesis, and the definition and some properties of the dot product (also called scalar product) and the cross product (also called vector product).

Addendum A: Conventions for Names

The following conventions for names will be used throughout this thesis:

- ACO: *A*nt *C*olony *O*ptimization.
- ASSRS: *A*daptive *S*tep *S*ize *r*andom *S*earch.
- CAGD: *C*omputer *A*ided *G*eometric *D*esign.
- CAD: *C*omputer *A*ided *D*esign.
- CAM: *C*omputer *A*ided *M*anufacturing.
- CAE: *C*omputer *A*ided *E*ngineering.
- FFA: *F*irefly *A*lgorithm.
- GA: *G*enetic*A*lgorithm.
- NURBS: *N*on *U*niform *R*ational *B*-*s*pline.
- ODE: *O*rdinary *D*ifferential *E*quation.
- PSO: *P*article *S*warm *O*ptimization.
- RMSE: *R*oot *M*ean *S*quare *E*rror.
- SA: *S*imulated *A*nnealing.

Addendum B: General Notation

Unless otherwise stated, the following notation will be used throughout this thesis:

$\mathbf{V}$	All vectors are denoted in bold
$\mathbf{V}^{\perp}$	Orthogonal vector to vector $\mathbf{V}$
$\|\|.\|\|$	Modulus of a vector
$\|.\|$	Determinant of a square matrix
$\mathbf{A}^{T}$	Transpose of a matrix $\mathbf{A}$
$vec(\mathbf{M})$	Vectorization of a vector or matrix $\mathbf{M}$
"."	Dot product / Scalar product
$\times$	Cross Product / Vector product
$\mathbf{C}(t)$	Parametric curve of variable t
$\mathbf{C}'$	Tangent vector to the curve $\mathbf{C}'$
$\mathbf{S}(u, v)$	Parametric surface of variables (u, v)
$\mathbf{S}_u$	First partial derivative of surface $\mathbf{S}$ with respect to variable u
$\mathbf{S}_{uv}$	second partial derivative of surface $\mathbf{S}$ with respect to u, v
$\vec{\nabla}\Phi$	Gradient vector of scalar function Φ
$\oplus$	Entry-wise multiplication (in Chapter 4)
$\odot$	Tensor product of vectors (in Chapter 7)
$\langle . \rangle$	Vectorization of a vector or matrix (in Chapter 8)

Addendum C: Dot Product (also Scalar Product)

in this thesis, we denote by"." the *dot product or scalar product* of two vectors in the Euclidean space $\mathbb{R}^n$. Given two vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ the scalar product returns the scalar number:

$$\mathbf{a}.\mathbf{b} = \sum_{i=1}^{n} a_i b_i.$$

The functions*norm* $||.||$ and *distance* $d(.,.)$ in $\mathbb{R}^n$ are defined by:

$$||\mathbf{a}|| = \sqrt{\mathbf{a}.\mathbf{a}}$$

and

$$d(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}||$$

respectively. These functions hold the following properties:

$$\mathbf{a}.\mathbf{b} = \mathbf{b}.\mathbf{a}$$

$$(\mathbf{a} + \mathbf{b}).\mathbf{c} = \mathbf{a}.\mathbf{c} + \mathbf{b}.\mathbf{c}$$

$$(\lambda\mathbf{a}).\mathbf{b} = \lambda(\mathbf{a}.\mathbf{b}) = \mathbf{a}.(\lambda\mathbf{b})$$

$$||\lambda\mathbf{a}|| = |\lambda|\,||\mathbf{a}||$$

where $\lambda \in \mathbb{R}$.

The angle θ between two vectors $\mathbf{a}$ y $\mathbf{b}$ can be computed as:

$$cos(\theta) = \frac{\mathbf{a}.\mathbf{b}}{||\mathbf{a}||\,||\mathbf{b}||}, \qquad 0 \leqslant \theta \leqslant \pi.$$

Note that this means that:

$$-1 \leqslant \frac{\mathbf{a}.\mathbf{b}}{||\mathbf{a}||\,||\mathbf{b}||} \leqslant 1$$

Addendum D: Cross Product (also Vector Product)

To define the cross product (or vector product) it is convenient to introduce the following notation:

$$\begin{aligned}\mathbf{i} &= (1,0,0)\\ \mathbf{j} &= (0,1,0)\\ \mathbf{k} &= (0,0,1)\end{aligned}$$

Every vector $\mathbf{a} \in \mathbb{R}^3$ can be written as:

$$\mathbf{a} = (a_1, a_2, a_3) = a_1\,\mathbf{i} + a_2\,\mathbf{j} + a_3\,\mathbf{k}.$$

Given two vectors $\mathbf{a} = (a_1, a_2, a_3)$ y $\mathbf{b} = (b_1, b_2, b_3)$ in $\mathbb{R}^3$, their *cross product* is given by:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

that is:

$$\mathbf{a} \times \mathbf{b} = (a_2b_3 - a_3b_2)\mathbf{i} - (a_1b_3 - a_3b_1)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}.$$

The cross product holds the following properties:

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$$

$$(\mathbf{a} + \mathbf{b}) \times \mathbf{c} = \mathbf{a} \times \mathbf{c} + \mathbf{b} \times \mathbf{c}$$

$$(\lambda\mathbf{a}) \times \mathbf{b} = \lambda(\mathbf{a} \times \mathbf{b}) = \mathbf{a} \times (\lambda\mathbf{b})$$

$$(\mathbf{a} \times \mathbf{b}).(\mathbf{c} \times \mathbf{d}) = (\mathbf{a}.\mathbf{b})(\mathbf{c}.\mathbf{d}) - (\mathbf{a}.\mathbf{d})(\mathbf{b}.\mathbf{c})$$

The following properties relate the dot product and the cross product:

$$||\mathbf{a} \times \mathbf{b}||^2 = ||\mathbf{a}||^2||\mathbf{b}||^2 - (\mathbf{a}.\mathbf{b})^2$$

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}.\mathbf{c}).\mathbf{b} - (\mathbf{a}.\mathbf{b})\mathbf{c}$$

$$(\mathbf{a} \times \mathbf{b}).\mathbf{c} = \mathbf{a}.(\mathbf{b} \times \mathbf{c})$$

for all $\lambda \in \mathbb{R}$.

Part VI

BIBLIOGRAPHY

Bibliography

[1] Ahn, C., An, J., Yoo, J.: Estimation of particle swarm distribution algorithms: combining the benefits of PSO and EDAs. *Information Sciences* (2010), doi: 10.1016/j.ins.2010.07.014

[2] Alhanaty, M., Bercovier, M.: Curve and surface fitting and design by optimal control methods. *Computer-Aided Design* **33** (2001) 167-182.

[3] Alvino, C.V., Yezzi, A.J.: Tomographic Reconstruction of Piecewise Smooth Images. In: Proceedings of Computer Vision and Pattern Recognition - CVPR'04. *IEEE Computer Society Press*, Los Alamitos, CA, Vol. 1 (2004) 576-581

[4] Bajaj, C., Bernardini, F., Xu, G.: Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans, *Proc. SIGGRAPH'95* (1995) 109-118

[5] Bajaj, C., Coyle, E.J., Lin, K.N.: Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing*, **58** (1996) 524-543

[6] Barhak, J., Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. *IEEE Trans. on Visualization and Computer Graphics*, **7**(1) (2001) 1-16.

[7] Barnhill, R.E.: *Geometric Processing for Design and Manufacturing.* SIAM, Philadelphia (1992).

[8] Beielstein, T., Mehnen, J., Schnemann, L., Schwefel, H.P., Surmann, T., Weinert, K., Wiesmann, D.:. Design of evolutionary algorithms and applications in surface reconstruction. In: Schwefel, H.P., Wegener, I., Weinert, K. (editors): Advances in Computational Intelligence - Theory and Practice, Springer, Berlin (2003) 164-193

[9] Bolle, R.M., Vemuri, B.C.: On three-dimensional surface reconstruction methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **13**(1) (1991) 1-13

[10] Brunnett, G., Kiefer, J.: Interpolation with minimal-energy splines. *Computer-Aided Design*, **26**(2) (1994) 37-144.

[11] Burchard, H.G.: Splines (with optimal knots) are better. *Applicable Analysis,* **3** (1974) 309-319.

[12] Castillo, E., Iglesias, A.: Some characterizations of families of surfaces using functional equations. *ACM Transactions on Graphics*, **16**(3) (1997) 296-318.

[13] Castillo, E., Iglesias, A., Ruiz-Cobo, R.: *Functional Equations in Applied Sciences*. Elsevier Science, Amsterdam (2005)

[14] De Castro, L.N., Timmis, J.: *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London (2002).

[15] De Castro, L.N., Von Zuben, F.J.: *Artificial Immune Systems: Part I - Basic Theory and Applications*, Technical Report-RT DCA 01/99 (1999).

[16] De Castro, L.N., Von Zuben, F.J.: The Clonal Selection Algorithm with Engineering Applications. In: *Proceedings of GECCO00, Workshop on Artificial Immune Systems and their Applications*, Las Vegas, USA, (2000) 36-37.

[17] De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, **6**(3) (2002) 239-251.

[18] Chang, F., Huang, H.: A refactoring method for cache-efficient swarm intelligent algorithms. *Information Sciences* (2010), doi:10.1016/j.ins.2010.02.025

[19] Chang, J. Shi, P.: Using investment satisfaction capability index based particle swarm optimization to construct a stock portfolio. *Information Sciences* (2010), doi:10.1016/j.ins.2010.05.008

[20] Cobo, A., Gálvez, A., Puig-Pey, J., Iglesias, A., Espinola, J.: Bio-inspired metaheuristic methods for fitting points in CAGD. *Int. Journal of Computer Information Systems and Industrial Management Applications*, **1** (2009) 36-47.

[21] Cosido, O., de José, J., Piquero, D., Iglesias, A., Sainz, E.: Implementation and Deployment of Geographical Information System Services in the Municipality of Santander. *Proc. of ICCSA 2011*, IEEE Computer Society Press, Los Alamitos CA (2011) 267-270.

[22] Cosido, O., Loucera, C., Iglesias, A.: Automatic calculation of bicycle routes by combining metaheuristics and GIS techniques within the framework of smart cities. *Proc. of Smartmile 2013*, IEEE Computer Society Press, Los Alamitos CA (2013) 259-266.

[23] Cosido, O., Iglesias, A., Gálvez, A., Catuogno, R., Campi, M., Terán, L., Sainz, E.: Hybridization of convergent photogrammetry, computer vision, and artificial intelligence for digital documentation of cultural heritage. A case study: the Magdalena Palace. *Proc. of Cyberworlds 2014*, IEEE Computer Society Press, Los Alamitos CA (2014) 369-376.

[24] Cox, M. G.: Algorithms for spline curves and surfaces. In: *Fundamental Developments of Computer-Aided Geometric Design*. Piegl, L. (Ed.) Academic Press, London, San Diego (1993) 51-76

[25] Crampin, M., Guifo, R., Read, G.A.: Linear approximation of curves with bounded curvature and a data reduction algorithm. *Computer Aided Design* **17**(6) (1985) 257-261.

[26] de Boor, C.A.: *Practical Guide to Splines*. Springer-Verlag (2001).

[27] de Boor, C.A., Rice, J.R.: *Least squares cubic spline approximation. I: fixed knots*. CSD TR 20, Purdue University, Lafayette, IN (1968); *ibid*: *Least squares cubic spline approximation. II: variable knots*. CSD TR 21, Purdue University, Lafayette, IN (1968)

[28] Denison, D.G.T., Mallick, B.K., Smith, A.F.M.: Automatic Bayesian curve fitting. *Journal of the Royal Statistical Society, Series B*, **60**(2), (1998) 330-350.

[29] Dierckx, P.: *Curve and Surface Fitting with Splines*. Oxford University Press, Oxford (1993).

[30] Draper, N. R., Smith, H.: *Applied Regression Analysis, 3rd ed*. Wiley-Interscience (1998).

[31] DiMatteo, I., Genovese, C.R., Kass, R. E.: Bayesian curve fitting with free-knot splines. *Biometrika*, **88** (2001) 1055-1071.

[32] Eberhart, R.C., Shi, Y.: Particle swarm optimization: developments, applications and resources. *Proceedings of the 2001 Congress on Evolutionary Computation* (2001) 81-86

[33] Echevarría, G., Iglesias, A., Gálvez, A.: Extending neural networks for B-spline surface reconstruction. *Lectures Notes in Computer Science*, **2330** (2002) 305-314.

[34] Eck, M., Hadenfeld, J.: Local energy fairing of B-spline curves. In: Faring, G., Hagen, H., Noltemeier, H. (eds): *Computing* 10. Springer-Verlag (1995)

[35] Eck, M., Hoppe, H.: Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, *Proc. SIGGRAPH'96* (1996) 325-334

[36] Engelbretch, A.P.: *Fundamentals of Computational Swarm Intelligence.* John Wiley and Sons, Chichester, England (2005)

[37] Fang, L., Gossard, D.C.: Multidimensional curve fitting to unorganized data points by nonlinear minimization. *Computer-Aided Design*, **27**(1) (1995) 48-58.

[38] Farin, G.: *Curves and surfaces for CAGD (5th ed.).* Morgan Kaufmann, San Francisco (2002).

[39] Fister I., Fister Jr., I., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, (in press) DOI: *http://dx.doi.org/10.1016/j.swevo.2013.06.001.*

[40] Fister I., Yang, X.S., Brest, J., Fister Jr., I.: Memetic self-adaptive firefly algorithm. In: Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M. (Eds.): *Swarm Intelligence and Bio-Inspired Computation. Theory and Applications*, Elsevier (2013) 73-102.

[41] Fister I., Yang, X.S., Brest, J., Fister Jr., I.: Modified firefly algorithm using quaternion representation. *Expert Systems with Applications*,**40** (2013) 7220-7230.

[42] Fister Jr., I., Yang, X.S., Fister I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik (English edition)*, **80**(3) (2013) 1-7.

[43] Fister Jr., I., Yang, X.-S., Fister, I., Brest, J.: Memetic firefly algorithm for combinatorial optimization. In: Filipic, B., Silc, J. (Eds.): *Proc. of 5th. Int. Conf. on Bioinspired Optimization Methods and their Applications (BIOMA 2012)*, Jozef Stefan Institute, Ljubljana, Slovenia (2012).

[44] Fister Jr., I.: *A comprehensive review of bat algorithms and their hybridization.* M.Sc. thesis, University of Maribor, Slovenia (2013).

[45] Fister Jr., I., Perc, M., Ljubic, K., Kamal, S.M., Iglesias, A., Fister, I.: Particle swarm optimization for automatic creation of complex graphic characters. *Chaos, Solitons and Fractals*, **73** (2015) 29-35.

[46] Floater, M.S.: Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, **14** (1997) 231-250

[47] Foley, T.A.: Interpolation to scattered data on a spherical domain. In: Mason, J.C., Cox, M.G. (Eds.) *Algorithms for Approximation II*, Chapman and Hall, London, New York (1990) 303-310

[48] Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. *Proceedings of IEEE Symposium on Research in Security and Privacy*. Los Alamitos, CA. (1994) 202-212.

[49] Forsey, D.R., Bartels, R.H.: Surface fitting with hierarchical splines. *ACM Trans. Graph.* **14**, (1995) 134-161

[50] Franke, R.H., Schumaker, L.L.: A bibliography of multivariate approximation. In: *Topics in Multivariate Approximation*. Chui, C.K., Schumaker, L.L., Utreras, F.I. (Eds.) Academic Press, New York (1986).

[51] Fuchs, H., Kedem, Z.M., Uselton, S.P.: Optimal surface reconstruction from planar contours. *Communications of the ACM*, **20**(10) (1977) 693-702.

[52] Gálvez, A., Iglesias, A., Cobo, A., Puig-Pey, J., Espinola, J.: Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation. *Lectures Notes in Computer Science*, **4706** (2007) 680-693.

[53] Gálvez, A. Cobo, A., Puig-Pey, J. Iglesias, A.: Particle swarm optimization for Bézier surface reconstruction. *Lectures Notes in Computer Science*, **5102** (2008) 116-125.

[54] Gálvez, A., Iglesias A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. *Computer-Aided Design*, **43**(12) (2011) 1683-1692.

[55] Gálvez, A., Iglesias A.: Particle swarm optimization for computer graphics and geometric modeling: recent trends. In: *Particle Swarm Optimization: Theory, Techniques and Applications*, Nova Science Publishers, New York, USA (2011) 169-192.

[56] Gálvez, A., Iglesias A., Puig-Pey J.: Iterative two-step genetic-algorithm method for efficient polynomial B-spline surface reconstruction. *Information Sciences*, **182**(1) (2012) 56-76.

[57] Gálvez A., Iglesias A.: Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. *Information Sciences*, **192**(1) (2012) 174-192.

[58] Gálvez A., Iglesias A.: A new iterative mutually-coupled hybrid GA-PSO approach for curve fitting in manufacturing. *Applied Soft Computing*, **13**(3) (2013) 1491-1504.

[59] Gálvez A., Iglesias A.: Firefly algorithm for polynomial Bzier surface parameterization. *Journal of Applied Mathematics*, (2013) Article ID 237984, 9 pages.

[60] A. Gálvez, A. Iglesias, A. Avila, Discrete Bézier curve fitting with artificial immune systems. *Studies in Computational Intelligence*, **441** (2013) 59-75.

[61] Gálvez A., Iglesias A.: Firefly algorithm for explicit B-Spline curve fitting to data points. *Mathematical Problems in Engineering*, (2013) Article ID 528215, 12 pages.

[62] Gálvez A., Iglesias A.: From nonlinear optimization to convex optimization through firefly algorithm and indirect approach with applications to CAD/CAM. *The Scientific World Journal*, (2013) Article ID 283919, 10 pages.

[63] Gálvez A., Iglesias A.: An electromagnetism-based global optimization approach for polynomial Bézier curve parameterization of noisy data points. *Proceedings of Cyberworlds 2013*. IEEE Computer Society Press, (2013) 259-266.

[64] Gálvez A., Iglesias A.: Firefly algorithm for Bézier curve approximation. *Proceedings of Int. Conf. on Computational Science and Its Applications - ICCSA'2013*. IEEE Computer Society Press, (2013) 81-88.

[65] Gálvez A., Iglesias A., Cabellos, L.: Discrete rule-based local search metaheuristic for Bézier curve parameterization. *Advances in Science and Technology Letters*, **22** (2013) 13-18.

[66] Gálvez A., Iglesias A., Cabellos, L.: Tabu search-based method for Bézier curve parameterization. *Int. Journal of Software Engineering and Applications*, **7**(5) (2013) 283-296.

[67] Gálvez A., Iglesias A.: New memetic self-adaptive firefly algorithm for continuous optimization. *International Journal of Bio-Inspired Computation, (in press).*

[68] Gálvez A., Iglesias A.: Cuckoo search with Lévy flights for weighted Bayesian energy functional optimization in global-support curve data fitting. *The Scientific World Journal*, (2014) Article ID 138760, 11 pages.

[69] Gálvez A., Iglesias A., Avila, A., Otero, C., Arias, R., Manchado, C.: Elitist clonal selection algorithm for optimal choice of free knots in B-spline data fitting. *Applied Soft Computing*, **26** (2015) 90-106.

[70] Gálvez, A., Iglesias, A., Avila, A.: Applying clonal selection theory to data fitting with rational Bézier curves. *Proc. of Cyberworlds 2014*, IEEE Computer Society Press, Los Alamitos CA (2014) 221-228.

[71] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley (1989).

[72] Goldberg, D.E.: *Optimal Initial Population Size for Binary-Coded Genetic Algorithms*, TCGA Report No.85001. University of Alabama (1985)

[73] Goldenthal, R., Bercovier, M.: Spline curve approximation and design by optimal control over the knots. *Computing* **72** (2004) 53-64.

[74] Goinski, A.: Evolutionary surface reconstruction. In: *Proc. IEEE Conference on Human System Interactions*, Krakow, Poland (2008) 464-469.

[75] Gopalakrishnan, G.: On the convergence of the LJ search method. *Journal of Optimization Theory and Applications*, **28**(3) (1979) 429-434.

[76] Gordon, W. J.: Spline-blended surface interpolation through curve networks. *J. Math. Mech.*, **18**(10) (1969) 931-952

[77] Greensmith, J., Aickelin, U.: Artificial dendritic cells: multi-faceted perspectives. *Proceedings of Human-Centric Information Processing Through Granular Modelling* (2009) 375-395.

[78] Greiner, G.: Variational design and fairing of spline surfaces. *Computer Graphics Forum*, **13**(3) (1994) 143-154

[79] Gu, P., Yan, X.: Neural network approach to the reconstruction of free-form surfaces for reverse engineering. *Computer-Aided Design* **27**(1) (1995) 59-64.

[80] Guo, B.: Surface Reconstruction from Points to Splines, *Computer-Aided Design* **29**(4) (1997) 269-277

[81] Guo, Q., Zhang, M.: A novel approach for multi-agent-based intelligent manufacturing system. *Information Sciences*, **179**(18) (2009) 3079-3090.

[82] Hagen, H., Santarelli, P.: Variational design of smooth B-spline surfaces. In: Hagen, H. (ed.): *Topics in Surface Modeling*. SIAM 85-94 (1992)

[83] Hertz, J., Krogh, A., Palmer, R.G.: *Introduction to the Theory of Neural Computation*. Addison Wesley, Reading, MA (1991).

[84] Hoffmann, M., Varady, L.: Free-form surfaces for scattered data by neural networks. *Journal for Geometry and Graphics*, **2** (1998) 1-6

[85] Hoffmann M.: Modified Kohonen neural network for surface reconstruction. *Publicationes Mathematicae*, **54** (1999) 857-864.

[86] Hoffmann M.: Numerical control of Kohonen neural network for scattered data approximation. *Numerical Algorithms*, **39**, (2005) 175-186.

[87] Holland, J.H.: *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press (1975)

[88] Hölzle, G.E.: Knot placement for piecewise polynomial approximation of curves. *Computer Aided Design*, **15**(5) (1993) 295-296.

[89] Hoos, H.H., Stutzle, T.: *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Series in Artificial Intelligence (2005).

[90] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. *Proc. of SIGGRAPH'92, Computer Graphics*, **26**(2) (1992) 71-78

[91] Hoppe, H.: *Surface Reconstruction from Unorganized Points*. Ph. D. Thesis, Department of Computer Science and Engineering, University of Washington (1994)

[92] Hoschek, J.: Smoothing of curves and surfaces. *Computer Aided Design*, **2** (1985) 97-105.

[93] Hoschek, J., Lasser, D.: *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, Wellesley, MA (1993).

[94] Iglesias, A., Echevarría, G., Gálvez, A.: Functional networks for B-spline surface reconstruction. *Future Generation Computer Systems*, **20**(8) (2004) 1337-1353.

[95] Iglesias, A., Gálvez, A.: A new artificial intelligence paradigm for computer aided geometric design. *Lectures Notes in Artificial Intelligence*, **1930** (2001) 200-213.

[96] A. Iglesias, A. Gálvez, Applying functional networks to fit data points from B-spline surfaces. In: *Proceedings of the Computer Graphics International, CGI'2001*, Hong-Kong (China). IEEE Computer Society Press, Los Alamitos, California (2001) 329-332.

[97] Iglesias, A., Echevarría, G., Gálvez, A.: Functional Networks for B-spline Surface Reconstruction. *Future Generation Computer Systems*, **20**(8) (2004) 1337-1353.

[98] A. Iglesias, A. Gálvez, Curve fitting with RBS functional networks. In: *Proc. of International Conference on Convergence Information Technology-ICCIT'2008*, Busan (Korea). IEEE Computer Society Press, Los Alamitos, California (2008) 299-306.

[99] Iglesias, A., Gálvez, A.: Hybrid functional-neural approach for surface reconstruction. *Mathematical Problems in Engineering*, (2014) Article ID 351648, 13 pages.

[100] Iglesias, A., Gálvez, A.: Memetic firefly algorithm for data fitting with rational curves. In: *Congress Evolutionary Computation-CEC'2015*, Sendai (Japan). IEEE CS Press, CA (2015) 507-514.

[101] Iglesias, A., Gálvez, A., Collantes, M.: Bat Algorithm for Curve Parameterization in Data Fitting with Polynomial Bézier Curves. *Proc. of Cyberworlds 2015*, Visby (Sweden). IEEE Computer Society Press, Los Alamitos CA (2015) 107-114.

[102] Iglesias, A., Gálvez, A., Collantes, M.: Global-Support Rational Curve Method for Data Approximation with Bat Algorithm. *Proc. of Int. Conference Artificial Intelligence and Applications, AIAI'2015*, Bayonne (France). IFIP Advances in Information and Communication Technology, **458** (2015) 191-205.

[103] Iglesias, A., Gálvez, A., Collantes, M.: A Bat Algorithm for Polynomial Bézier Surface Parameterization from Clouds of Irregularly Sampled Data Points. *Proc. of Int. Conference Natural Computation 2015, ICNC'2015*, Zhangjiajie (China). IEEE Computer Society Press, Los Alamitos CA (2015) 1034-1039.

[104] Iglesias, A., Gálvez, A., Collantes, M.: Iterative Sequential Bat Algorithm for Free-Form Rational Polynomial Bézier Surface Reconstruction. *Int. Journal of Parallel Programming*, (*submitted*).

[105] Iglesias, A., Gálvez, A., Collantes, M.: Four Adaptive Memetic Bat Algorithm Schemes for Bézier Curve Parameterization. *Transactions on Computational Science*, (*submitted*).

[106] Isheila, A. Gonneta, J.P., Joannica, D., Fontaine, J.F.: Systematic error correction of a 3D laser scanning measurement device. *Optics and Lasers in Engineering*, **49**(1) (2011) 16-24.

[107] Jerne, N. K.: Toward a network theory of the immune system. *Ann. Immunol.*, 125 (1974) 373-389.

[108] Jing, L., Sun, L.: Fitting B-spline curves by least squares support vector machines. In: *Proc. of the 2nd. Int. Conf. on Neural Networks & Brain*. Beijing (China). IEEE Press (2005) 905-909.

[109] Jones, M., Chen, M.: A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, **13**(3) (1994) 75-84

[110] Jupp, D.L.B.: Approximation to data by splines with free knots. *SIAM Journal of Numerical Analysis*, **15** (1978) 328-343.

[111] Kaufman, E., Klass, R.: Smoothing surfaces using reflection lines for families of splines. *Computer Aided Design*, **20** (1988) 312-316

[112] Keller, R.E., Banshaf, W., Mehnen, J., Weinert, K: CAD surface reconstruction from digitized 3D point data with a genetic programming/evolution strategy hybrid. In: *Advances in Genetic Programming 3*, MIT Press, Cambridge, MA, USA (1999) 41-65

[113] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. *IEEE International Conference on Neural Networks*, Perth, Australia (1995) 1942-1948

[114] Kennedy, J., Eberhart, R.C., Shi, Y.: *Swarm Intelligence*, San Francisco: Morgan Kaufmann Publishers (2001)

[115] Kharal, A., Saleem, A.: Neural networks based airfoil generation for a given C_p using Bézier-PARSEC parameterization. *Aerospace Science and Technology*, **1** (2012) 330-344.

[116] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science*, **220**(4598) (1983) 671-680.

[117] Knopf, G.K., Kofman, J.: Adaptive reconstruction of free-form surfaces using Bernstein basis function networks. *Engineering Applications of Artificial Intelligence*, **14**(5) (2001) 577-588.

[118] Kodama, T., Li, X., Nakahira, K., Ito, D.: Evolutionary computation applied to the reconstruction of 3-D surface topography in the SEM. *Journal of Electron Microscopy*, 54(5) (2005) 429-435

[119] Kumar, S.G., Kalra, P.K., Dhande, S.G.: Parameter optimization for B-spline curve fitting using genetic algorithms. Proc. of *Congress on Evolutionary Computation*, **3** (2003) 1871-1878

[120] Kumar, S.G., Kalra, P.K., Dhande, S.G.: Curve and surface reconstruction from points: an approach based on self-organizing maps. *Applied Soft Computing*, **5**(5) (2004) 55-66

[121] Laarhoven, P.J. M., Aarts, E.H.L.: *Simulated Annealing: Theory and Applications*. Springer, Netherlands (1987).

[122] Lee, T.C.M.: On algorithms for ordinary least squares regression spline fitting: a comparative study. *J. Statist. Comput. Simul.* **72**(8) (2002) 647-663.

[123] Leu, M.C., Peng, X., Zhang, W.: Surface Reconstruction for Interactive Modeling of Freeform Solids by Virtual Sculpting. *CIRP Annals - Manufacturing Technology*, **54**(1) (2005) 131-134

[124] Levoy, M. Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The Digital Michelangelo Project: 3D scanning of large statues. In *SIGGRAPH 2000*, New Orleans, (2000) 131-144

[125] Li, W., Xu, S., Zhao, G., Goh, L.P.: Adaptive knot placement in B-spline curve approximation. *Computer Aided Design* **37** (2005) 791-797.

[126] Li, M., Yan Y.: Bayesian adaptive penalized splines. *Journal of Academy of Business and Economics*, **2** (2006) 129-141.

[127] Li, M., Wang, Z.: A hybrid coevolutionary algorithm for designing fuzzy classifiers. *Information Sciences*, **179**(12) (2009) 1970-1983.

[128] Lim, C., Turkiyyah, G., Ganter, M., Storti, D.: Implicit reconstruction of solids from cloud point sets. *Proc. of 1995 ACM Symposium on Solid Modeling*, Salt Lake City, Utah (1995) 393-402

[129] Lindstrom, M.J.: Penalized estimation of free-knot splines. *Journal of Computational and Graphical Statistics*, **8**(2) (1999) 333-352

[130] Locatelli, M.: Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, **104**(1) (2000) 121-133.

[131] Loucera, C., Gálvez, A., Iglesias, A.: Simulated annealing algorithm for Bezier curve approximation. *Proc. of Cyberworlds 2014*, IEEE Computer Society Press, Los Alamitos CA (2014) 182-189.

[132] Luus, R., Jaakola, T.H.I.: Optimization by direct search and systematic reduction of the size of search region. *American Institute of Chemical Engineers Journal (AIChE)*, **19**(4) (1973) 760-766.

[133] Lyche T, Morken K. A data-reduction strategy for splines with applications to the approximation of functions and data. *IMA Journal of Numerical Analysis*, **8** (1988) 185-208.

[134] Ma, W.Y., Kruth, J.P.: Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. *Computer-Aided Design*, **27**(9) (1995) 663-675.

[135] Maekawa, I., Ko, K.: Surface construction by fitting unorganized curves. *Graphical Models* **64**, (2002) 316-332.

[136] Malek, M., Guruswamy, M., Pandya, M., Owens, H.: Serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. *Annals of Operations Research*, **21**(1) (1981) 59-84.

[137] Marsaglia, G.: Choosing a point from the surface of a sphere. *Annals of Mathematical Statistics*, **43** (1972) 645-646.

[138] Martinsson, H., Gaspard, F., Bartoli, A., Lavest, J.M.: Energy-based reconstruction of 3D curves for quality control. *Lecture Notes in Computer Science*, **4679** (2007) 414-428.

[139] Mendel, E., Krohling, R.A., Campos, M.: Swarm algorithms with chaotic jumps applied to noisy optimization problems. *Information Sciences* (2010), doi:10.1016/j.ins.2010.06.007

[140] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A. H., Teller, E.: Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**(6) (1953) 1087.

[141] Meyers, D., Skinnwer, S., Sloan, K.: Surfaces from contours. *ACM Transactions on Graphics,* **11**(3) (1992) 228-258

[142] Mirjalili, S., Mirjalili, S.M.,Yang, X.S.: Binary bat algorithm. Neural Computing and Applications, 25, 663–681 (2014)

[143] Mitchell, M.: *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. MIT Press (1998).

[144] Molinari, N., Durand, J.F., Sabatier, R.: Bounded optimal knots for regression splines. *Computational Statistics & Data Analysis*, **45**(2) (2004) 159-178.

[145] Moreton, H.P., Sequin, C.H.: Functional optimisation for fair surface design. *Computer Graphics*, **26**(2) (1992) 167-176.

[146] Mukai, T.: Latent nonuniform splines for animation approximation. *ACM Siggraph Asia 2012 Technical Briefs*, (2012) Article 3.

[147] Muller, M. E.: A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, **2**(4) (1959) 19-20.

[148] Oblonsek, C., Guid, N.: A fast surface-based procedure for object reconstruction from 3D scattered points. *Computer Vision and Image Understanding*, 69(2) (1998) 185-195

[149] Park, H., Kim, K.: Smooth surface approximation to serial cross-sections. *Computer Aided Design*, **28**(12) (1997) 995-1005

[150] Park, H.: An error-bounded approximate method for representing planar curves in B-splines. *Computer Aided Geometric Design* **21** (2004) 479-497.

[151] Park, H., Lee, J.H.: B-spline curve fitting based on adaptive curve refinement using dominant points. *Computer-Aided Design* **39** (2007) 439-451.

[152] Patrikalakis, N.M., Maekawa, T.: *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer Verlag, Heidelberg (2002).

[153] Picard, D., Revel, A., Cord, M.: An application of swarm intelligence to distributed image retrieval. *Information Sciences* (2010), doi:10.1016/j.ins.2010.03.003

[154] Piegl, L., Tiller, W.: *The NURBS Book*, Springer Verlag, Berlin Heidelberg (1997).

[155] Piegl, L., Tiller, W.: Least-square B-spline curve approximation with arbitrary end derivatives. *Eng Comput.* **16** (2000) 109-116.

[156] Pottmann, H., Leopoldseder, S. Hofer, M., Steiner, T., Wang, W.: Industrial geometry: recent advances and applications in CAD. *Computer-Aided Design*, **37** (2005) 751-766.

[157] Powell, M.J.D.: Curve fitting by splines in one variable. In: Hayes, J.G. (editor): *Numerical approximation to functions and data*. Athlone Press, London (1970).

[158] Prasad, M. , Zisserman, A., Fitzgibbon, A. W.: Single View Reconstruction of Curved Surfaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, *IEEE Computer Society Press*, Los Alamitos, CA (2006)

[159] Pratt, V.: Direct least-squares fitting of algebraic surfaces. *Proc. of SIGGRAPH'87, Computer Graphics*, **21**(4) (1987) 145-152.

[160] Puig-Pey, J., Gálvez, A., Iglesias, A., Rodríguez, J., Corcuera, P., Gutiérrez, F.: Polar isodistance curves on parametric surfaces. *Lectures Notes in Computer Science*, **2330** (2002) 161-170.

[161] Puig-Pey, J., Gálvez, A., Iglesias, A.: Helical curves on surfaces for computer-aided geometric design and manufacturing. *Lectures Notes in Computer Science*, **3044** (2004) 771-778.

[162] Puig-Pey, J., Gálvez, A., Iglesias, A.: Some applications of scalar and vector fields to geometric processing of surfaces. *Computers & Graphics*, **29**(5) (2005) 723-729.

[163] Puig-Pey, J., Gálvez, A., Iglesias, A., Corcuera, P., Rodríguez, J.: Some problems in geometric processing of surfaces. In: *Advances in Mathematical and Statistical Modeling* (Series: Statistics for Industry and Technology, SIT) Birkhauser, Boston (2008) 293-304.

[164] Rastrigin, L.A.: The convergence of the random search method in the extremal control of a many parameter system. *Automation and Remote Control*, **24**(10), (1963) 1337-1342.

[165] Razdan, A.: *Knot Placement for B-spline curve approximation.* Tempe, AZ: Arizona State University (1999).

[166] Reeves, C.R.: *Modern Heuristic Techniques for Combinatorial Problems.* Orient Longman (1993).

[167] Rekaby, A.: Directed artificial bat algorithm (DABA): a new bio-inspired algorithm. *Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Mysore (India) (2013) 1241-1246.

[168] Rice, J.R.: *The Approximation of Functions. Vol. 2.* Addison-Wesley, Reading, MA (1969).

[169] Rice, J.R.: *Numerical Methods, Software and Analysis. 2nd. Edition.* Academic Press, New York (1993).

[170] Saleem, M., Farooq, M., Di Caro, G.A.: Swarm intelligence based routing protocol for wireless sensor networks: survey and future directions. *Information Sciences* (2010), doi: 10.1016/j.ins.2010.07.005

[171] Sarfraz, M., Raza, S.A.: Capturing outline of fonts using genetic algorithms and splines. *Proc. of Fifth International Conference on Information Visualization IV'2001*, IEEE Computer Society Press (2001) 738-743.

[172] Sarioz, E.: An optimization approach for fairing of ship hull forms. *Ocean Engineering*, **33** (2006) 2105-2118.

[173] Sauter, J. A., Matthews, R., Parunak, H. V. D., Brueckner, S. A.: Evolving Adaptive Pheromone Path Planning Mechanisms. In: *Proceedings of First International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2002). Bologna, Italy (2002)

[174] Sauter, J. A., Matthews, R., Parunak, H. V. D., Brueckner, S. A.: Effectiveness of Digital Pheromones Controlling Swarming Vehicles in Military Scenarios. *Journal of Aerospace Computing, Information, and Communication*, **4**(5) (2007) 753-769

[175] Savchenko, V., Pasko, A., Okunev, O., Kunii, T.: Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, **14**(4) (1995) 181-188

[176] Schmitt, F., Barsky, B.A., Du, W.: An adaptive subdivision method for surface fitting from sampled data. *Proc. of SIGGRAPH'86, Computer Graphics*, **20**(4) (1986) 179-188.

[177] Schumer, M.A., Steiglitz, K.: Adaptive step size random search. *IEEE Transactions on Automatic Control*, **13**(3), (1968) 270-276.

[178] Sclaroff, S., Pentland, A.: Generalized implicit functions for computer graphics. *Proc. of SIGGRAPH'91, Computer Graphics*, **25**(4) (1991) 247-250

[179] Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multi-objective optimization. *Journal of the Operational Research Society*, **57**(10) (2005) 1143-1160.

[180] Sundar, S., Singh, A.: A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences*, **180**(11) (2010) 3182-3191.

[181] Tavares, R.S., Martins, T.C., Tsuzuki, M.S.G.: Simulated annealing with adaptive neighbourhood. A case study in off-line robot path planning. *Expert Systems with Applications*, **4** (2011) 2951–2965.

[182] Tilahun, S.L., Ong, H.C.: Modified firefly algorithm. *Journal of Applied Mathematics*, (2012) Article ID 467631.

[183] Ulker, E., Isler, V.: An artificial immune system approach for B-spline surface approximation problem. *Lecture Notes in Computer Science*, **4488** (2007) 49-56.

[184] Ulker, E., Arslan, A.: Automatic knot adjustment using an artificial immune system for B-spline curve approximation. *Information Sciences*, **179** (2009) 1483-1494.

[185] Unler, A., Murat, A., Chinnam, R.B.: mr^{2}PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. *Information Sciences* (2010), doi:10.1016/j.ins.2010.05.037

[186] Vanderbilt, D., Louie, S. G.: A Monte Carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, **56**(2) (1984) 259-271.

[187] Varady, T., Martin, R.: Reverse Engineering. In: Farin, G., Hoschek, J., Kim, M. (eds.): *Handbook of Computer Aided Geometric Design*. Elsevier Science (2002).

[188] Varady, T., Martin, R.R., Cox, J.: Reverse engineering of geometric models - an introduction. *Computer Aided Design*, **29**(4) (1997) 255-268.

[189] Vassilev, T.I.: Fair interpolation and approximation of B-splines by energy minimization and points insertion. *Computer-Aided Design*, **28**(9) (1996) 753-760.

[190] Veltkamp, R.C., Wesselink, W.: Modeling 3D curves of minimal energy. *Computer Graphics Forum*, **14**(3) (1995) 97-110.

[191] Wagner, T., Michelitsch, T., Sacharow, A.: On the design of optimizers for surface reconstruction. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference-GECCO2007* London, England (2007) 2195-2202

[192] Wang, W.P., Pottmann, H., Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics*, **25**(2) (2006) 214-238.

[193] Wang, Z., Chang, C., Li, M.: Optimizing least-significant-bit substitution using cat swarm optimization strategy. *Information Sciences* (2010), doi: 10.1016/j.ins.2010.07.011

[194] Weinert, K., Mehnen, J., Albersmann, F., Dreup, P.: New solutions for surface reconstruction from discrete point data by means of computational intelligence. *Proceedings of Intelligent Computation in Manufacturing Engineering-ICME'98*, Capri, Italy (1998) 431-438.

[195] Weinert, K., Surmann, T., Mehnen, J.: Evolutionary surface reconstruction using CSG-NURBS-hybrids. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference-GECCO2001*, San Francisco, USA (2001) 1456-1463.

[196] Weiss, V., Or, L., Renner, G., Varady, T.: Advanced surface fitting techniques. *Comput.-Aided Geomet. Design* **19** (2002) 19-42.

[197] Wen, A.S., : New solutions for surface reconstruction from discrete point data by means of computational intelligence. *Proceedings of Intelligent Computation in Manufacturing Engineering-ICME'98*, Sydney, Australia, IEEE CS Press (2006) 431-438.

[198] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**(1) (1997) 67-82.

[199] Yang, H.P., Wang, W.P., Sun, J.G.: Control point adjustment for B-spline curve approximation. *Computer-Aided Design* **36** (2004) 639-652.

[200] Yang, X.S.: Firefly algorithms for multimodal optimization. *Lectures Notes in Computer Science*, **5792** (2009) 169-178.

[201] Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: J. R. Gonzalez et al. (Eds.) Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). *Studies in Computational Intelligence*, **284** 65-74 (2010).

[202] Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. *Int. Journal of Bio-Inspired Computation*, **2**(2) (2010) 78-84.

[203] Yang, X.-S.: *Nature-Inspired Metaheuristic Algorithms (2nd. Edition).* Luniver Press, Frome, UK (2010).

[204] Yang, X.-S.: *Engineering Optimization: An Introduction with Metaheuristic Applications.* Wiley & Sons, New Jersey (2010).

[205] Yang, X. S..: Bat algorithm for multiobjective optimization. Int. J. Bio-Inspired Computation, 3(5), 267-274 (2011).

[206] Yang, X.S.: Bat algorithm: literature review and applications. *International Journal of Bio-Inspired Computation*, **5**(3), 141–149 (2013)

[207] Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: *Proc. World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE (2009) 210-214.

[208] Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. *Int. J. Mathematical Modelling and Numerical Optimization*, **1**(4) (2010) 330-343.

[209] Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, **29**(5), 464–483 (2012).

[210] Yoshimoto F., Moriyama, M., Harada T.: Automatic knot adjustment by a genetic algorithm for data fitting with a spline. *Proc. of Shape Modeling International'99*, IEEE Computer Society Press (1999) 162-169.

[211] Yoshimoto F., Harada T., Yoshimoto Y.: Data fitting with a spline using a real-coded algorithm. *Computer-Aided Design*, **35** (2003) 751-760.

[212] Yu, Y.: Surface reconstruction from unorganized points using self-organizing neural networks. In: *Proceedings of IEEE Visualization 99* (1999) 61-64.

[213] Zhang, C., Zhang, P., Cheng, F.: Fairing spline curves and surfaces by minimising energy. *Computer-Aided Design*, **33**(13) (2001) 913-923.

[214] Zhao, X., Zhang, C., Yang, B., Li, P.: Adaptive knot adjustment using a GMM-based continuous optimization algorithm in B-spline curve approximation. *Computer-Aided Design*, **43** (2011) 598-604.

[215] Zhao, L., Jiang, J., Song, C., Bao, L., Gao, J.: Parameter optimization for Bézier curve fitting based on genetic algorithm. *Advances in Swarm Intelligence* (2013) 451-458.