mathematics

MDPI

Article

Parameter Identification of Bivariate Fractal Interpolation Surfaces by Using Convex Hulls

Vasileios Drakopoulos [1,*], Dimitrios Matthes [1], Dimitrios Sgourdos [2] and Nallapu Vijender [3]

[1] Department of Computer Science and Biomedical Informatics, University of Thessaly, 35131 Lamia, Greece; dmatthes@uth.gr
[2] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, 15784 Athens, Greece; sgdimitris@gmail.com
[3] Department of Mathematics, Visvesvaraya National Institute of Technology Nagpur, Nagpur 440006, India; vijendernallapu@mth.vnit.ac.in
* Correspondence: vdrakop@uth.gr

Abstract: The scope of this article is to identify the parameters of bivariate fractal interpolation surfaces by using convex hulls as bounding volumes of appropriately chosen data points so that the resulting fractal (graph of) function provides a closer fit, with respect to some metric, to the original data points. In this way, when the parameters are appropriately chosen, one can approximate the shape of every rough surface. To achieve this, we first find the convex hull of each subset of data points in every subdomain of the original lattice, calculate the volume of each convex polyhedron and find the pairwise intersections between two convex polyhedra, i.e., the convex hull of the subdomain and the transformed one within this subdomain. Then, based on the proposed methodology for parameter identification, we minimise the symmetric difference between bounding volumes of an appropriately selected set of points. A methodology for constructing continuous fractal interpolation surfaces by using iterated function systems is also presented.

Keywords: convex hull; volume of a convex polyhedron; intersection of two convex polyhedra; fractal interpolation; iterated function system

MSC: 28A80; 41A30; 65D05

Citation: Drakopoulos, V.; Matthes, D.; Sgourdos, D.; Vijender, N. Parameter Identification of Bivariate Fractal Interpolation Surfaces by Using Convex Hulls. *Mathematics* **2023**, *11*, 2850. https://doi.org/10.3390/math11132850

Academic Editor: Jay Jahangiri

Received: 19 March 2023
Revised: 4 June 2023
Accepted: 22 June 2023
Published: 25 June 2023

1. Introduction

Interpolation focuses in general on constructing a continuous function which passes over a set of points that are considered as samples of an unknown function. Since most of the traditional interpolants are defined or constructed using the infinitely differentiable functions such as polynomials, exponential functions, trigonometric functions, and rational functions, generally traditional interpolants are "smooth" in nature.

On the other hand, many real-world and experimental signals are intricate and rarely show a sensation of smoothness in their traces. To address this issue, the interpolation by fractal (graph of) functions is introduced by M. F. Barnsley in Refs. [1,2], which is based on the theory of *iterated function system*. *Fractal interpolation* based on iterated function systems provides a convenient way for constructing continuous functions that intervene sets with irregular shape, such as seismic, medical, geographic data, etc. Fractal interpolation does not use an analytical formula or algorithm for computing the graph for given coordinates. Instead, the whole function is constructed through algorithms as regulated cardinality set of points.

A *fractal interpolation function* is a continuous function whose graph has non-integer *Hausdorff–Besicovitch* or simply *fractal dimension*. If its fractal dimension lies between 1 and 2, then it is called *fractal interpolation curved line* or *fractal interpolation curve*. Similarly, if its fractal dimension lies between 2 and 3, then it is called *fractal interpolation surface*.

The construction of fractal interpolation surfaces by using an appropriately chosen iterated function system was initially proposed by Peter R. Massopust [3], who examined triangular passages with coplanar boundary data. Jeffrey S. Geronimo and Douglas Hardin in [4] studied polygonal regions with arbitrary interpolation points, but equal vertical scaling factors. Nailiang Zhao in [5] faced the factors as a continuous 'contraction function', while Peter R. Massopust [6], proposed the construction of FIS as a tensor product of two continuous univariate functions. Bivariable fractal interpolation functions as non-tensor products on some rectangular passages were constructed by Xiao-yuan Qian [7]. Hong-Yong Wang [8] used a wide class of three-dimensional IFS and proved that their attractors are a class of fractal interpolation surfaces. A method based on bivariable functions on rectangular grids for generating fractal interpolation surfaces is presented by V. Drakopoulos and P. Manousopoulos in [9]. Can every attractor of an iterated function system be the graph of a continuous bivariable fractal function? We answer this question by providing a necessary condition. Furthermore, we address the advantages and disadvantages of few foremost constructions of bivariable fractal interpolation functions on rectangular grids.

By exploiting the concepts of geostatistics and trend surface analyses, the partition of the local field and the determination of a vertical scaling factor are studied in [10]. This is very useful for simulation of local stochastic irregular roughness on the fracture surface. By using the principles of the trend surface analyses, the deviations on the information points are used in [11] as a vertical scaling factor. Another methodology to approximate any natural surface by using recurrent bivariate fractal interpolation surfaces is outlined in [12]. The freedom in selecting the parameters that define fractal function plays a vital role in many applications [13]. At the same time the suitable choice of free parameters is a tedious inverse problem as the closeness if fit of fractal interpolation function is sensitive with respective to vertical scaling factors. However, it should be noted that the proper selection of the free parameters is a challenging "inverse problem", since the closeness of fit of a fractal interpolation function is mainly influenced by the determination of its vertical scaling factors, for which no straightforward method is available.

In this article, the parameter identification presented in [14] is extended to bivariate fractal interpolation surfaces by reducing the construction to a problem of computational geometry. Our goal is to find the optimum vertical scaling factors of fractal interpolation, such as to achieve minimising the symmetric difference between bounding volumes of appropriately selected points. For the implementation of the proposed methodology, we use the algorithms of finding the convex hull of a set of points, of computing the volume of a convex polyhedron as well as the pairwise intersections between two convex polyhedra. The use of optimum vertical scaling factors leads to the maximum overlap between corresponding bounding volumes, achieving better representation of the data points.

2. Finding the Convex Hull of a Set of Points

The *convex hull* of a set of points is the smallest convex set that contains these points. Specifically, for a set S of n points, the convex hull denoted by $CH(S)$ is the minimum convex polygon in $\mathbb{R}^2$ or convex polyhedron in $\mathbb{R}^3$ such that each one of the n points belongs to the boundary of $CH(S)$ or is interior of the $CH(S)$. A convex hull is represented with a set of facets and a set of adjacency lists giving the neighbours and vertices for each facet. The boundary elements of a facet are called *ridges*. Each ridge signifies the adjacency of two facets.

Various algorithms for finding convex hull have been proposed. Of particular interest are those which are variations of a randomised incremental algorithm that was proposed by Clarkson and Shor [15], as they have optimal expected performance. An incremental algorithm for the convex hull repeatedly adds a point to the convex hull of the previously processed points.

In this article, the Quickhull algorithm [16] is selected since it optimises the selection of a point in each step of the procedure, as it processes the furthest point of an outside set instead of an arbitrary point. It executes faster than the corresponding randomised

algorithms because it processes fewer interior points and it also reuses the memory occupied by old facets; see Listing 1. For a precision of the input points equal to $O(\log n)$, the worst-case complexity of Quickhull is $O(n \log u)$ for $d \leq 3$, and $O(nf_u/u)$ for $d \geq 4$, where n the number of input points in $\mathbb{R}^d$ and u the number of output vertices. For an efficient comparison of Quickhull with the randomised incremental algorithms, the effect of randomisation on time efficiency should be isolated.

Listing 1. Quickhull algorithm for the convex hull in $\mathbb{R}^d$.

1. Create a simplex of $d + 1$ points
2. For each facet F.
   - 2.1. For each unassigned point p
     - 2.1.1. If p is above F
       - i. Assign p to F's outside set
3. For each facet F with a non-empty outside set
   - 3.1. Select the furthest point p of F's outside set
   - 3.2. Initialize the visible set V to F
   - 3.3. For all unvisited neighbors N of facets in V
     - 3.3.1. If p is above N
       - i. Add N to V
   - 3.4. The boundary of V is the set of horizon ridges H
   - 3.5. For each ridge R in H
     - 3.5.1. Create a new facet from R and p
     - 3.5.2. Link the new facet to its neighbors
   - 3.6. For each new facet F'
     - 3.6.1. For each unassigned point q in an outside set of a facet in V
       - i. IF q is above F', assign q to F' outside set,
   - 3.7. Delete the facets in V

3. Calculating the Volume of a Convex Polyhedron

Volume is the quantity of three-dimensional space enclosed by some closed boundary, for example the space that a substance (solid, liquid, gas or plasma) or shape occupies or contains. The volume of objects with simple geometric shape like a cube, sphere, pyramid, cone, etc., can easily be calculated by using arithmetic formulas. However, for polyhedra with more complicated shape, the procedure becomes much more intricate. The basic idea of the proposed methodology is to simplify the problem by splitting the initial convex polyhedron in separate, non-overlapping convex tetrahedra, where it is possible to calculate their volumes directly. Then, the volume of the initial polyhedron is equal to the sum of volumes of all encountered tetrahedra.

For partitioning the initial convex polyhedron, one vertex must be selected, that will be the fixed point for the formation of the tetrahedrons, while it is prerequisite all the facets of the polyhedron to be triangulated. Each tetrahedron is formed as the compound of this vertex with one of the triangles of the triangulated polyhedron. Triangles, which are coplanar with the fixed vertex, are excluded from the procedure, as they would lead to tetrahedra with zero volume. The algorithm of the procedure is outlined in Listing 2.

Listing 2. Volume of a convex polyhedron.

1. Select a vertex A from the initial convex polyhedron P as a starting point for the formation of the tetrahedra that partition P.
2. Set the *Volume* V of P equal to 0.
3. For each triangle Tr_i of the triangulated polyhedron P
   - 3.1. If the vertex A isn't coplanar with the triangle Tr_i
     - 3.1.1. Form a tetrahedron T_i from the vertices of the triangle Tr_i and the vertex A
     - 3.1.2. Compute the *Volume* V_i of the tetrahedron T_i
     - 3.1.3. Add the V_i to the cumulative calculated *Volume* V of P
4. Return the *Volume* V of the convex polyhedron P

The complexity of the algorithm is proportionate to the number of tetrahedrons that the initial convex polyhedron has been partitioned into, and thus proportionate to the number of triangles that have emerged during the triangulation. If n is the number of triangles, then the complexity of the proposed methodology is $O(n)$. As the triangulated form of the initial polyhedron has resulted from its faces, the complexity of the algorithm can be expressed as a function of the *vertices* (v), the *edges* (e) or the *faces* (f) considering the relation $v + f - e = 2$.

4. Intersection between Two Convex Polyhedra

The intersection between two convex polyhedra in three-dimensional space corresponds to their overlapping area. It is equivalent to the finite shape of space, which is bounded by n polygonal planes, and simultaneously enclosed in both initial convex polyhedra. Provided that the initial polyhedra overlap, their intersection is also a convex polyhedron.

While the definition of intersection seems simplistic, finding an efficient algorithm to calculate it, constitutes a complicated problem in computational geometry. The difficulty lies in the complexity that the schema of the polyhedra may have and in the way that they overlap.

The backbone structure of the proposed methodology is the creation of a plane from each facet of the one polyhedron and the partition of the second polyhedron by each plane that has emerged. The part of the second polyhedron, which is located on the opposite half-space compared to the first polyhedron, is rejected, while the part located in the same half-space remains. At each iteration of the algorithm presented as Listing 3, the second polyhedron is replaced by the polyhedron constructed in the previous step. A prerequisite for the implementation of the algorithm is the second polyhedron to be triangulated. This condition has its origin in the way the second polyhedron is partitioned by each resulting plane.

Listing 3. Intersection of two convex polyhedra.

1. Take as input the two polyhedra A and B
2. Set the polyhedron B as the processing polyhedron P
3. For each facet of the polyhedron A
   - 3.1. Set the plane E that divides the $\mathbb{R}^3$ into two half-spaces
   - 3.2. For each triangle Tr_i of the triangulated polyhedron P
     - 3.2.1. If all the vertices of the Tr_i are located in the same half-space with the A
       - i. Add the triangle to the triangulation form of the polyhedron T
     - 3.2.2. Else, if all the vertices of the Tr_i are located in the opposite half-space from the A
       - i. Reject the triangle
     - 3.2.3. Else, if only one vertex of the Tr_i is located in the same half-space with the A
       - i. Add to the triangulation form of the polyhedron T, the triangle that is constructed from this vertex and the intersection points of the sides of the Tr_i with the plane E.
     - 3.2.4. Else (two vertices of the Tr_i are located in the same half-space with the A)
       - i. Triangulate the quadrilateral that is constructed from the two vertices of the Tr_i, that are located in the same half-space with the A, and the two intersection points of the sides of the Tr_i with the plane E
       - ii. Add to the triangulation form of the polyhedron T, the two triangles that are constructed from the step (i)
   - 3.3. Triangulate the n-gon that is constructed from the intersection points of the triangles of P with the plane E, and combine the resulting triangles to construct the polyhedron T
   - 3.4. Set as the processing polyhedron P the convex polyhedron T and initialise T
4. Return the polyhedron P

5. Parameter Identification of 2D Fractal Interpolation Functions

We apply the methods presented in the previous sections in the field of fractal interpolation for parameter identification in $\mathbb{R}^3$. They are constructing blocks for the implementation of the proposed algorithm for the parameter identification of 2D fractal interpolation functions by using bounding volumes. In what follows, we abbreviate by f^k the k-fold composition $f \circ f \circ \cdots \circ f$.

5.1. Iterated Function Systems

A (hyperbolic) *Iterated Function System*, or IFS for short, on the metric space $(\mathbb{R}^d, \|\cdot\|)$ is defined as a pair $\{\mathbb{R}^d; w_{1-N}\}$, where $w_n \colon \mathbb{R}^d \to \mathbb{R}^d$, $n = 1, 2, \ldots, N$, is a finite set of contractions with *contractivity factor* s_n, i.e.,

$$\|w_n(\mathbf{x}) - w_n(\mathbf{y})\| \leq s_n \|\mathbf{x} - \mathbf{y}\|$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and for some $s_n \in [0, 1)$.

The *attractor* of a (hyperbolic) IFS is the unique set

$$\mathcal{A}_\infty = \lim_{k \to \infty} W^k(E_0)$$

for every starting set E_0, where

$$W(E) = \bigcup_{n=1}^{N} w_n(E) \text{ for all } E \in \mathcal{H}(\mathbb{R}^d),$$

where $\mathcal{H}(\mathbb{R}^d)$ is the metric space of all nonempty, compact subsets of $\mathbb{R}^d$ with respect to some metric, e.g., the Hausdorff metric. The map W is called the *Hutchinson operator* or the *collage map* to alert us to the fact that $W(E)$ is formed as a union or 'collage' of sets.

Recall that a transformation w is *affine*, if it may be represented by a matrix A and translation $\mathbf{t}$ as $w(\mathbf{x}) = A\mathbf{x} + \mathbf{t}$, or (if $X = \mathbb{R}^3$)

$$w\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & g \\ h & k & s \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} l \\ m \\ r \end{pmatrix}.$$

The *code* of w is the 12-tuple $(a, b, c, d, e, g, h, k, s, l, m, r)$, and the *code of an IFS* is a table whose rows are the codes of $w_1, w_2, \ldots, w_N$.

5.2. Fractal Interpolation Surfaces

Let $\Delta_{x,1}$, $\Delta_{x,2}$ be two partitions of the real compact interval $I_x = [a, b]$, i.e., $\Delta_{x,1} = \{u_0, u_1, \ldots, u_{M'}\}$ satisfying $a = u_0 < u_1 < \cdots < u_{M'} = b$ and $\Delta_{x,2} = \{x_0, x_1, \ldots, x_M\}$ satisfying $a = x_0 < x_1 < \cdots < x_M = b$, such that $\Delta_{x,1}$ is a *refinement* of $\Delta_{x,2}$. Likewise, let $\Delta_{y,1}$, $\Delta_{y,2}$ be two partitions of the real compact interval $I_y = [c, d]$, i.e., $\Delta_{y,1} = \{v_0, v_1, \ldots, v_{N'}\}$ satisfying $c = v_0 < v_1 < \cdots < v_{N'} = d$ and $\Delta_{y,2} = \{y_0, y_1, \ldots, y_N\}$ satisfying $c = y_0 < y_1 < \cdots < y_N = d$, such that $\Delta_{y,1}$ is a *refinement* of $\Delta_{y,2}$.

Let $K = D \times \mathbb{R}$, such that $D = I_x \times I_y$, be a complete metric space. Let us represent as

$$P = \{(u_k, v_l, \hat{z}_{k,l} = \hat{z}(u_k, v_l)) \in K : k = 0, 1, \ldots, M';\ l = 0, 1, \ldots, N'\}$$

the given set of *data points* and as

$$Q = \{(x_i, y_j, z_{i,j} = z(x_i, y_j)) \in K : i = 0, 1, \ldots, M;\ j = 0, 1, \ldots, N\}$$

the *interpolation points* satisfying $Q \subset P$.

The $I_{x_m} = [x_{m-1}, x_m]$, $I_{y_n} = [y_{n-1}, y_n]$, $D_{m,n} = I_{x_m} \times I_{y_n}$, correspond to the defined interpolation intervals, while $P_{m,n} = \{(u, v, \hat{z}) \in P : (u, v) \in D_{m,n}\}$ are the data points within the $m \times n$ interpolation domain, for all $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$.

Let $\{K; w_{1-m,1-n}\}$ be an IFS with transformations

$$w_{m,n}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_m & 0 & 0 \\ c_n & 0 & 0 \\ e_{m,n} & f_{m,n} & s_{m,n} \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} b_m \\ d_n \\ g_{m,n}xy + k_{m,n} \end{pmatrix}$$

constrained to satisfy

$$w_{m,n}\begin{pmatrix} x_0 \\ y_0 \\ z_{0,0} \end{pmatrix} = \begin{pmatrix} x_{m-1} \\ y_{n-1} \\ z_{m-1,n-1} \end{pmatrix}, \quad w_{m,n}\begin{pmatrix} x_M \\ y_0 \\ z_{M,0} \end{pmatrix} = \begin{pmatrix} x_m \\ y_{n-1} \\ z_{m,n-1} \end{pmatrix},$$

$$w_{m,n}\begin{pmatrix} x_0 \\ y_N \\ z_{0,N} \end{pmatrix} = \begin{pmatrix} x_{m-1} \\ y_n \\ z_{m-1,n} \end{pmatrix}, \quad w_{m,n}\begin{pmatrix} x_M \\ y_N \\ z_{M,N} \end{pmatrix} = \begin{pmatrix} x_m \\ y_n \\ z_{m,n} \end{pmatrix}$$

for every $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. Solving the above equations results in

$$a_m = \frac{x_m - x_{m-1}}{\Delta_x}, \quad b_m = \frac{x_M x_{m-1} - x_0 x_m}{\Delta_x}, \quad c_n = \frac{y_n - y_{n-1}}{\Delta_y}, \quad d_n = \frac{y_N y_{n-1} - y_0 x_n}{\Delta_y},$$

$$g_{m,n} = \frac{z_{m,n} + z_{m-1,n-1} - z_{m-1,n} - z_{m,n-1} - s_{m,n}(z_{0,0} + z_{M,N} - z_{0,N} - z_{M,0})}{\Delta_x \Delta_y},$$

$$e_{m,n} = \frac{z_{m,n-1} - z_{m-1,n-1} + s_{m,n}(z_{0,0} - z_{M,0}) - g_{m,n}\Delta_x y_0}{\Delta_x},$$

$$f_{m,n} = \frac{z_{m-1,n} - z_{m-1,n-1} + s_{m,n}(z_{0,0} - z_{0,N}) - g_{m,n}\Delta_y x_0}{\Delta_y},$$

$$k_{m,n} = z_{m,n} - e_{m,n}x_M - f_{m,n}y_N - s_{m,n}z_{M,N} - g_{m,n}x_M y_N$$

i.e., the real parameters a_m, b_m, c_n, and d_n, are determined by the interpolation points while $e_{m,n}$, $f_{m,n}$, $g_{m,n}$, and $k_{m,n}$ are determined by the interpolation points as well as by the free parameters $s_{m,n}$. The transformations $w_{m,n}$ are bivariable transformations, where $s_{m,n}$ are their *vertical scaling factors*, where $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$; see Figure 1.

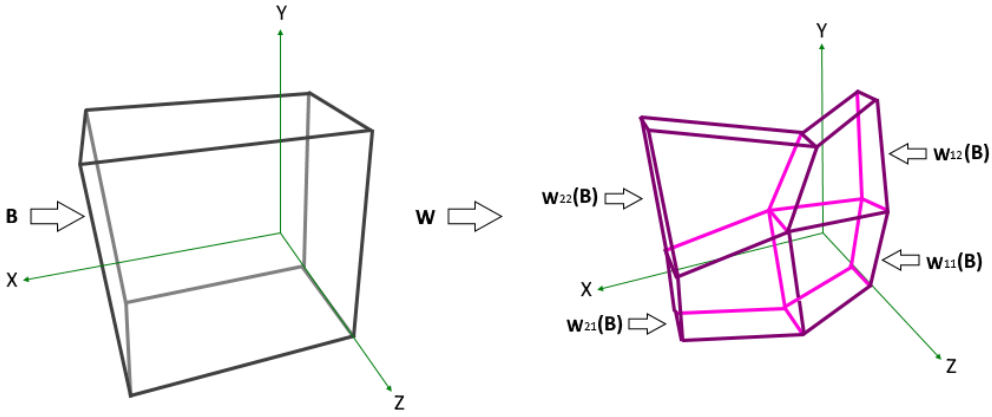


Figure 1. A collage map $W\colon \mathcal{H}(\mathbb{R}^3) \to \mathcal{H}(\mathbb{R}^3)$ for $M = N = 2$.

From [9], we have the following.

Theorem 1. *Let* $\{X; w_{1-N,1-M}\}$ *be a hyperbolic IFS with contractivity factor* $s = \max\{s_{m,n} : m = 1, 2, \ldots, M;\ n = 1, 2, \ldots, N\}$. *The transformation* $W\colon \mathcal{H}(X) \to \mathcal{H}(X)$, *where*

$$W(B) = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} w_{m,n}(B), \quad \text{for all } B \in \mathcal{H}(X),$$

is a contraction mapping of the complete metric space $(\mathcal{H}(X),\ h(\rho))$ *with contractivity factor s. The unique compact set* $\mathcal{A} \in \mathcal{H}(X)$ *satisfies*

$$\mathcal{A} = W(\mathcal{A}) = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} w_{m,n}(\mathcal{A}),$$

is called the attractor *of the hyperbolic IFS and it is equal to*

$$\mathcal{A} = \lim_{k \to \infty} W^k(B), \text{ for all } B \in \mathcal{H}(X).$$

5.3. Identifying the Vertical Scaling Factors

Let $B \in K_0^3$ be the convex bounding volume of P and $B_{m,n} \in K_0^3$ be the convex bounding volumes of $P_{m,n}$. The parameters $s_{m,n}$ must be computed in such a way as to derive maximum overlap between the corresponding bounding volumes. The objective is to minimise the volume of the non-overlapping parts of the convex hull of the set $P_{m,n}$ and the transformed by $w_{m,n}$ convex hull of the set P, for all $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. That corresponds to the minimisation of the symmetrical distance:

$$\begin{aligned}&\delta_s(\, CH(P_{m,n}),\, w_{m,n}(CH(P))\,)\\&= Volume\{CH(P_{m,n})\} \,+\, Volume\{w_{m,n}(CH(P))\} \,-\, 2\cdot Volume\{CH(P_{m,n}) \cap w_{m,n}(CH(P))\}\\&= Volume\{CH(P_{m,n})\} \,+\, Volume\{w_{m,n}(CH(P))\} \,-\, 2\cdot Volume\{CH(P_{m,n} \cap w_{m,n}(P))\}\end{aligned} \tag{1}$$

Since $w_{m,n}$ are affine, it implies that $w_{m,n}(CH(\cdot)) = CH(w_{m,n}(\cdot))$. As implied by Equation (1), the calculation of δ_S is algorithmic and involves the computation of convex hulls, intersections of polyhedra and volumes. Therefore, a method for two-dimensional minimisation without derivatives should be used, such as Brent's method [17], which is a bracketing method with parabolic interpolation. Initially, for the method to converge to optimum $\hat{s}^+_{m,n}$, we must initially bracket it, i.e., provide three constants $s^a_{m,n}, s^b_{m,n}, s^c_{m,n}$, such that

$$\begin{gathered}0 \le s^a_{m,n} < s^b_{m,n} < s^c_{m,n} \le 1,\\ Vol(s^b_{m,n}) < Vol(s^a_{m,n}),\, Vol(s^b_{m,n}) < Vol(s^c_{m,n}),\\ s^a_{m,n} < \hat{s}^+_{m,n} < s^c_{m,n}.\end{gathered}$$

A good approximation is to estimate $s^b_{m,n}$ such that

$$Vol\{w_{m,n}(CH(P))\} \approx Vol\{CH(P_{m,n})\}.$$

Next $s^a_{m,n}, s^c_{m,n}$ follow from $s^b_{m,n}$, respectively, by a factor, that is

$$\begin{aligned}s^b_{m,n} &= Vol\{CH(P_{m,n}\}/(a_{m,n}\cdot c_{m,n}\cdot Vol\{CH(P)\}),\\ s^a_{m,n} &= s^b_{m,n}/c,\\ s^c_{m,n} &= min\{cs^b_{m,n}, 1\},\end{aligned}$$

for some $c > 1$. The constant c can be determined a priori, e.g., $c = 2$, or more safely using an iterative procedure of checking successively larger values until a suitable bracketing triplet is found. In cases of negative factors, the optimum $\hat{s}^-_{m,n}$ can be similarly calculated by using the bracketing triplet

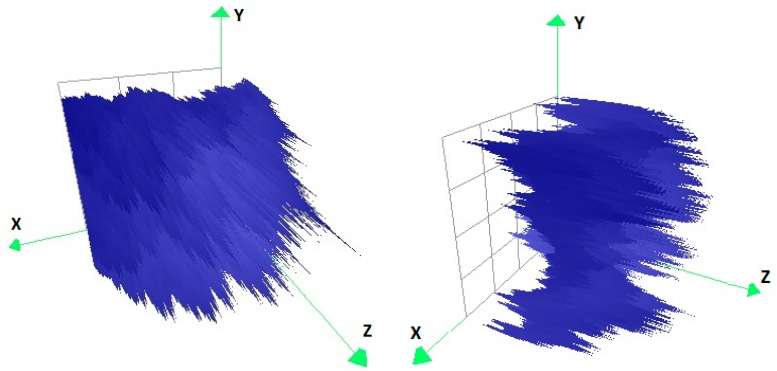
$$\begin{aligned}s^{b'}_{m,n} &= -s^b_{m,n},\\ s^{a'}_{m,n} &= max\{cs^{b'}_{m,n}, -1\},\\ s^{c'}_{m,n} &= s^{b'}_{m,n}/c.\end{aligned}$$

The algorithm for finding the optimum vertical scaling factors can be outlined in Listing 4.

Listing 4. Calculation of the optimum vertical scaling factors of FIS.

1. Compute the convex hull of the set of the data points P.
2. For each subdomain $D_{m,n}$, where $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$
   - 2.1. Compute the known parameters $a_{m,n}$, $b_{m,n}$, $c_{m,n}$, $d_{m,n}$, $e_{m,n}$, $f_{m,n}$, $g_{m,n}$, $k_{m,n}$ of the transformation $w_{m,n}$
   - 2.2. Compute the convex hull of the set $P_{m,n}$
   - 2.3. Find the optimum $\hat{s}^+_{m,n} \in [0, 1)$ that minimise the volume of non-overlapping parts of volumes $CH(P_{m,n})$ and $w_{m,n}(CH(P))$, using the bracketing triplet $(s^b_{m,n}/c,\ s^b_{m,n},\ \min\{cs^b_{m,n}, 1\})$,
     where $c > 1$ and $s^b_{m,n} = Volume\{CH(P_{m,n})\}\ /\ a_{m,n}\ \cdot\ c_{m,n}\ \cdot\ Volume\{CH(P)\}$
   - 2.4. Find the optimum $\hat{s}^-_{m,n} \in (-1, 0]$ that minimise the volume of non-overlapping parts of volumes $CH(P_{m,n})$ and $w_{m,n}(CH(P))$, using the bracketing triplet $(\max\{cs^{b'}_{m,n}, -1\},\ s^{b'}_{m,n},\ s^{b'}_{m,n}/c)$, where $c > 1$ and $s^{b'}_{m,n} = -s_b$
   - 2.5. If $Volume\{\hat{s}^+_{m,n}\} \le Volume\{\hat{s}^-_{m,n}\}$, then $\hat{s}_{m,n} = \hat{s}^+_{m,n}$, else $\hat{s}_{m,n} = \hat{s}^-_{m,n}$

The calculation of a vertical scaling factor $s_{m,n}$ depends on the number of data points within the respective interpolation interval surface, while in the worst case scenario it requires time $O(K \log(K) \ / \ (M \cdot N))$. Since there are $M \cdot N$ factors $s_{m,n}$, the worst-case computational complexity of the algorithm is proportionate to the number of the data points, i.e., $O(K \log(K))$. Moreover, convex hulls usually have few vertices, so calculations are quick. Figure 2 shows the results of three examples of fractal interpolation according to the proposed methodologies.



(**a**) (**b**)

Figure 2. Example of fractal interpolation for (**a**) 3×3, (**b**) 4×4, partitions of the XY plane.

6. Examples and Discussion

In this section, examples for finding the convex hull of a set of points, calculating the volume of a convex polyhedron and determining the intersection of two convex polyhedra, and finally identifying the parameters of two-dimensional fractal interpolation functions by using bounding volumes are presented and discussed. The implementation of the proposed algorithms and their integration into a software tool were written in the *C++* programming language by using $Microsoft^{®}$ $VisualStudio^{®}$ IDE. The library *CGAL* was extensively used together with the *DirectX* library for the three-dimensional representation of the results.

The most common algorithm to compute fractals derived by IFSs is called the *chaos game* or *random iteration algorithm*. It consists of picking a random point in the plane, then iteratively applying one of the functions chosen at random from the function system and drawing the point. An alternative algorithm, the *deterministic iteration algorithm*, or *DIA* for short, is to generate each possible sequence of functions up to a given maximum length and then to plot the results of applying each of these sequences of functions to an initial point or shape.

As an initial data set of points for 2×2 partitioning of the XY plane, the points in Table 1a are used. The interpolation points corresponding to this original data set are listed in Table 1b. Figure 3 corresponds to the original mesh of the interpolation points of the example, while Figure 4 corresponds to the convex hull of the original data set, calculated with the Quickhull algorithm.

To calculate the optimal factors $s_{m,n}$, with $m = 1.2$ and $n = 1.2$ of each subdomain of the example, according to the algorithm presented in Section 5.3, the factors that minimize the volume of the non-overlapping parts of the convex hull of the set $P_{m,n}$ and of the convex hull of the set P transformed by $w_{m,n}$ for each $m = 1, 2$ and $n = 1, 2$ must be chosen. The convex hulls of $P_{m,n}$, for each $m = 1, 2$ and $n = 1, 2$, of the subdomains of the example, are represented in Figures 5–8.

Table 1. (a) The initial data set for 2 × 2 partitioning of the XY plane, (b) the correspong interpolation points.

(a)			(b)			
x	**y**	**z**	**y**	**x** **0**	**0.5**	**1**
0.00	0.00	0.20	0	0.2	0.2	0.2
0.00	0.50	0.20	0.5	0.2	1.0	0.2
0.00	1.00	0.20	1	0.2	0.2	0.2
0.25	0.25	0.50				
0.25	0.75	0.80				
0.50	0.00	0.20				
0.50	0.50	1.00				
0.50	1.00	0.20				
0.75	0.25	0.80				
0.75	0.75	0.50				
1.00	0.00	0.20				
1.00	0.50	0.20				
1.00	1.00	0.20				

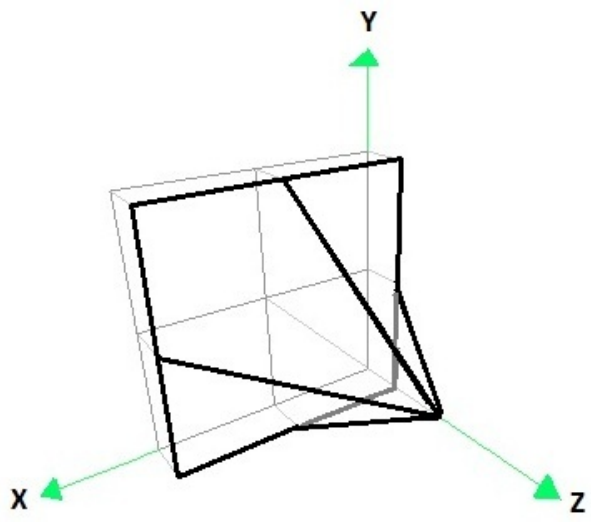
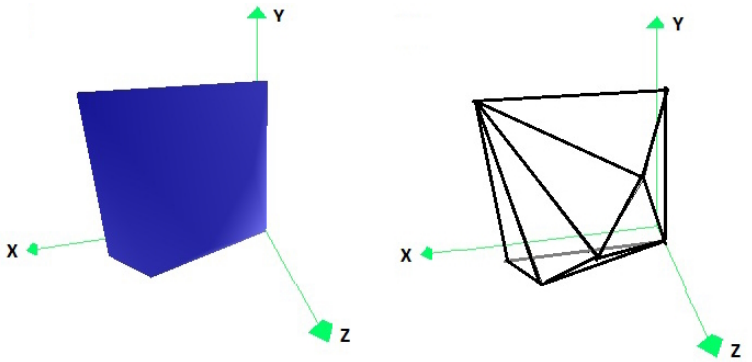
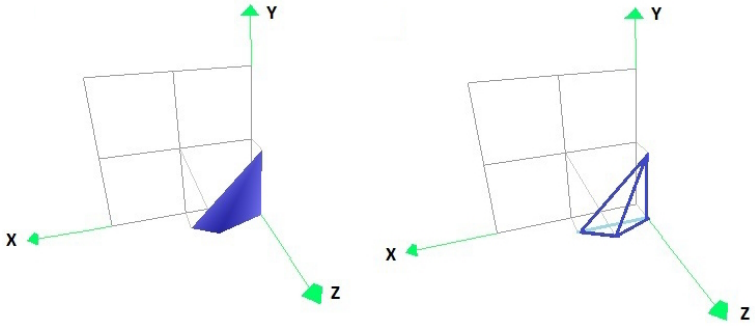


Figure 3. The original grid of interpolation points (2 × 2 partition).



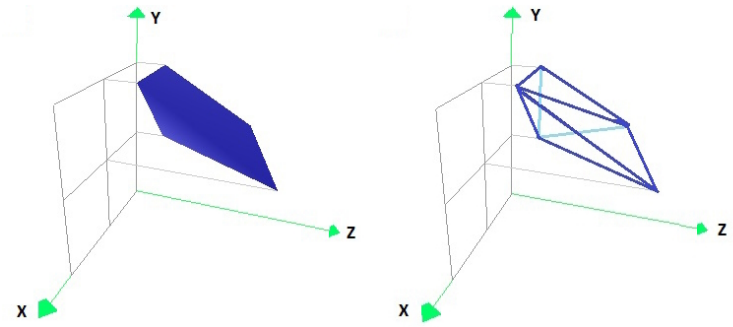
(**a**) (**b**)

Figure 4. The convex hull of the original data set (**a**) in the form of coloured vertices, and (**b**) with its edge representation.



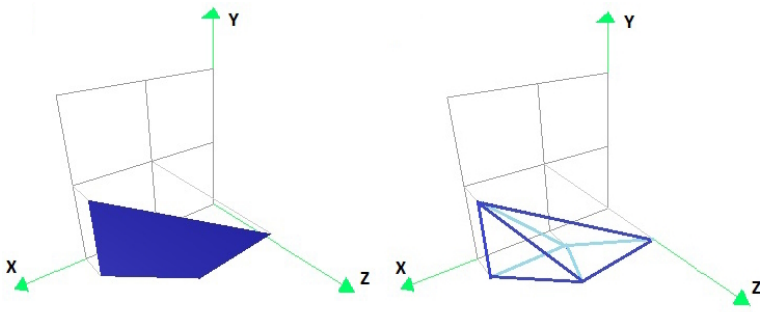
(**a**) (**b**)

Figure 5. The convex hull of the original data set $P_{1,1}$ (**a**) in the form of coloured vertices, and (**b**) with its edge representation.



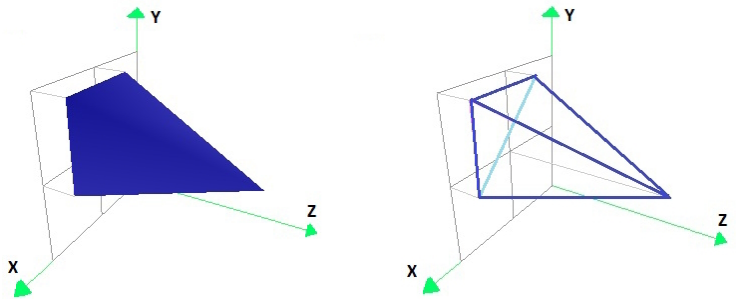
(**a**) (**b**)

Figure 6. The convex hull of the original data set $P_{1,2}$ (**a**) in the form of coloured vertices, and (**b**) with its edge representation.



(**a**) (**b**)

Figure 7. The convex hull of the original data set $P_{2,1}$ (**a**) in the form of coloured vertices, and (**b**) with its edge representation.



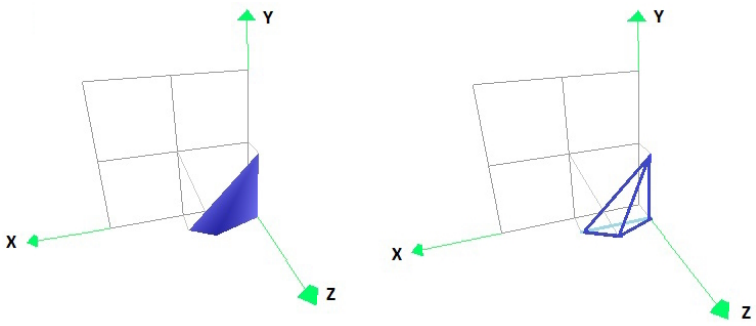
(**a**) (**b**)

Figure 8. The convex hull of the original data set $P_{2,2}$ (**a**) in the form of coloured vertices, and (**b**) with its edge representation.

By applying the algorithm for finding the optimal factors $s_{m,n}$, for $m = 1, 2$ and $n = 1, 2$, the factors listed in Table 2 are obtained. Based on these optimal factors, the transformed points for each sub-region are calculated, with their convex hulls depicted in Figures 9–12.

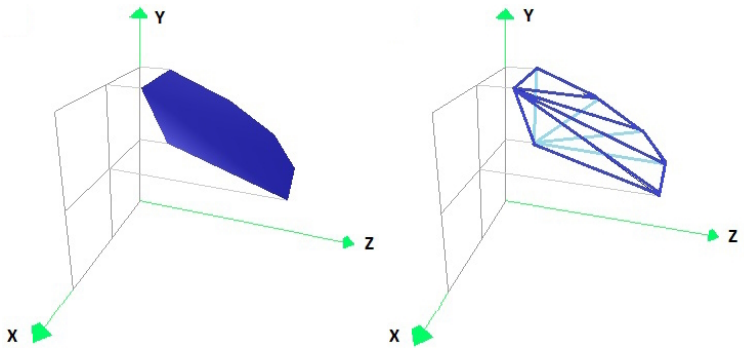
Table 2. (a) Optimum factors $s_{m,n}$ for $m = 1, 2$ and $n = 1, 2$, and (b) symmetric differences of subdomains (m, n).

(a)			**(b)**		
	m			***m***	
n	**1**	**2**	**n**	**1**	**2**
1	0.05	0.60	1	0.0	0.0125
2	0.60	0.05	2	0.0125	0.0



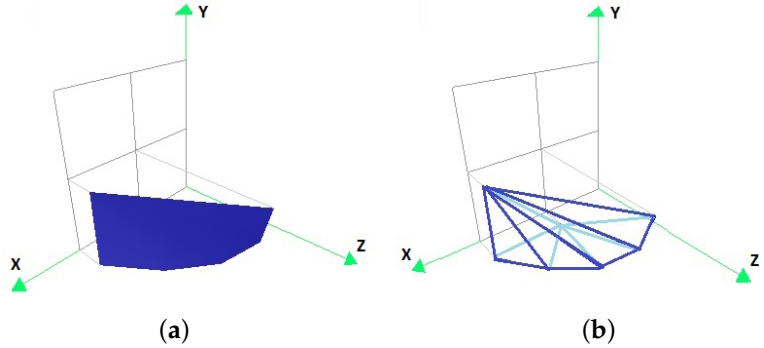
(**a**) (**b**)

Figure 9. The convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{1,1}$ (**a**) in the form of coloured faces, and (**b**) in its edge representation.



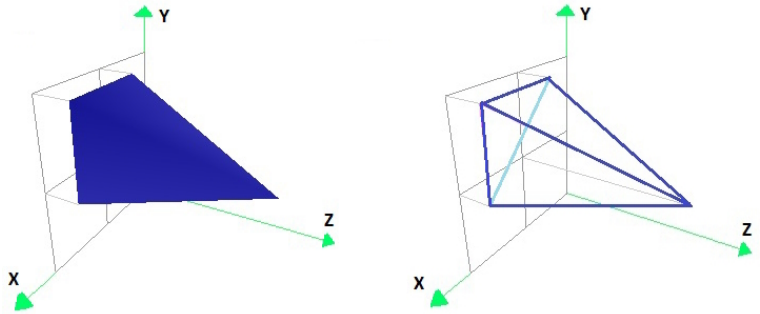
(**a**) (**b**)

Figure 10. The convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{1,2}$ (**a**) in the form of coloured faces, and (**b**) in its edge representation.



(**a**) (**b**)

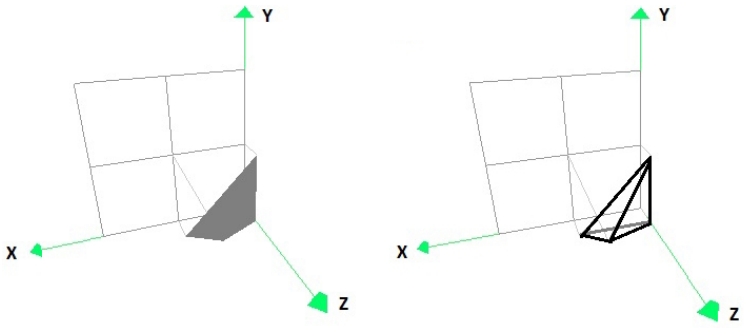
Figure 11. The convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{2,1}$ (**a**) in the form of coloured faces, and (**b**) in its edge representation.



(**a**) (**b**)

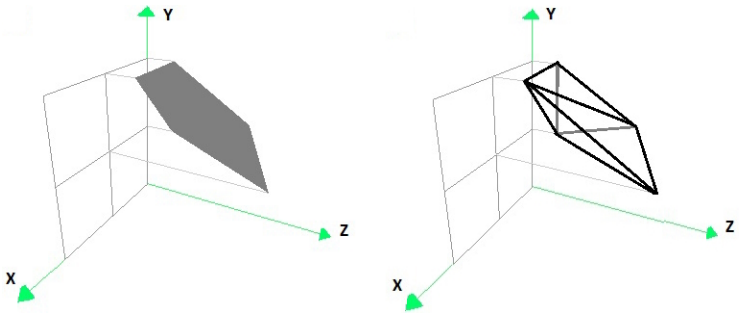
Figure 12. The convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{2,2}$ (**a**) in the form of coloured faces, and (**b**) in its edge representation.

Their intersections with the corresponding convex hulls of Pm, n, for each $m = 1, 2$ and $n = 1, 2$, are illustrated in Figures 13–16.



(**a**) (**b**)

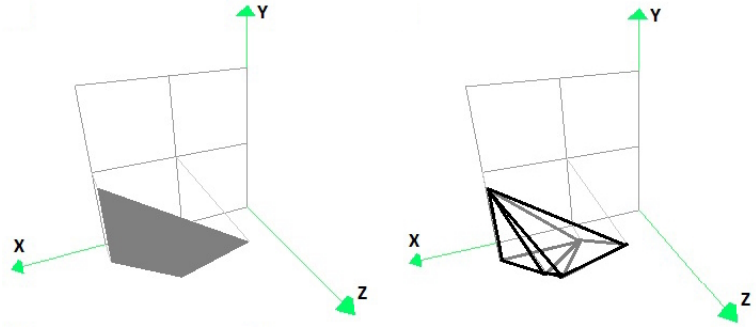
Figure 13. The intersection of the convex hull of the original data set $P_{1,1}$ with the convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{1,1}$ (**a**) in the form of coloured faces, and (**b**) with its edge representation.



(**a**) (**b**)

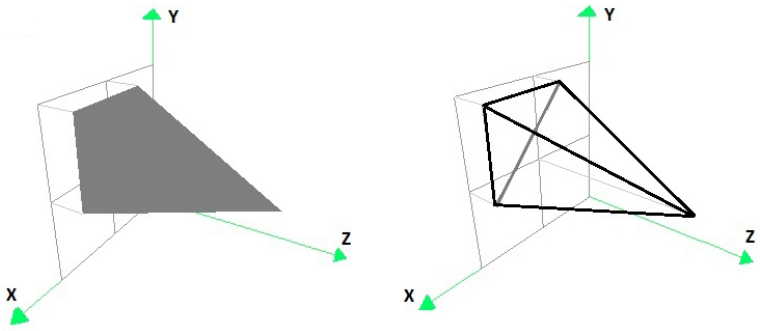
Figure 14. The intersection of the convex hull of the original data set $P_{1,2}$ with the convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{1,2}$ (**a**) in the form of coloured faces, and (**b**) with its edge representation.

The symmetric differences of the convex hulls of $P_{m,n}$, for each $m = 1, 2$ and $n = 1, 2$ with the convex hulls of the points resulting from the transformations of the vertices of the convex hull of the set P by $w_{m,n}$, for each $m = 1, 2$ and $n = 1, 2$, respectively, are presented in Table 2b. The fractal interpolation image constructed with five iterations of the DIA is shown under two different viewing angles in Figures 17 and 18.



(**a**) (**b**)

Figure 15. The intersection of the convex hull of the original data set $P_{2,1}$ with the convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{2,1}$ (**a**) in the form of coloured faces, and (**b**) with its edge representation.



(**a**) (**b**)

Figure 16. The intersection of the convex hull of the original data set $P_{2,2}$ with the convex hull of the points resulting from the transformation of the vertices of the convex hull of the set P by $w_{2,2}$ (**a**) in the form of coloured faces, and (**b**) with its edge representation.

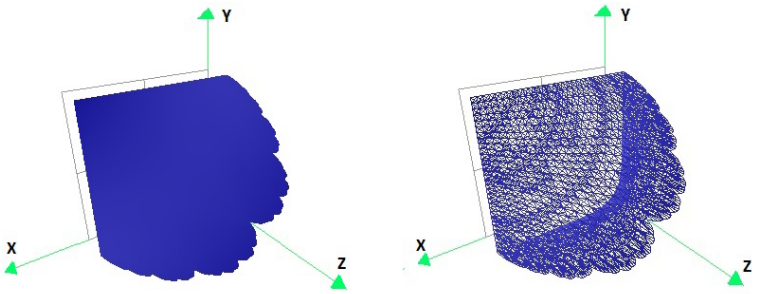


Figure 17. Fractal interpolation function for the example with 2×2 partitioning of the XY plane.

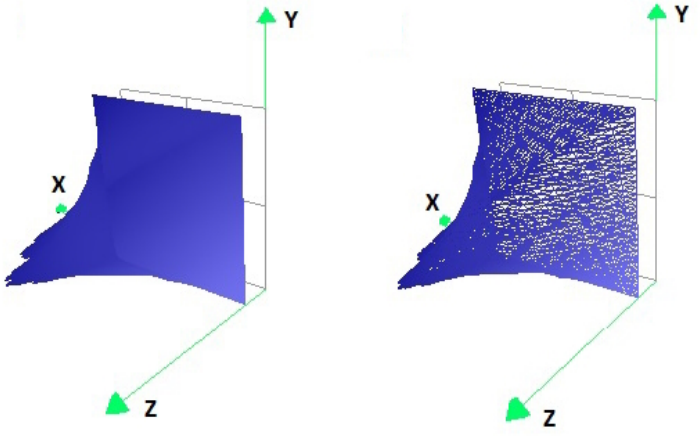


Figure 18. Fractal interpolation function for the example with 2×2 partitioning of the XY plane.

7. Conclusions

The subject of this study is the theoretical study and implementation of algorithms for convex polyhedra and their application in the field of fractal interpolation. The goal is, by finding and implementing efficient algorithms for convex polyhedra, the computation of parameters and the algorithmic construction of bivariate fractal interpolation functions using bounding volumes. Several methods for computing the convex hull of an initial set of points, calculating the volume of a convex polyhedron and finding the intersection of two convex polyhedra are presented and analysed. In the selection of a pre-existing algorithm (convex hull) as well as in the introduction of new algorithms (volume and intersection), special emphasis was placed on their efficiency.

Furthermore, we present a methodology for constructing continuous fractal interpolation surfaces, which uses the above algorithms for identifying non-free parameters, while it is based on iterated function systems. Given a set of points in three-dimensional space $\mathbb{R}^3$, a subset of them is selected as interpolation points and a fractal representation is constructed which passes through them. The identification of the parameters (free or not) of such a representation is important as is determines the quality of interpolation with respect to the initial set of points. We present a new method of identifying non-free parameters, extending in $\mathbb{R}^3$ an existing one in $R2$. The proposed methodology is based on minimising the symmetric difference between the bounding volumes of appropriately chosen points, and it achieves a lower error in the accuracy of the result.

These algorithms are applied to identify the parameters, and thus in the algorithmic construction of bivariable fractal interpolation functions by using bounding volumes. An algorithm for calculating the vertical scaling factors of fractal interpolation surfaces, such as to achieve optimal representation of the data points was proposed and implemented. In the procedure of constructing a fractal interpolation surface, the main difficulty that had to be addressed was to ensure continuity. The proposed methodology is limited to convex bounding volumes, since it is uncertain whether non-convex can be combined with efficient algorithms or improve the results. This will be the subject of our potential future research.

Author Contributions: Conceptualization, V.D.; methodology, D.M., D.S. and N.V.; software, D.M. and D.S.; formal analysis, V.D.; investigation, D.S.; writing—original draft preparation, V.D.; writing—review and editing, D.S.; visualization, N.V.; supervision, V.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barnsley, M.F. Fractal functions and interpolation. *Constr. Approx.* **1986**, *2*, 303–329. [CrossRef]
2. Barnsley, M.F. *Fractals Everywhere*, 3rd ed.; Dover Publications, Inc.: New York, NY, USA, 2012.
3. Massopust, P.R. Fractal surfaces. *J. Math. Anal. Appl.* **1990**, *151*, 275–290. [CrossRef]
4. Geronimo, J.S.; Hardin, D. Fractal interpolation surfaces and a related 2-D multiresolution analysis. *J. Math. Anal. Appl.* **1993**, *176*, 561–586. [CrossRef]
5. Zhao, N. Construction and application of fractal interpolation surfaces. *Vis. Comput.* **1996**, *12*, 132–146. [CrossRef]
6. Massopust, P.R. *Fractal Functions, Fractal Surfaces and Wavelets*; Academic Press: San Diego, CA, USA, 1994.
7. Qian, X.Y. Bivariate fractal interpolation functions on rectangular domains. *J. Comp. Math.* **2002**, *20*, 349–362.
8. Wang, H.Y. On smoothness for a class of fractal interpolation surfaces. *Fractals* **2006**, *14*, 223–230. [CrossRef]
9. Drakopoulos, V.; Manousopoulos, P. On non-tensor product bivariate fractal interpolation surfaces on rectangular grids. *Mathematics* **2020**, *8*, 525. [CrossRef]
10. Xie, H.; Sun, H.; Ju, Y.; Feng, Z. Study on generation of rock fracture surfaces by using fractal interpolation. *Int. J. Solids Struct.* **2001**, *38*, 5765–5787. [CrossRef]
11. Sun, H.; Xie, H. Study on the improved fractal interpolation surface of the attitude and surface fault. In *Thinking in Patterns*; Novak, M.M., Ed.; World Scientific: Singapore, 2004; pp. 243–254.
12. Bouboulis, P.; Dalla, L.; Drakopoulos, V. Construction of recurrent bivariate fractal interpolation surfaces and computation of their box-counting dimension. *J. Approx. Theory* **2006**, *141*, 99–117. [CrossRef]

13. Verma, S.; Viswanathan, P. Parameter Identification for a Class of Bivariate Fractal Interpolation Functions and Constrained Approximation. *Numer. Funct. Anal. Optim.* **2020**, *41*, 1109–1148. [CrossRef]
14. Manousopoulos, P.; Drakopoulos, V.; Theoharis, T. Parameter identification of 1D fractal interpolation functions using bounding volumes. *J. Comput. Appl. Math.* **2009**, *233*, 1063–1082. [CrossRef]
15. Clarkson, K.; Shor, P. Applications of random sampling in computational geometry, i. *Disc. Comput. Geom.* **1989**, *4*, 387–421. [CrossRef]
16. Barber, C.B.; Dobkin, D.P.; Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* **1996**, *22*, 469–483. [CrossRef]
17. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes in C (2nd ed.): The Art of Scientific Computing*; Cambridge University Press: Cambridge, MA, USA, 1992.