Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Representation of Gosper Curves by Means of Chain Coding

Carlos Velarde, Wendy Aguilar*, Ernesto Bribiesca

Department of Computer Science, Institute for Research in Applied Mathematics and Systems, National Autonomous University of Mexico, Ciudad de Mexico, Mexico
E-mail: weam@turing.iimas.unam.mx

Received: 12 May 2025; **Revised:** 4 July 2025; **Accepted:** 4 August 2025

Abstract: This work introduces two novel methods for constructing the Gosper curve using symbolic encodings. The first method describes how the 2D Gosper curve can be expressed using the Slope Chain Code (SCC) as a sequence of slope transitions on a hexagonal grid. The second method extends this construction to 3D by interpreting the grid as the projection of a cubic lattice along one of its body diagonals. This leads to a new 3D formulation of the Gosper curve based solely on orthogonal turns, which is encoded using the Orthogonal Direction Change Chain Code (ODCCC). These methods offer compact and invariant representations of the curve by focusing on local geometric transitions rather than absolute directions. A complexity analysis is provided for both representations, including the examination of their bit-length requirements. A comparative analysis is presented between the proposed methods and the classical encoding schemes. The analysis emphasizes key aspects such as geometric expressiveness, alphabet size, invariance properties, and encoding efficiency. Finally, we discuss potential applications of the proposed representations in spatial data reduction, topological indexing, and trajectory encoding. The proposed symbolic schemes contribute a principled framework for encoding structurally complex curves, offering both mathematical insight and practical utility.

***Keywords*:** Gosper curves, Slope Chain Code (SCC), Orthogonal Direction Change Chain Code (ODCCC), chain coding, 2D curves, 3D curves

MSC: 65L05, 34K06, 34K28

1. Introduction

The generation, representation, visualization, and analysis of complex geometric shapes have long been central to various disciplines, including computer graphics and computational geometry. In particular, fractal geometry has proven to be a powerful framework for modeling complex curves and structures that exhibit self-similarity and intricate patterns that appear in both natural and artificial systems [1]. Such patterns, ranging from coastlines and clouds to biological forms and man-made designs, highlight the unique ability of fractals to capture irregular recursive structures across multiple scales. This stands in contrast to traditional Euclidean geometry, which disregards such shapes as lacking a defined form [2]. In recent years, fractal techniques have been applied in a variety of domains. For instance, fractal geometry has been used to extend spin models in quantum systems into higher-dimensional fractal versions, allowing the reinterpretation of complex quantum phases and the derivation of new quantum error-correcting codes based on fractal symmetries and self-similar structures [3], to complement computational intelligence methods in the analysis of scale-invariant structures in neuroscience [4], and to analyze malicious code propagation in wireless sensor networks using fractal-fractional models that improve understanding of network security dynamics [5].

A notable example of a self-similar fractal with both theoretical interest and applied value is the Gosper curve, also known as "Flowsnake" [2, 6]. This curve belongs to a class of fractals capable of filling a two-dimensional space in a continuous self-avoiding pattern. Unlike other curves of this kind, it is based on a hexagonal grid rather than traditional rectangular or triangular grids, allowing for a symmetrical and efficient plane tiling. Building on the foundational properties of the original Gosper design, a family of variations known as Generalized Gosper Curves preserves self-similarity and space-filling attributes [7, 8]. These properties make them ideal for applications that require efficient coverage of a plane, such as antenna design [9–11], sensor localization [12], hierarchical spatial layout [13], as well as data encoding and compression tasks [14].

The Gosper curve is generated by recursively replicating and transforming a pattern that traces a path through seven interconnected hexagonal cells, forming the base unit known as Gosper island [15] (see Figure 1a). At each iteration, every straight segment of the previous level is replaced by a scaled and rotated copy of the base pattern, preserving self-similarity while increasing structural complexity. The progressive development of the first three levels in the recursive construction is shown in Figure 1. This recursive process can be formally captured through symbolic representations, which abstract geometric transformations into rule-based descriptions. Among these, grammar-based methods—particularly Lindenmayer systems (L-systems) [2, 15–18]—offer a compact and expressive framework for encoding the self-similar structure of the Gosper curve via iterative string rewriting rules.

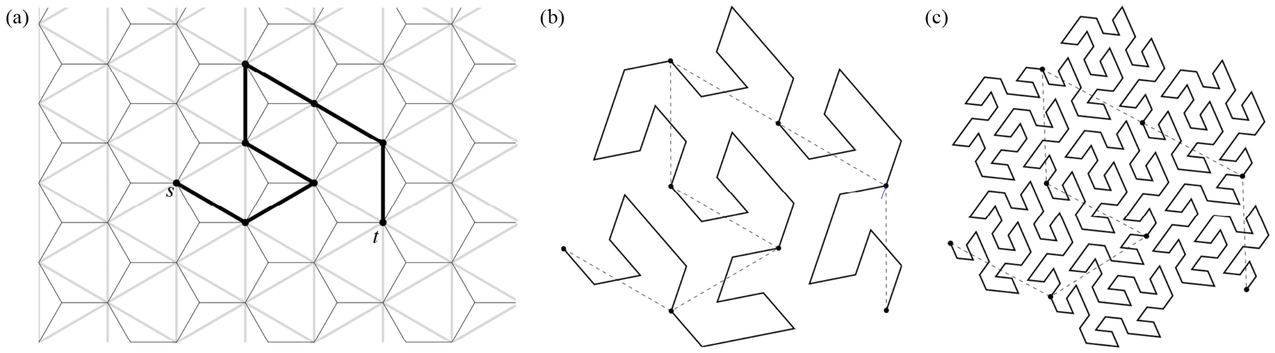

Figure 1. Illustration of the recursive generation process of the Gosper curve. (a) The basic Gosper curve is composed of seven hexagon minor diagonals corresponding to seven regular hexagons. The curve edges are also the triangle edges of a triangular grid (shadow lines). (b)-(c) Subsequent levels are generated by replacing each directional segment of the previous level with a scaled and rotated copy of the base pattern. Dotted lines indicate the base pattern, highlighting the recursive nature of the construction. This recursive method produces a continuous, self-avoiding, and space-filling curve based on a hexagonal grid

Riddle provides an alternative approach [19]. His method begins with the initial level-1 Gosper island pattern and then iteratively replaces each segment with a scaled version of the same pattern. Each substitution is reduced by a factor of $r = 1/\sqrt{7}$ and is positioned to the left of the segment it replaces. This placement ensures that the scaled pattern fits within the boundaries of the hexagon containing the edge. Repeating this recursive process indefinitely yields the fractal structure. In [15], the authors note that the construction of the Gosper curve by connecting the vertices of hexagons is not suitable for point clustering applications. To address this, they proposed an alternative procedure construction called the bottom-to-top Node-Gosper curve. This new method adheres strictly to the properties of regular hexagonal grids, ensuring that each Gosper island is composed entirely of complete, regular hexagons. A drawback of this method is its increased complexity. A different approach is presented in [14], where the authors propose a novel use of the Gosper curve to determine an efficient encoding sequence for data compression. Using the strong spatial aggregation properties of the curve, this approach reduces the amount of encoding needed. The method includes three types of encoding: a straightforward base encoding, a lossless compression using run-length encoding, and a lossy compression variant, all aimed at optimizing data storage and retrieval within spatially structured datasets. These different methods produce the same fractal but offer different flexibility in terms of computational efficiency, visual appearance, and ease of implementation.

While grammar-based methods provide an elegant mechanism for generating the Gosper curve through recursive rules and hierarchical construction, chain codes can be used as a complementary means of encoding its output as a symbolic sequence. This separation between generation and representation enables the use of compact, invariant encodings that are not naturally addressed by grammar-based methods—such as L-systems—alone. By focusing on the step-by-step traversal of a curve, chain codes are particularly well suited for analyzing spatial trajectories and comparing morphological features. One of the earliest and most influential methods is the Freeman chain code [20, 21], which encodes object boundaries as sequences of directional steps on a discrete grid. In 2D, it uses either 4- or 8-neighbor connectivity to indicate orientation, while in 3D, an extended version known employs 26 possible directions to capture volumetric contours. However, the combined effects of relying on a fixed grid, using coarse directional resolution, and being sensitive to transformations reduce its geometric fidelity. To overcome these drawbacks, the Slope Change Code (SCC) [22] was proposed. It represents shapes using contiguous, constant-length segments fitted directly to the curve, encoding slope transitions in a continuous range from -1 to 1, and achieving invariance to translation, rotation, starting point, and mirroring. These same invariance properties are also present in the Vertex Chain Code (VCC) [23], which defines boundary encodings based on the number of cell vertices that contact the contour of the shape. An alternative approach is the Mid-Crack Chain Code [24], which improves contour precision by following the cracks between pixels, allowing fast reconstruction and efficient storage of binary and grayscale images. For 3D curves, the Orthogonal Direction Change Chain Code (ODCCC) [25] uses an encoding based on five possible orthogonal direction changes—three planar and two interplanar—yielding a compact representation that is invariant to translation and rotation, and well-suited for 3D shape analysis. Its simplified 2D version, the 3OT code [26], retains only the three planar changes, supporting lossless compression and maintaining geometric invariance. Another symbolic strategy for shape representation is the enclosing tree notation [27], which represents the enclosing surface of a 3D voxel object through a tree whose vertices correspond to surface vertices. To address voxel-based 3D paths, the relative chain code was introduced in [28], which encodes directional changes using a reference, a support, and a direction-change vector within a 26-connected 3D grid. A more recent contribution is the Multi-resolution Chain Code (MrCC) [29], which builds upon the 4-connected Freeman code to generate compact, scalable shape representations, enabling progressive visualization, efficient storage, and support for tasks such as contour approximation and Geographic Information Systems (GIS) applications.

Although these diverse representations have proven effective for various 2D and 3D shapes, including voxel-based structures and binary contours, the Gosper curve remains, to the best of our knowledge, unexplored in the context of chain code encoding. Motivated by this gap, we introduce two novel methods for constructing the Gosper curve using symbolic encodings. The first method uses the SCC for 2D curves, contributing to the applicability of symbolic representations to a well-known fractal curve that had not previously been encoded in this way. The second method is based on the geometric insight that the triangular grid underlying the 2D Gosper curve (see Figure 1a) can be interpreted as the orthogonal projection of a cubic lattice in a three-dimensional space along one of its body diagonals. This perspective enables the interpretation of the Gosper curve as the projection of a spatial path embedded in 3D, which can be independently encoded using the ODCCC. These chain codes were selected for their advantages in terms of geometric fidelity, compactness, and invariance properties. The SCC employs an alphabet ranging within $(-1, 1)$ that provides fine-grained resolution for 2D representations, while the ODCCC adopts a more compact alphabet $\{0, 1, 2, 3, 4\}$, for efficient representation in 3D space. Both preserve essential geometric features and are inherently invariant to rotation and translation; they can also be adapted to account for mirroring and variations in the starting point. This robustness ensures consistent and reliable representations. The proposed methods open new avenues for incorporating the Gosper curve into domains that demand compact, scalable, and transformation-invariant representations, such as Geographic Information Systems (GIS), network topology design, and symbolic shape analysis.

This paper is organized as follows. In Section 2, we provide an overview of the key concepts underlying the SCC and the ODCCC. In Sections 3 and 4, we detail the construction and representation of the Gosper curve family at various hierarchical levels, using the SCC for 2D curves and the ODCCC for 3D curves, respectively. Section 5 presents a complexity and robustness analysis of both representations, including bit-length requirements, scalability across recursion levels, and a comparative evaluation against classical chain code schemes. Section 6 explores potential applications, with particular emphasis on scenarios that benefit from compact, invariant, and geometrically meaningful encodings. Finally, conclusions are presented in Section 7.

2. Concepts and definitions

To ensure that this paper is self-contained, in this section, we provide a concise summary of the foundational concepts of the SCC [30] and the ODCCC [25, 31]. This groundwork establishes the necessary theoretical basis for introducing the proposed methods for representing and constructing Gosper curves. We start with a sequence of general definitions and concepts outlined below.

• The term *Shape* refers to the form or outline of an object, which is regarded as a geometric entity defined by its spatial characteristics.

• A chain a is a sequence of n elements, written as $a = a_1 a_2 \ \ldots \ a_n$, and can be formally described by the indexed set $\{a_i\}_{i=1}^{n}$.

• Each element a_i of a chain encodes the local directional change between adjacent straight line segments at position i, whether in a 2D polygonal curve (SCC) or a 3D discrete curve (ODCCC). For the SCC, this change represents the slope variation or angular deviation between consecutive segments, with values normalized continuously between -1 (corresponding to a turning angle of $-180°$) and 1 (180°). In the case of the ODCCC, each element encodes an orthogonal direction change at a vertex, selected from a discrete set of labels indicating specific types of directional transitions along the 3D polygonal path.

• Each straight line segment is assigned a unit length, that is, $l = 1$.

2.1 *The SCC for 2D curves*

The Slope Chain Code (SCC) provides a symbolic representation of a 2D curve based on the angular variation between fixed-length straight segments placed along its trajectory. This is achieved by placing consecutive straight line segments of equal length such that each connects two points on the curve. At each junction between two adjacent segments, the slope change is calculated and normalized to be within the interval $(-1, 1)$, where -1 and 1 correspond to $-180°$ and 180°, respectively. The resulting ordered list of these normalized angular changes forms a numerical sequence that characterizes the geometry of the curve. This sequence is known as the SCC of the curve.

The procedure for obtaining the SCC of a curve begins with defining the length of the straight-line segments, which determines the resolution of the representation (see Figure 2a). Shorter segments capture finer details, while longer ones may overlook narrow or highly curved regions. Once the segment length is established, a point on the curve is selected as the origin (see Figure 2b). From this point, a straight segment of the predefined length is drawn to the next point on the curve, and the process is repeated, chaining segments until the entire curve is traversed (see Figure 2c). If the final segment overshoots the curve endpoint, reducing the length of the segment is an effective strategy to improve alignment.

Once the segment sequence is constructed, the slope changes between adjacent segments are normalized: non-negative changes are assigned to $[0, 1)$ and negative ones to $(-1, 0]$ (see Figure 2d). The final chain of normalized values encodes the directional evolution and geometric structure (see Figure 2e). For example, the curve in Figure 2e yields the following chain:

$$-.2 \quad -.2 \quad -.2 \quad -.2 \quad -.2 \quad -.2 \quad -.2 \quad .7 \quad 0 \quad .5 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1 \quad -.1$$

Slope changes are typically quantized using a fixed precision. In Figure 2e, each value is rounded to the nearest tenth (10^{-1}), although this granularity can be adapted. This quantization produces a finite symbolic alphabet; in this case, 19 distinct symbols, excluding the boundary values -1 and 1, which are not attained in practice.

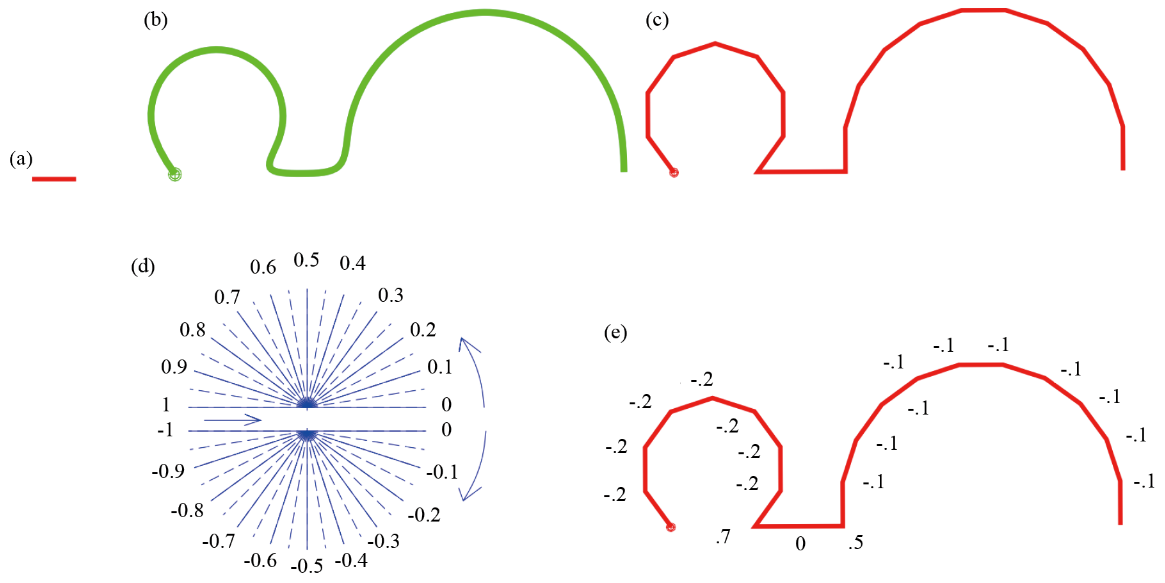

Figure 2. Steps to derive the SCC for a 2D curve: (a) Definition of a fixed-length straight segment; (b) Selection of the starting point on a continuous curve; (c) Construction of the discrete curve by chaining segments along the original; (d) Normalization intervals for positive and negative slope changes, $[0, 1)$ and $(-1, 0]$, respectively; (e) Resulting discrete curve paired with its corresponding sequence of normalized slope changes

Beyond geometric description, several curve properties can be derived from the chain. The *accumulated slope* is defined as the total sum of slope changes along the curve, while the *tortuosity* corresponds to the sum of their absolute values, quantifying the curve's overall complexity. Transformations of the chain provide further analytical tools. The *inverse chain* is obtained by reversing the order of the elements and changing their signs. This transformation is crucial to achieve invariance with respect to the starting point of the curve. Similarly, reflecting a curve across an axis produces a mirror chain, where the signs of the slope changes are inverted without altering their sequence.

The SCC is applicable to both open and simple closed curves and does not depend on any underlying grid. It is inherently invariant under translation and rotation. Scale invariance can also be introduced by normalizing the total curve length to a fixed number of segments.

Moreover, the symbolic nature of discretized SCC enables the application of formal language techniques, such as grammatical inference or pattern recognition, for curve classification and analysis. By toggling between continuous and discrete representations, researchers can tailor the method to exploit either the precision of geometric modeling or the abstraction and compactness of symbolic computation, depending on the goals of the study.

A complete review of the concepts mentioned above can be found in [30].

2.2 *The ODCCC for 3D curves*

This subsection presents an overview of the key principles behind the Orthogonal Direction Change Chain Code (ODCCC). A *3D discrete curve* is represented by a sequence of straight segments of equal length. Each pair of consecutive segments forms a direction change, and a pair of these direction changes together defines a chain element. Specifically, an element a_i in the chain, chosen from the set $\{0, 1, 2, 3, 4\}$, corresponds to a vertex on the 3D polygonal curve and encodes the orthogonal change in direction at that point [25, 31].

Figures 3a-e illustrate the labeling rules: A vertex forming a straight angle is assigned the label "0", while the right-angle vertices receive labels from the remaining numbers, depending on their spatial relationship with the previous right-angle vertex along the curve.

To determine the label at a given vertex, consider the directions of the two edges that form the reference angle as vectors b and c, each with unit length, and let d be the direction vector of the segment emanating from the vertex to be labeled (see Figure 3a). The chain element at that vertex is then computed using the following function introduced in [31]:

$$\text{chain\_element}(b, c, d) = \begin{cases} 0, & \text{if } d = c, \text{ in green;} \\ 1, & \text{if } d = b \times c, \text{ in cyan;} \\ 2, & \text{if } d = b, \text{ in blue;} \\ 3, & \text{if } d = -(b \times c), \text{ in magenta;} \\ 4, & \text{if } d = -b, \text{ in red;} \end{cases} \tag{1}$$

where, $\times$ represents the vector product in $\mathbb{R}^3$. To enhance the visualization and interpretation of the chain elements, each straight-line segment has been color-coded according to its associated chain value: segments corresponding to the element "0" are shown in green, "1" in cyan, "2" in blue, "3" in magenta, and "4" in red. This color scheme provides a clearer mapping between the numerical chain labels and their geometric representation along the curve.

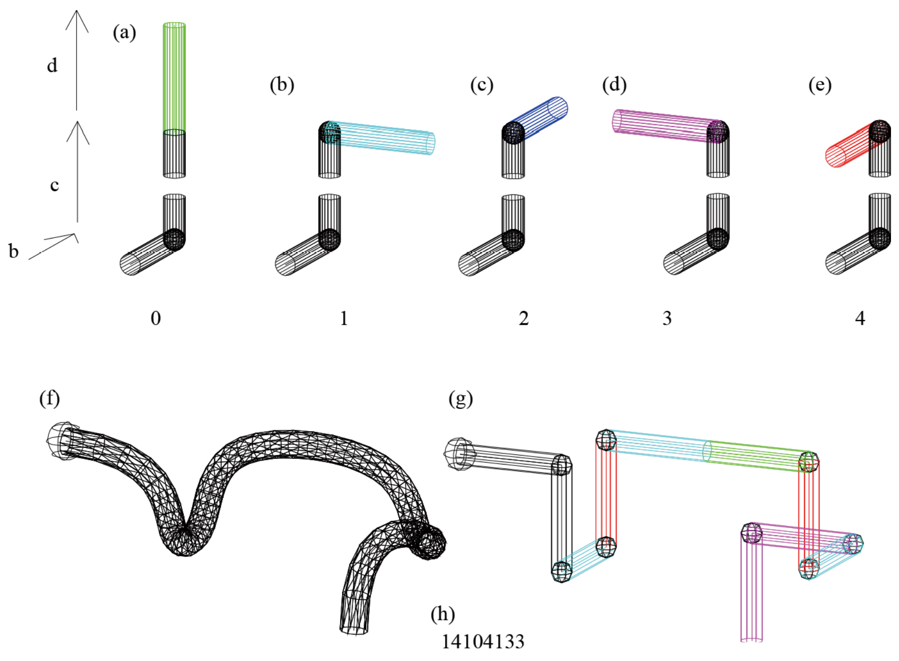

Figure 3. Illustration of the process to generate the chain representation of a 3D curve: (a)-(e) Depict the five possible orthogonal direction changes, each associated with a specific chain element and color—green for "0", cyan for "1", blue for "2", magenta for "3", and red for "4"; (f) Shows a continuous 3D curve with a marked origin; (g) Presents its discretized counterpart composed of straight-line segments; (h) Displays the resulting chain code corresponding to the discrete curve in (g)

Figure 3 illustrates the procedure for constructing the chain representation of a 3D curve. The Figure 3a through 3e define the five possible orthogonal direction changes that can occur between consecutive segments. In the Figure 3f, a continuous 3D curve with a designated origin, marked by a sphere, is shown. Its corresponding discretized version is presented in Figure 3g, where the curve is approximated by a sequence of straight-line segments. Finally, the Figure 3h displays the resulting chain code associated with the discrete curve in Figure 3g.

For further details about ODCCC, see [25, 31].

3. The Gosper curves via the SCC

The *Gosper curves* constitute a sequence $\{G_n : n \in \mathbb{N}\}$ of planar simple polygonal paths convergent to a region of the plane known as Gosper Island [2, 6]. Here, we construct a SCC representation for such a family of curves. The basic Gosper curve, G_0, is a path composed of seven edges which are minor diagonals of an equal number of regular hexagons. The curve edges are also the triangle edges of a triangular grid. See Figure 1a.

To construct the SCC representation of G_0, we assign to each pair of consecutive segments a symbol from the alphabet $\Sigma = \{-\beta, -\alpha, 0.0, \alpha, \beta\}$, where $\alpha = \pi/3$ and $\beta = 2\alpha$, based on their relative turning angle. This results in a symbolic sequence that captures the geometric structure of the curve in a direction-invariant manner.

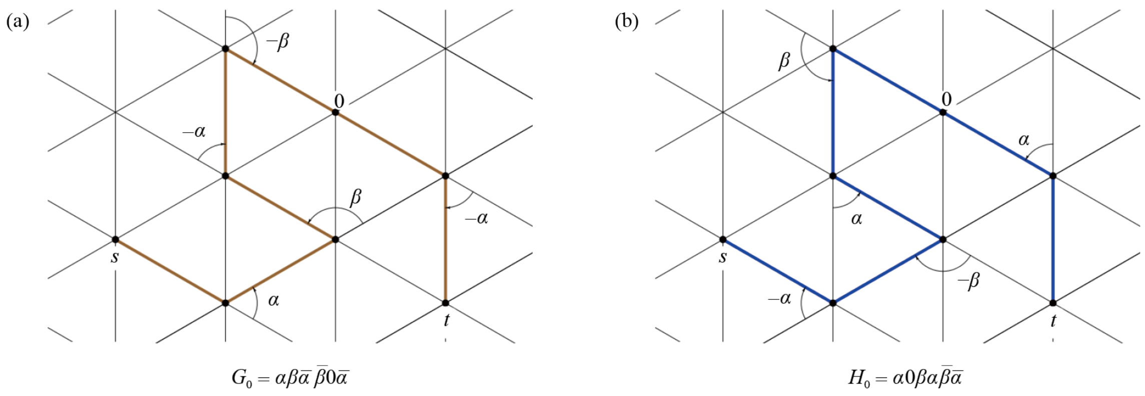
(a)
$-\beta$
0
$-\alpha$
β
$-\alpha$
s
α
t
$G_0 = \alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha}$
(b)
β
0
α
α
s
$-\beta$
$-\alpha$
t
$H_0 = \alpha0\beta\alpha\bar{\beta}\bar{\alpha}$

Figure 4. (a) The basic Gosper curve G_0 over a triangular grid. The figure highlights the direction changes along the *s*-*t* path. The direction-change sequence is written in the compact form $\alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha}$, where $\bar{\alpha} = -\alpha$ and $\bar{\beta} = -\beta$, and defines the SCC of G_0. For readability, we write expressions like $G_0 = \alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha}$. (b) The curve inverse of G_0 is the *t*-*s* path denoted by H_0 and has SCC $H_0 = \alpha0\beta\alpha\bar{\beta}\bar{\alpha}$

Let SCC(G_0) denote the slope change code associated with the curve G_0. Figure 4a shows that the sequence of slope transitions along the path from vertex s to t is given by:

$$\mathrm{SCC}(G_0) = \alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha} \tag{2}$$

where $\bar{\alpha} = -\alpha$ and $\bar{\beta} = -\beta$, and each symbol corresponds to a discrete change in slope between consecutive segments.

For readability, we adopt a notational convention in which the curve and its symbolic encoding are written interchangeably, using expressions like:

$$G_0 = \alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha}. \tag{3}$$

The inverse curve of G_0 is the path H_0 that goes from t to s in Figure 4b, its SCC is formed of the opposite-sign elements of G_0 arranged in reverse order, that is,

$$H_0 = \mathbf{inv}(G_0) = \mathbf{inv}(\alpha\beta\bar{\alpha}\bar{\beta}0\bar{\alpha}) = \alpha0\beta\alpha\bar{\beta}\bar{\alpha} \tag{4}$$

Observation 1 The accumulated slope change for G_0 and H_0 are respectively

$$\begin{aligned} \sum G_0 &= \alpha + \beta - \alpha - \beta + 0 - \alpha = -\alpha \quad \text{and} \\ \sum H_0 &= -\sum G_0 = \alpha. \end{aligned} \tag{5}$$

The second Gosper curve, G_1, is obtained by replacing the seven edges of G_0 with four replicas of G_0 and three replicas of H_0 appropriately scaled and following the order $(G_0, H_0, H_0, G_0, G_0, G_0, H_0)$, as illustrated in Figure 5. Note that the six inner vertices of the original G_0 become the join vertices for the replicas. The substitution determines the direction change of G_1 at each of these union vertices. Note that the path G_1 is located on a triangular grid different from the original one used to define G_0. However, for this work, we do not need either the triangle size or the orientation of the triangulation. What is important at this point is that the direction changes must be multiples of α (the equilateral triangle angle). Following the path G_1 over Figure 5 by simple inspection, the next values for the direction-changes in question are obtained:

$$\alpha,\ \alpha,\ -\alpha,\ -\alpha,\ \alpha,\ -\alpha. \tag{6}$$

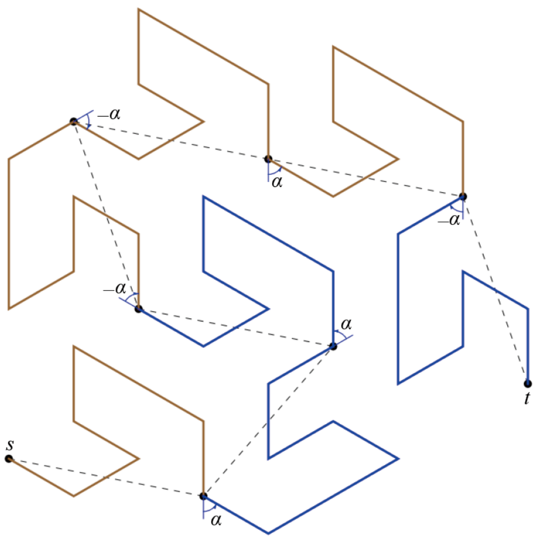

Figure 5. Gosper curve G_1. Obtained from G_0 (dashed s-t path) by replacing the seven edges oe G_0 with scaled copies of G_0, H_0, H_0, G_0, G_0, G_0 and H_0, respectively. The G_0 copies appear in brown and the H_0 copies appear in blue. The inner vertices of the original G_0 are equal to the joining vertices for the replicas and appear labeled with the direction change suffered there in going along the new path G_1

Thus, the SCC for G_1 is

$$G_1 = G_0 \alpha H_0 \alpha H_0 \bar{\alpha} G_0 \bar{\alpha} G_0 \alpha G_0 \bar{\alpha} H_0, \tag{7}$$

and for the inverse of G_1 we obtain:

$$H_1 = G_0\alpha H_0\bar{\alpha}H_0\alpha H_0\alpha G_0\bar{\alpha}G_0\bar{\alpha}H_0. \tag{8}$$

Observation 2 The accumulated slope for G_1 and H_1 are

$$\begin{aligned}\sum G_1 &= \sum(G_0\alpha H_0\alpha H_0\bar{\alpha}G_0\bar{\alpha}G_0\alpha G_0\bar{\alpha}H_0)\\ &= 4\sum G_0 + 3\sum H_0 + 3\alpha + 3\bar{\alpha} = 4(-\alpha)+3\alpha+3\alpha+3(-\alpha) = -\alpha \quad \text{and}\\ \sum H_1 &= -\sum G_1 = \alpha.\end{aligned} \tag{9}$$

The Gosper curves G_{n+1} $(n > 1)$ are defined in a similar way as for G_1, replacing the seven edges of G_0 with four replicas of G_n and three replicas of H_n appropriately scaled and following the same Gs' and Hs' pattern used before: $(G_n, H_n, H_n, G_n, G_n, G_n, H_n)$. See the case of G_2 in Figure 6.

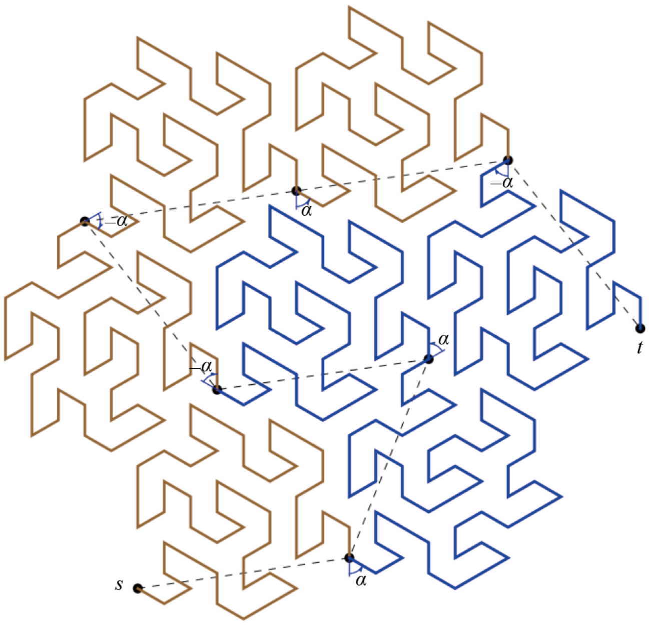

Figure 6. Gosper curve G_2. The curve G_0 is the dashed s-t path. The inner vertices of G_0 are the join vertices between the successive scaled copies of G_1 and H_1. Note that the direction changes of G_2 in such vertices coincide with those for G_1 shown in Figure 5

Correspondingly, for the SCC we have the following result.

Proposition 1 For each $n \in \mathbb{N}$,

$$\begin{aligned}&G_{n+1} = G_n\alpha H_n\alpha H_n\bar{\alpha}G_n\bar{\alpha}G_n\alpha G_n\bar{\alpha}H_n \quad \text{and}\\ &\sum G_{n+1} = -\alpha\end{aligned} \tag{10}$$

where $\bar{\alpha} = -\alpha$ and $H_n = \mathbf{inv}(G_n)$.

Proof. The proof is by induction on n. For the base case ($n = 0$) see the proofs for equations (7) and (9). For the inducive step ($n > 0$), the induction hypothesis is

$$G_n = G_{n-1}\alpha H_{n-1}\alpha H_{n-1}\bar{\alpha}G_{n-1}\bar{\alpha}G_{n-1}\alpha G_{n-1}\bar{\alpha}H_{n-1} \text{ and } \sum G_n = -\alpha; \tag{11}$$

$\sum G_n = -\alpha$ implies that the direction changes in the linking points of G_{n+1} are the same as those of G_n, therefore $G_{n+1} = G_n\alpha H_n\alpha H_n\bar{\alpha}G_n\bar{\alpha}G_n\alpha G_n\bar{\alpha}H_n$ and $\sum G_{n+1} = \sum(G_n\alpha H_n\alpha H_n\bar{\alpha}G_n\bar{\alpha}G_n\alpha G_n\bar{\alpha}H_n) = -\alpha$. □

Figure 7 illustrates the representation of the Gosper curve at the third iteration level constructed using the Slope Chain Code (SCC).

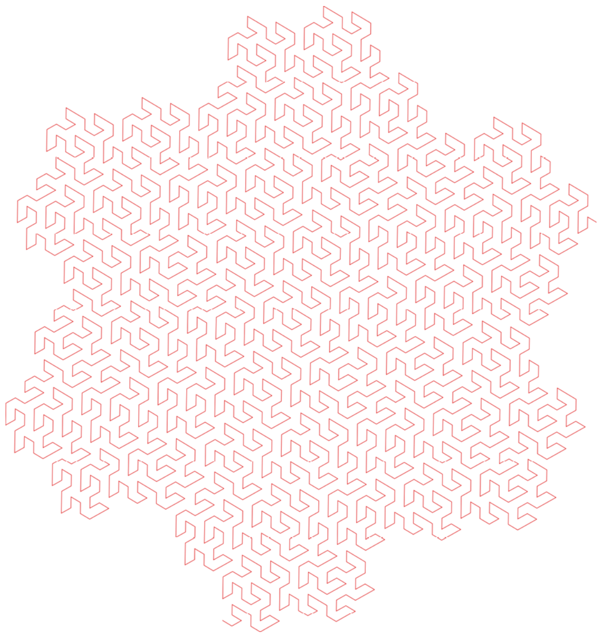

Figure 7. Gosper curve G_3

4. The Gosper curves via the ODCCC

The representation of Gosper curves using the ODCCC arises from the following pair of observations: each of such curves can be drawn on the lines of a triangular grid, and this grid can be obtained as the projected image of a corresponding cube grid.

The justification of these observations is based on the following lemma.

Lemma 1 Let $p, q \in \mathbb{R}^3$ be the vertices of a body diagonal of a cube and let $\varphi : \mathbb{R}^3 \to \mathscr{H}$ be the projection along the line pq over a plane $\mathscr{H}$ orthogonal to pq. Then, φ maps the cube edges onto the six edges of a regular hexagon with circumcenter $\varphi(p)$ plus the six radii incident in the hexagon vertices, respectively (See Figure 8).

Proof. The three vertices adjacent to p form an equilateral triangle, as do the three vertices adjacent to q. The two triangles are equal because all their edges are face diagonals of the cube. The line pq is perpendicular to each triangle at the corresponding circumcenter. Let $\mathscr{H}$ be the projection plane perpendicular to pq at the center c of the cube. So, φ sends the triangles' circumcircles to an equal circle $\mathscr{C}$ contained in $\mathscr{H}$ and with center c. The projections of the vertices of the triangles lie in $\mathscr{C}$. Each one of the other three body diagonals joins a vertex u adjacent to p with a vertex v adjacent to q. Since $c \in \overline{uv}$ and $\varphi(c) = c$, then $\varphi(\overline{uv}) = \overline{\varphi(u)\varphi(v)}$ is a diameter of $\mathscr{C}$. Hence, φ maps the vertices of the triangles onto the vertices of a regular hexagon circumscribed by $\mathscr{C}$. Furthermore, since $\varphi(p) = \varphi(q) = c$, it follows that φ projects the six edges incident to p or Q onto six radii of $\mathscr{C}$, correspondingly.

To complete the proof, observe that the two vertices adjacent to p other than u, say u' and u'', also are adjacent to v, thus the two cube edges $\overline{vu'}$ and $\overline{vu''}$ are mapped onto the hexagon edges $\overline{\varphi(v)\varphi(u')}$ and $\overline{\varphi(v)\varphi(u'')}$, respectively. This reasoning is valid for each body diagonal $\overline{uv}$, hence φ maps six cube edges onto the edges of the regular hexagon, respectively. □

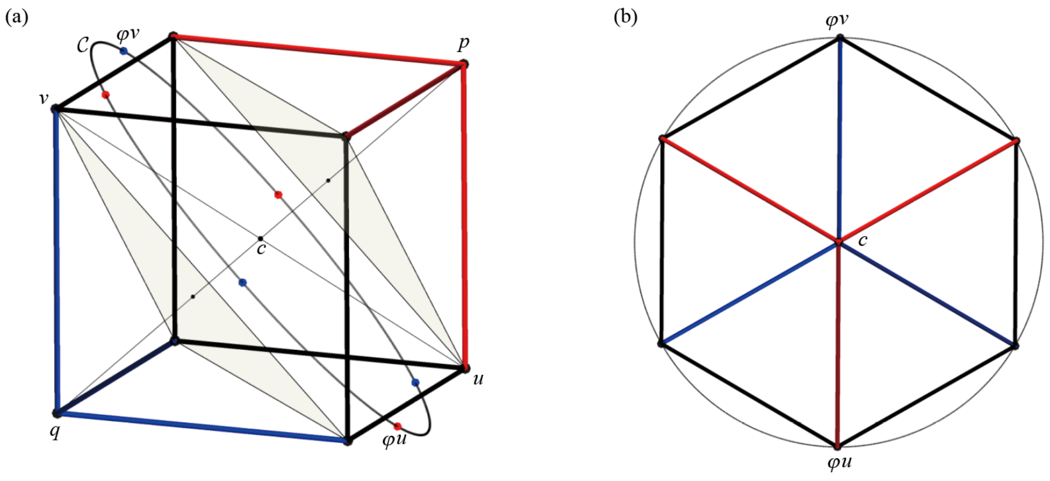

Figure 8. Projection of a cube over a plane $\mathscr{H}$ orthogonal to $\overline{pq}$. (a) The two shadow triangles are equal because all its edges are lateral diagonals of the cube; the corners of one triangle are the vertices adjacent to p and to the other triangle correspond the vertices adjacent to q. The line pq passes through the center c of the cube and is perpendicular to each triangle at the corresponding circumcenter. $\mathscr{H}$ is chosen to include c, the circumcircles of the triangles are projected by φ to the circle $\mathscr{C}$. The six points shown on $\mathscr{C}$ are the projections of the vertices of the triangles. Points u and v are vertices of another body diagonal. (b) The projection of the cube

Let us suppose that the cube of Figure 8a pertains to a cubic grid on $\mathbb{R}^3$. Then the line pq is the union of an infinite number of body diagonals of the corresponding cubes, all these cubes mapped by φ to the common image shown in Figure 8b. Analogously, any parallel line to pq through any grid vertex is the union of an infinite number of body diagonals of cubes, and all of these are projected by φ onto a common image similar to the one shown in Figure 8b.

Corollary 1 Let $\mathscr{L}$ be a cube grid on $\mathbb{R}^3$ and let $\varphi : \mathbb{R}^3 \to \mathscr{H}$ be the projection along a line ℓ over a plane $\mathscr{H}$ orthogonal to ℓ. Suppose ℓ contains a body diagonal of a cube of $\mathscr{L}$. Then, $\varphi(\mathscr{L})$ is a triangular grid on $\mathscr{H}$.

Next, we use this corollary to interpret each Gosper curve G_n as a corresponding curve X_n in $\mathbb{R}^3$ such that $\varphi(X_n) = G_n$. At the same time, we construct the ODCCC of X_n. Figure 9 illustrates the basic case, panel Figure 9a shows G_0 positioned over a triangular grid, emphasizing the pair of hexagons over which G_0 rests. These hexagons are similar to the one shown in Figure 8b. Then, considering the hexagonal pair as φ-projected cubes, changing the viewpoint a bit to appreciate all the sides of the cube, we can see in Figure 9b the curve X_0 such that $\varphi(X_0) = G_0$. From now on φ projects along lines parallel to $(1, -1, 1) = i - j + k$ and $\mathscr{H}$ is the plane perpendicular to this vector in 0; moreover, $\mathscr{H}$ contains every Gosper curve G_n with its initial vertex $s = 0$, and the first two edges of X_n with directions i and j, respectively. In this way, since curve G_n is simple X_n is well defined.

The ODCCC of X_0 is obtained by visiting the s-t path and applying (1) to obtain the ODCCC elements that label the vertices. Following the same convention adopted for the SCC, we identify the curve X_0 with its ODCCC representation and denote it as

$$X_0 = 41404 \tag{12}$$

Let Y_0 be the inverse curve of X_0, its ODCCC is constructed as done for X_0 but now following the t-s path, we obtain

$$Y_0 = 04414 \tag{13}$$

(note that it is not the X_0 ODCCC in reverse order, the element 0 appears left shifted; for ODCCC algebraic properties see [31]).

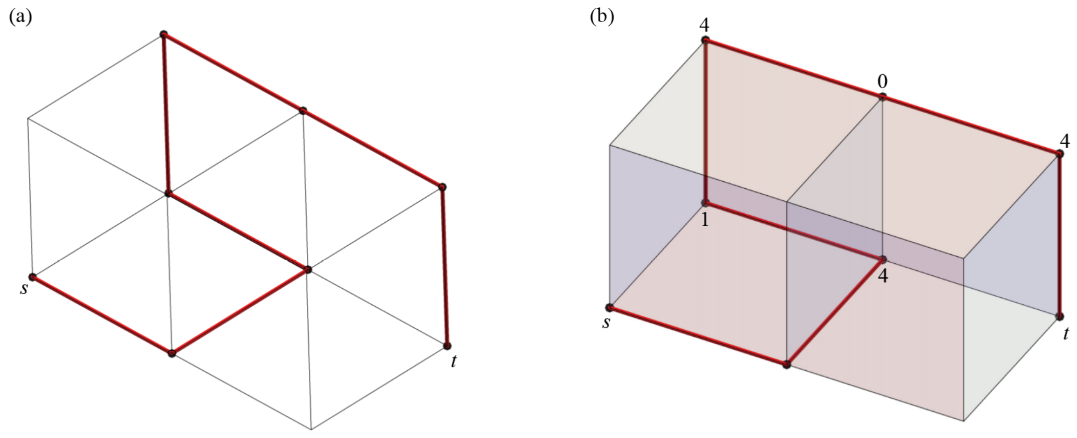

Figure 9. (a) The basic Gosper curve over a triangular grid; (b) Its interpretation as a 3D curve over a cubic grid, with vertices labeled by the corresponding ODCCC elements

Figure 10 shows the curve X_1 corresponding to G_1 (see Figure 5). It is the concatenation of four replicas of X_0 and three of Y_0 scaled appropriately, following the order X_0, Y_0, Y_0, X_0, X_0, X_0 and Y_0. The ODCCC of X_1 is obtained as was done for X_0, traveling along the s-t path and applying (1) to label each vertex with the corresponding ODCCC element, in particular each union vertex and its successor vertex. In the figure, these pairs of elements appear in boldface. Hence, the ODCCC for X_1 is

$$X_1 = X_0 23 Y_0 23 Y_0 32 X_0 32 X_0 23 X_0 32 Y_0 \tag{14}$$

and for its inverse we obtain:

$$Y_1 = X_0 23 Y_0 32 Y_0 23 Y_0 23 X_0 32 X_0 32 Y_0. \tag{15}$$

In summary, for the ODCCC we have the following result.

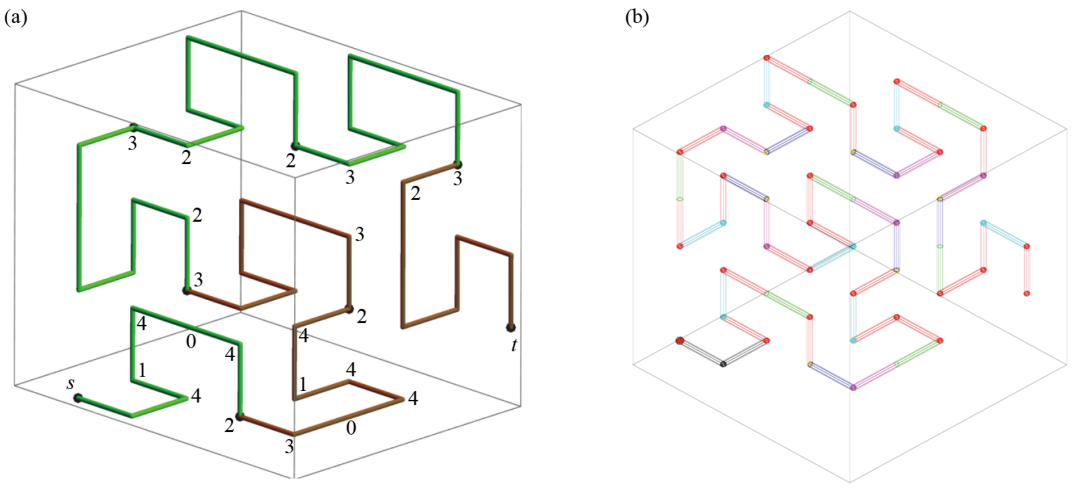

Figure 10. The Gosper curve X_1 via the ODCCC. (a) Visualization using the numeric code. The black points are the join vertices for the curves of the replicas sequence $(X_0, Y_0, Y_0, X_0, X_0, X_0, Y_0)$. $X_1 = 41404230441423044143241404324140423414043204414$. (b) Visualization using the color code as defined in Eq. (1)

Proposition 2 For each $n \in \mathbb{N}$,

$$X_{n+1} = X_n 23 Y_n 23 Y_n 32 X_n 32 X_n 23 X_n 32 Y_n \tag{16}$$

where $X_0 = 41404$ and $Y_n = \textbf{inverse}(X_n)$.

Figure 11 illustrates the case for X_2.

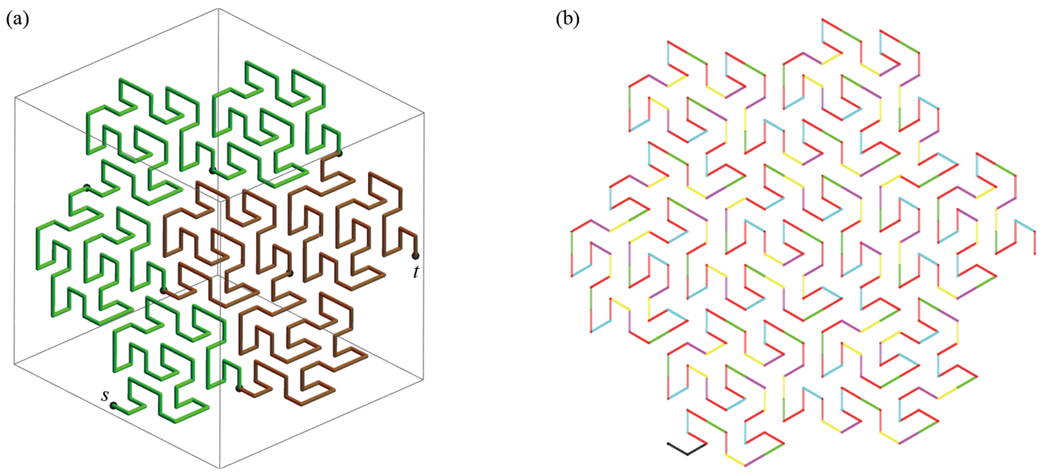

Figure 11. The Gosper curve X_2 via the ODCCC. (a) The black points are the join vertices for the curves of the replicas sequence $(X_1, Y_1, Y_1, X_1, X_1, X_1, Y_1)$, thus $X_2 = X_1 23 Y_1 23 Y_1 32 X_1 32 X_1 23 X_1 32 Y_1$. (b) Visualization using the color code as defined in Eq. (1)

5. Complexity analysis and robustness of Gosper curve representations

In this section, an analysis of the complexity and robustness associated with the representation of subsequent levels of the Gosper curve is presented, using both the SCC and the ODCCC. Specifically, the number of bits required to encode higher levels of the curve is examined, detailing how the length of the chain code increases as the curve progresses. In addition, the representational boundaries are investigated, offering insights into the limitations and scalability of the encoding method. This analysis serves to quantify the growth in complexity and the capacity of the encoding approach to capture increasingly intricate fractal structures. Finally, a comparative analysis is provided to contextualize the proposed encodings in relation to existing chain code schemes, highlighting their advantages in terms of compactness, geometric expressiveness, and invariance properties.

5.1 *Bit encoding and complexity for the SCC*

The representation of the Gosper curve using the SCC involves encoding directional changes that are defined by five angular values: 0, $\alpha = \pi/3$, $\beta = 2\alpha$, $-\alpha$, and $-\beta$, as discussed in Section 3. These angles define a uniform quantization of the slope change range from -1 to 1, dividing it into five equally spaced intervals. This ensures a uniform distribution of directional changes across the entire range of possible slope variations. To encode these five distinct slope values, a minimum of 3 bits is required, as shown in Table 1.

Table 1. Angular values used in the SCC representation of the Gosper curve, along with their corresponding normalized SCC values, number of bits required, and binary encodings

Slope (degrees)	SCC value	Bits required	Binary encoding
0	0.000	3	000
60	0.333	3	001
120	0.666	3	010
-60	-0.333	3	011
-120	-0.666	3	100

Having established the quantization scheme and the minimum number of bits required to encode each directional change, we now analyze how the overall size of the Gosper curve representation grows as the curve is recursively expanded across successive levels using the SCC.

Proposition 3 Let $\|G_n\|$ denote the number of elements in the chain G_n. Then, for all $n \in \mathbb{N}$,

$$\|G_n\| = 7^{n+1} - 1. \tag{17}$$

Proof. The proof is by induction on n. For the base case ($n = 0$), $\|G_0\| = 7^{0+1} - 1 = 6$, which is true by the definition of G_0 (see Eq. (3)). For the inductive step ($n > 0$), the induction hypothesis is $\|G_n\| = 7^{n+1} - 1$. From Proposition 1, $\|G_n\| = \|H_n\|$, $\|\alpha\| = \|\bar{\alpha}\| = 1$, and

$$\|G_{n+1}\| = \|G_n\| + \|\alpha\| + \|H_n\| + \|\alpha\| + \|H_n\| + \|\bar{\alpha}\| + \|G_n\| + \|\bar{\alpha}\| + \|G_n\| + \|\alpha\| + \|G_n\| + \|\bar{\alpha}\| + \|H_n\| \tag{18}$$

Applying the induction hypothesis:

$$\|G_{n+1}\| = 7\|G_n\| + 6 = 7(7^{n+1} - 1) + 6 = 7^{n+2} - 1. \tag{19}$$

□

This proposition shows that the length of the Gosper curve increases by a factor of 7 at each recursion level, following an exponential growth pattern. Since each slope-change value can be represented using a minimum of 3 bits (see Table 1), the total number of bits required to encode G_n is given by $3(7^{n+1} - 1)$. This bit-level analysis is crucial for evaluating the scalability of the encoding scheme as the recursion depth increases and higher-order approximations of the Gosper curve are generated. Table 2 illustrates this exponential growth in practice, showing the number of bits required to encode the curve from $n = 0$ to $n = 10$, along with their approximate sizes in bits. For instance, while the base case G_0 requires only 18 bits, the tenth level G_{10} already demands over 5.9 billion bits, equivalent to approximately 741 megabytes. These results highlight the rapidly increasing memory requirements, which may become a limiting factor in computational implementations at higher recursion levels.

Table 2. Bit-length required to encode the Gosper curve using SCC for recursion levels $n = 0$ to 10

n	SCC length ($\|G_n\|$)	Bits
0	6	18
1	48	144
2	342	1,026
3	2,400	7,200
4	16,806	50,418
5	117,648	352,944
6	823,542	2,470,626
7	5,764,800	17,294,400
8	40,353,606	121,060,818
9	282,475,248	847,425,744
10	1,977,326,742	5,931,980,226

5.2 *Bit encoding and complexity for the ODCCC*

ODCCC encodes directional transitions in a 3D discrete curve using a fixed set of five symbols: $\{0, 1, 2, 3, 4\}$. This finite symbol set is sufficient to represent the orthogonal direction changes required by the Gosper curve, which allows for a compact 3-bit representation per chain element, as presented in Table 3.

Table 3. Symbol-to-binary encoding for the ODCCC, using 3 bits per symbol to represent the five distinct direction changes

ODCCC code	Bits required	Binary encoding
0	3	000
1	3	001
2	3	010
3	3	011
4	3	100

To analyze the representational complexity of the Gosper curve under the ODCCC, we examine the recursive growth of the symbolic sequence X_n. As discussed in Section 4, the sequence X_{n+1} is constructed by combining multiple copies of X_n and Y_n, interleaved with transition symbols, where Y_n is the inverse curve of X_n. By carefully inspecting this recursive construction and tracking the number of symbols added at each level, we derive a closed-form expression for the length of X_n, which is formalized in the following proposition.

Proposition 4 Let $\|X_n\|$ denote the number of elements in the chain X_n. Then, for all $n \in \mathbb{N}$,

$$\|X_n\| = 7^{n+1} - 2. \tag{20}$$

Proof. The proof is by induction on $n \in \mathbb{N}$. For the base case $n = 0$, we have $\|X_0\| = 7^{0+1} - 2 = 5$, which is true by the definition of X_0 (see Eq. (12)). For the inductive step $n > 0$, the induction hypothesis is $\|X_n\| = 7^{n+1} - 2$. From Proposition 2, $\|X_n\| = \|Y_n\|$, and

$$\|X_{n+1}\| = \|X_n\| + 2 + \|Y_n\| + 2 + \|Y_n\| + 2 + \|X_n\| + 2 + \|X_n\| + 2 + \|X_n\| + 2 + \|Y_n\| \tag{21}$$

Applying the inductive hypothesis:

$$\|X_{n+1}\| = 7\|X_n\| + 12 = 7(7^{n+1} - 2) + 12 = 7^{n+2} - 14 + 12 = 7^{n+2} - 2 \tag{22}$$

□

Given that each symbol in the ODCCC can be encoded using 3 bits (see Table 3), the total number of bits required to represent the Gosper curve at level n is $3\|X_n\| = 3(7^{n+1} - 2)$. As in the case of SCC, this results in an exponential increase in storage requirements with each level of recursion (see Table 4).

Table 4. Growth in the number of bits required to encode the Gosper curve using the ODCCC, from $n = 0$ to $n = 10$. Approximate storage requirements are shown in MB or GB depending on the magnitude

n	ODCCC length $\|X_n\|$	Bits
0	5	15
1	47	141
2	341	1,023
3	2,399	7,167
4	16,805	50,415
5	117,647	352,941
6	823,541	2,470,623
7	5,764,799	17,294,397
8	40,353,605	121,060,815
9	282,475,247	847,425,741
10	1,977,326,741	5,931,980,223

5.3 *Comparative analysis*

This section presents a comparison of various chain code representations in terms of their ability to encode the Gosper curve. Since the curve is defined on a hexagonal grid, we focus on how effectively each scheme captures its geometry and what storage demands it entails. We begin with 2D chain codes, followed by an analysis of 3D alternatives.

Among the available encoding schemes for 2D curves, the SCC is especially appropriate for representing the Gosper curve, due to its ability to encode direction changes whose slope values vary continuously within the normalized range from -1 to 1. This flexibility is essential to capture the characteristic $\pm$ 60° and $\pm$ 120° turns inherent to the hexagonal structure of the Gosper curve. In contrast, the Freeman Chain Code [20, 21] is constrained to 4 or 8 fixed directions aligned to a square grid, making it fundamentally unable to represent non-orthogonal angles without distortion. A recent extension, the Multi-resolution Chain Code (MrCC) [29], builds on the 4-connected Freeman code to enable scalable and compact representations across multiple resolutions. However, it inherits the geometric limitations of F4, and thus remains restricted to orthogonal contours on square grids. The Vertex Chain Code (VCC) [23] operates by encoding the number of adjacent cell vertices along a shape's boundary and is robust under translation, rotation, and reflection; however, its classical formulation captures topological rather than angular transitions and does not directly encode the hexagonal direction changes vital to the Gosper curve. Similarly, the Mid-Crack Chain Code [24], designed for edge-between-pixel movements, is constrained to raster-grid geometry and does not support angular variations outside that regime. Finally, the 3OT code [26], is not suited for curves that rely on non-orthogonal angular variation or hexagonal topologies. Overall, while the Freeman Chain Code, MrCC, VCC, Mid-Crack Chain Code, and 3OT each serve well in their intended domains (raster contours, topological vertex representation, or orthogonal movement coding), none naturally handles the hexagonal structure of the Gosper curve. SCC stands alone in accurately and compactly preserving its fractal directional complexity.

In the case of the proposed 3D construction of the Gosper curve, the hexagonal geometry of its 2D representation is preserved by interpreting the triangular grid as the orthogonal projection of a cubic lattice along one of its body diagonals. This geometric reinterpretation allows the curve to be lifted into a three-dimensional space as a path composed exclusively of right-angle turns, making it particularly well suited for encoding schemes based on orthogonal direction changes, such as the ODCCC. Although the resulting 3D curve could, in principle, be encoded using the Freeman F26 code—since all orthogonal steps belong to its set of 26 possible directions—this approach introduces unnecessary overhead. The F26 code requires 5 bits per symbol to distinguish among all directions in the cubic lattice, while the ODCCC operates over a reduced alphabet of orthogonal transitions and encodes relative changes rather than absolute directions. This not only reduces the number of required bits per step to 3, but also confers invariance under global rotations and reflections. Moreover, the relative nature of the ODCCC facilitates symbolic regularity and statistical compression, making it significantly more efficient than F26 to represent structurally constrained paths such as the 3D Gosper curve. Another alternative is the relative chain code introduced in [28], which also aims to reduce orientation dependence by encoding directional changes relative to the previous step. Unlike ODCCC, however, this method operates over the full 26-connected 3D neighborhood and encodes each movement using a triplet: a reference vector, a support vector, and a direction-change vector. This allows it to represent not only face-adjacent transitions, but also edge- and vertex-connected movements, resulting in an alphabet of 25 symbols. While this approach offers a general and rotation-aware framework for 3D shape encoding, it remains overparameterized for the type of curve considered here, which requires only orthogonal turns. Consequently, for curves constrained to axis-aligned directions—such as the 3D Gosper curve proposed in this work—the ODCCC achieves comparable invariance properties with a smaller, more efficient alphabet and a simpler encoding mechanism.

6. Potential applications

The encoding strategies proposed in this work—the use of the SCC for the 2D Gosper curve and the ODCCC for its 3D formulation—offer symbolic and compact representations that preserve the geometric and topological structure of the curve, making them suitable for a variety of practical applications. In the following, we outline three potential use cases.

Spatial data reduction. The compact symbolic representations introduced in this work offer a rigorous approach to spatial data reduction in hexagonal raster environments. Previous studies have shown that traversing a hexagonal grid using the Gosper curve preserves spatial locality and enables the transformation of two-dimensional spatial structures into one-dimensional sequences, facilitating both lossless and lossy compression schemes [14]. Building on these insights, we consider that the SCC could be effectively applied to encode the trajectory of the Gosper not through absolute directions—as is typical in classical chain codes—but through local slope transitions between consecutive segments. This differential encoding reduces informational redundancy by capturing only geometric variation and avoids repeated symbols when encoding smooth or self-similar structures. Moreover, the SCC exhibits desirable invariance properties under translation, rotation, and starting point shifts, allowing structurally equivalent curve instances—e.g., from different tiles or spatial partitions—to yield identical symbolic sequences. Such invariance is especially advantageous for tasks involving compression, indexing, and pattern recognition, where spatial structure is more relevant than absolute position. In the three-dimensional setting, our proposed formulation maintains the spatial coherence and regularity of the original curve while enabling compact symbolic representation of volumetric spatial structures. Thus, the ODCCC provides an effective encoding mechanism for three-dimensional domains such as voxel-based models, layered spatial data, and simulation outputs, complementing the 2D advantages of the SCC in a broader spatial data reduction framework.

Topological indexing. The invariant properties of the proposed encodings make them well suited for tasks involving topological indexing. In this context, indexing refers not to spatial coordinates per se, but to the symbolic structure of connectivity and directional transitions that characterize a shape or trajectory. The SCC, by encoding only slope transitions rather than absolute directions, assigns identical symbolic sequences to structurally equivalent curve segments regardless of their position or orientation in the plane. Similarly, the ODCCC preserves local turning behavior in three-dimensional space, allowing consistent representation of volumetric paths or features. These properties enable efficient indexing, retrieval, and comparison of geometric patterns across datasets, supporting applications such as pattern recognition in spatial databases, symbolic search in compressed spatial formats, and redundancy elimination in hierarchical data representations.

Symbolic encoding of structured trajectories. The Gosper curve can serve as a structural framework for trajectory encoding across spatially discretized domains. Its continuous, self-avoiding traversal, and strong spatial locality make it suitable for representing movement paths that preserve neighborhood relationships, which is critical in applications such as robotics, navigation, and movement pattern analysis. The proposed the SCC enables a compact and invariant representation of such trajectories by encoding local slope transitions rather than absolute directions, thereby reducing redundancy and supporting translation- and rotation-invariant comparisons. In particular, this symbolic encoding proves useful in scenarios where trajectories follow or are projected onto a hexagonal grid, such as mobile anchor path planning in wireless sensor networks. Previous work has shown that employing a Gosper-based trajectory for anchor movement ensures uniform spatial coverage and systematic exposure of unknown sensor nodes to beacon signals [12]. Using the SCC to encode such paths could facilitate efficient transmission, storage, and reuse. Furthermore, in three-dimensional sensor deployments or volumetric environments, our ODCCC formulation supports analogous benefits by representing axis-aligned volumetric trajectories compactly and invariantly. Together, these representations provide a principled mechanism for encoding structured movement patterns in both 2D and 3D domains, supporting tasks such as spatial indexing, trajectory replay, localization, and communication optimization.

In summary, the above applications highlight that the proposed symbolic encoding methods (both the SCC and the ODCCC) are particularly useful when:

• A compact and invariant representation of a Gosper-based structure is needed.
• Spatial data are organized on a hexagonal grid (2D) or cubic lattice (3D).
• Applications require spatial locality preservation, such as in compression, indexing, or trajectory encoding.
• Structurally equivalent patterns must be matched regardless of orientation or position.
• An efficient representation of path-planning or sensor localization trajectories is required.

7. Conclusions

Two novel methods based on symbolic encodings have been introduced to construct the Gosper curve. The first uses the SCC chain code, which offers a compact and invariant encoding of the curve by capturing relative slope transitions. In parallel, a novel 3D formulation of the Gosper curve has been proposed, based on the geometric reinterpretation of the hexagonal grid as a projection of a cubic lattice. This construction allowed the encoding of the 3D curve trajectory using the ODCCC chain code, which also provides a compact and invariant representation while preserving axis-aligned structural features in volumetric environments.

A detailed complexity analysis has demonstrated that both encoding schemes—SCC and ODCCC—exhibit exponential growth in bit-length as the recursion level of the Gosper curve increases, with the number of encoded elements following a closed-form expression of the form $7^{n+1} - 1$ and $7^{n+1} - 2$, respectively. Despite this exponential behavior—an intrinsic property of fractal constructions—each method maintains a compact 3-bit representation per symbol due to the reduced and structured alphabets employed. Moreover, the proposed encodings are shown to preserve geometric and topological features critical to the Gosper curve, offering robustness across scales. A comparative analysis demonstrates that the proposed symbolic encodings—SCC and ODCCC—achieve greater compactness and structural fidelity than classical alternatives. In two dimensions, the SCC offers a more appropriate encoding for the Gosper curve than methods like the Freeman Chain Code, which are inherently limited to fixed orthogonal directions and square-grid geometries. Unlike those, the SCC captures the non-orthogonal angular transitions (e.g., $\pm 60°$ and $\pm 120°$) characteristic of the Gosper curve, resulting in a more faithful and compact representation. In three dimensions, the proposed ODCCC outperforms encodings such as Freeman F26 and relative 3D chain codes by leveraging the geometric regularity of axis-aligned turns while using a reduced symbol alphabet. This leads to lower bit-length requirements, invariance under global transformations, and improved symbolic regularity. Overall, the SCC and ODCCC provide efficient and geometrically aligned representations of the Gosper curve that are not addressed by traditional chain code formulations. In addition to offering compact and invariant representations, the proposed symbolic encodings enable the direct computation of a wide range of geometric descriptors, including tortuosity, accumulated slope, mirror and rotational symmetries, convexity, concavity, and circularity [22, 30, 32–34].

Beyond their theoretical significance, the proposed symbolic representations support practical applications such as spatial data reduction, trajectory encoding, and topological indexing—particularly in contexts where preserving spatial locality and structural invariance is critical. Future work may extend these encoding strategies to other space-filling or fractal curves and integrate them into applied frameworks for compression, indexing, and spatial analysis, thereby broadening their impact in real-world domains.

Acknowledgement

We express our deepest gratitude to our coauthor, Ernesto Bribiesca, whose invaluable contributions were fundamental to shaping the key ideas and main outcomes of this work. We deeply regret his passing during its preparation. His legacy as a brilliant researcher and esteemed colleague will continue to inspire the scientific community.

We also thank anonymous reviewers for their careful reading, insightful comments, and constructive suggestions, which significantly helped improve the quality and clarity of the manuscript.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Hastings HM, Sugihara G. *Fractals: A User's Guide for the Natural Sciences*. Oxford University Press; 1993.

[2] Mandelbrot BB. *The Fractal Geometry of Nature*. Macmillan; 1983.

[3] Devakul T, Williamson D. Fractalizing quantum codes. *Quantum*. 2021; 5: 438. Available from: https://doi.org/10.22331/q-2021-04-22-438.

[4] Livi L, Sadeghian A, Di Ieva A. Fractal geometry meets computational intelligence: Future perspectives. In: *The Fractal Geometry of the Brain*. New York, NY: Springer; 2016. p.567-580.

[5] Zarin R, Ullah N, Khan A, Humphries UW. A numerical study of a new non-linear fractal fractional mathematical model of malicious codes propagation in wireless sensor networks. *Computers & Security*. 2023; 135: 103484. Available from: https://doi.org/10.1016/j.cose.2023.103484.

[6] Gardner M. Mathematical games—in which "monster" curves force redefinition of the word "curve". *Scientific American*. 1976; 235(6): 124-133.

[7] Fukuda H, Shimizu M, Nakamura G. New Gosper space filling curves. In: *Proceedings of the International Conference on Computer Graphics and Imaging (CGIM2001)*. DC, United States; 2001. p.34-38.

[8] Akiyama J, Fukuda H, Ito H, Nakamura G. Infinite series of generalized Gosper space filling curves. In: *Discrete Geometry, Combinatorics and Graph Theory (CJCDGCGT 2005)*. Springer; 2007. p.1-9. Available from: https://doi.org/10.1007/978-3-540-70666-3_1.

[9] Spence T, Werner D. Generalized space-filling gosper curves and their application to the design of wideband modular planar antenna arrays. *IEEE Transactions on Antennas and Propagation*. 2010; 58: 3931-3941. Available from: https://doi.org/10.1109/TAP.2010.2078439.

[10] Werner D, Kuhirun W, Werner P. The Peano-Gosper fractal array. *IEEE Transactions on Antennas and Propagation*. 2003; 51: 2063-2072. Available from: https://doi.org/10.1109/TAP.2003.815411.

[11] Werner D, Kuhirun W, Werner P. A new class of modular broadband arrays based on Gosper islands and associated Peano-Gosper curves. In: *2003 IEEE Topical Conference on Wireless Communication Technology*. Honolulu, HI, USA: IEEE; 2003. p.267-268. Available from: https://doi.org/10.1109/WCT.2003.1321518.

[12] Chen C, Wang S. Node-Gosper curve-based unknown sensor localization using single mobile anchor in wireless sensor networks. *International Journal of Distributed Sensor Networks*. 2016; 12. Available from: https://doi.org/10.1177/155014775780101.

[13] Auber D, Huet C, Lambert A, Renoust B, Sallaberry A, Saulnier A. GosperMap: Using a Gosper curve for laying out hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*. 2013; 19(11): 1820-1832. Available from: https://doi.org/10.1109/TVCG.2013.91.

[14] Xin R, Ai T, Ai B. Encoding and compressing hexagonal raster data by the Gosper curve. *Transactions in GIS*. 2019; 23: 1204-1231. Available from: https://doi.org/10.1111/tgis.12569.

[15] Uher V, Gajdos P, Snasel V. Towards the Gosper space filling curve implementation. In: *2017 3rd IEEE International Conference on Cybernetics (CYBCON)*. IEEE; 2017. p.1-8. Available from: https://doi.org/10.1109/CYBConf.2017.7985819.

[16] Prusinkiewicz P. Graphical applications of L-systems. In: *Proceedings of Graphics Interface '86/Vision Interface '86*. Canada: Canadian Information Processing Society; 1986. p.247-253.

[17] Bader M. *Space-Filling Curves: An Introduction with Applications in Scientific Computing*. Springer; 2012.

[18] Szilard AL. Programming examples of space-filling curves. In: *The Role of Theory in Computer Science*. World Scientific; 2017. p.245-272.

[19] Riddle LH. *Classic Iterated Function Systems*. Agnes Scott College; 2024. Available from: https://larryriddle.agnesscott.org/ifs/ksnow/flowsnake.htm [Accessed 12th November 2024].

[20] Freeman H. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*. 1961; EC-10(2): 260-268. Available from: https://doi.org/10.1109/TEC.1961.5219197.

[21] Freeman H. Computer processing of line-drawing images. *ACM Computing Surveys (CSUR)*. 1974; 6(1): 57-97. Available from: https://doi.org/10.1145/356625.356627.

[22] Bribiesca E. A geometric structure for two-dimensional shapes and three-dimensional surfaces. *Pattern Recognition*. 1992; 25(5): 483-496. Available from: https://doi.org/10.1016/0031-3203(92)90047-M.

[23] Bribiesca E. A new chain code. *Pattern Recognition*. 1999; 32(2): 235-251. Available from: https://doi.org/10.1016/S0031-3203(98)00132-0.

[24] Shih FY, Wong WT. Reconstruction of binary and gray-scale images from mid-crack code descriptions. *Journal of Visual Communication and Image Representation*. 1993; 4(2): 121-129. Available from: https://doi.org/10.1006/jvci.1993.1011.

[25] Bribiesca E. A chain code for representing 3D curves. *Pattern Recognition*. 2000; 33(5): 755-765. Available from: https://doi.org/10.1016/S0031-3203(99)00093-X.

[26] Sanchez-Cruz H, Rodriguez-Dagnino RM. Compressing bilevel images by means of a three-bit chain code. *Optical Engineering*. 2005; 44(9): 097004. Available from: https://doi.org/10.1117/1.2052793.

[27] Bribiesca E, Guzmán A, Martínez LA. Enclosing trees. *Pattern Analysis and Applications*. 2012; 15(1): 1-17. Available from: https://doi.org/10.1007/s10044-011-0240-z.

[28] Sánchez-Cruz H, López-Valdez HH, Cuevas FJ. A new relative chain code in 3D. *Pattern Recognition*. 2014; 47(2): 769-788. Available from: https://doi.org/10.1016/j.patcog.2013.08.010.

[29] Nerat A, Strnad D, Zalik KR, Zalik B. An efficient multi-resolution chain coding. *IEEE Access*. 2024; 12: 54721-54731. Available from: https://doi.org/10.1109/ACCESS.2024.3389062.

[30] Bribiesca E. A measure of tortuosity based on chain coding. *Pattern Recognition*. 2013; 46: 716-724. Available from: https://doi.org/10.1016/j.patcog.2012.09.017.

[31] Bribiesca E, Velarde C. A formal language approach for a 3D curve representation. *Computers & Mathematics with Applications*. 2001; 42(12): 1571-1584. Available from: https://doi.org/10.1016/S0898-1221(01)00263-2.

[32] Aguilar W, Bribiesca E. Symmetry detection in 3D chain coded discrete curves and trees. *Pattern Recognition*. 2015; 48(4): 1420-1439. Available from: https://doi.org/10.1016/j.patcog.2014.09.024.

[33] Alvarado-Gonzalez M, Aguilar W, Garduño E, Velarde C, Bribiesca E, Medina-Bañuelos V. Mirror symmetry detection in curves represented by means of the Slope Chain Code. *Pattern Recognition*. 2019; 87: 67-79. Available from: https://doi.org/10.1016/j.patcog.2018.10.002.

[34] Aguilar W, Alvarado-Gonzalez M, Garduño E, Velarde C, Bribiesca E. Detection of rotational symmetry in curves represented by the slope chain code. *Pattern Recognition*. 2020; 107: 107421. Available from: https://doi.org/10.1016/j.patcog.2020.107421.