Noise injection into Freeman chain codes

Luka Lukač[1], Andrej Nerat[1], Damjan Strnad[1], Ivana Kolingerová[2] and Borut Žalik[1]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia
[2] Department of Computer Science and Engineering, University of West Bohemia, Pilsen, Czech Republic

ABSTRACT

This article presents a novel method for direct noise injection into geometric shapes described by eight-or four-directional Freeman chain codes. Noise is applied to randomly selected segments of a chain code sequence using a set of predefined actions. The design of alterations retains topological characteristics of shapes. The method is tested on various shapes, including open, self-intersecting, and simple shapes, among which the latest two may contain holes. Fractal dimension and mean distance from original are utilised in order to analyse the amount of injected noise in sequences of chain codes. The proposed method enables efficient noise injection directly into Freeman chain codes for use in data augmentation and regularization during neural network training.

Subjects Algorithms and Analysis of Algorithms, Multimedia, Natural Language and Speech, Optimization Theory and Computation, Software Engineering
Keywords Algorithm, Geometric shape, Boundary alteration, Fractal Dimension

INTRODUCTION

Noise typically represents an unwanted component of a signal, and dedicated denoising techniques have been developed to remove noise from different types of data (*Xu et al., 2016*; *Ali, El-Dahshan & Yahia, 2017*). On the other hand, there are also important applications for deliberate injection of noise into data, such as adding noise into training data of artificial neural networks. Noise injection is an established data augmentation and regularisation concept at artificial neural networks used to avoid occurrences of local minimum phenomena during the training phase (*Wang & Principe, 1999*) and to prevent overfitting when dealing with digitised shapes (*Yin et al., 2015*; *Jothimani & Premalatha, 2022*), and effectively increasing their robustness (*Sharma, Grau & Fritz, 2016*). Furthermore, some state-of-the-art methods also inject noise into internal components of neural networks in order to enhance their performance (*Duan, Ren & Duan, 2022*; *Chen et al., 2022*; *Zhang et al., 2023*).

Chain coding is a simple and efficient method for describing digitised geometric shapes and binary images (*Strnad et al., 2024*). Its usage can be found in industry (*Honi et al., 2022*), medicine (*Aldemir et al., 2023*; *Fanti et al., 2025*), computer vision (*Althobaiti & Lu, 2017*; *Boukharouba & Bennia, 2017*; *Zhao & Liu, 2020*; *Wang & Wang, 2022*), and efficient representations of tree models (*Strnad et al., 2019*). Furthermore, despite the fact that chain codes already provide a compact and storage-efficient representation of shapes

Submitted 27 February 2025
Accepted 5 July 2025
Published 7 August 2025

Corresponding author
Luka Lukač, luka.lukac@um.si

Academic editor
Robert Winkler

Additional Information and Declarations can be found on page 21

DOI **10.7717/peerj-cs.3084**

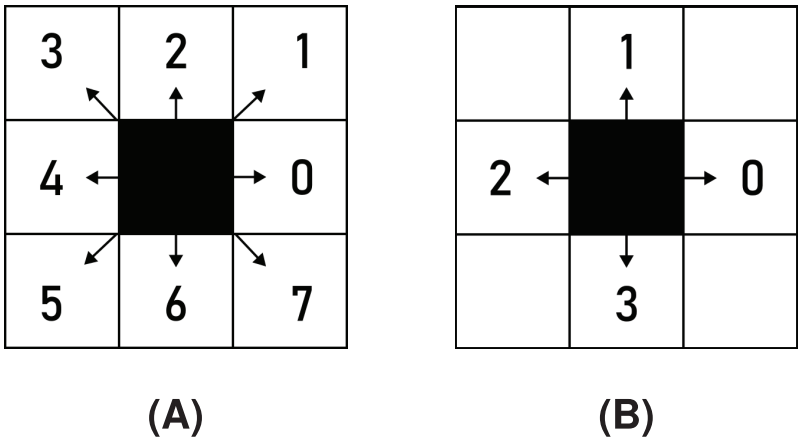

Figure 1 **Freeman chain code symbols: (A) F8, (B) F4.** Full-size DOI: 10.7717/peerj-cs.3084/fig-1

(*Sánchez-Cruz, Bribiesca & Rodríguez-Dagnino, 2007*; *Tapia-Dueñas, López & Sánchez-Cruz, 2025*), they can be compressed even further using entropy encoding methods (*Žalik et al., 2018*, *2021*). Particular efficiency of such techniques was achieved in cartoon (*Jeromel & Žalik, 2020*) and binary image compression (*Zahir, Dhou & George, 2007*).

Injecting noise into chain codes without the need to decode (*i.e.*, rasterise) and re-encode them would significantly improve the efficiency of the process, and enable better control over the preservation of geometric shapes' topological characteristics. If noise is injected into decoded raster images, there is no control over topological characteristics of geometric shapes. In the case of machine learning (ML), the changes of topological characteristics can significantly impact the classification accuracy. The main advantages of the proposed approach are the preservation of topological characteristics of geometric shapes and noise injection efficiency. To the best of our knowledge, there exist no methods for injecting noise directly into chain codes. In this article, we propose the first method for direct injection of noise into geometric shapes described by Freeman chain codes. The main limitation of the proposed method is its limitation to the Freeman chain codes as other chain code types would require different sets of noise injection rules.

Chain codes represent the shape's boundary by a sequence of consecutive directional commands. Various types of chain codes have been developed (*Bribiesca, 1999*; *Sánchez-Cruz & Rodríguez-Dagnino, 2005*; *Kabir, 2015*; *Žalik et al., 2016*). The most recent chain coding approaches include agent-based modelling and biological computing for encoding binary images (*Dhou & Cruzen, 2019*; *Dhou, 2020*; *Dhou & Cruzen, 2021*), and multi-resolution chain codes (*Nerat et al., 2024*; *Žalik, 2024*). However, the pioneer chain codes were proposed as far back as 1961 (*Freeman, 1961*). Fundamentally, two versions of Freeman chain codes were introduced. The first one is the eight-directional chain code (F8), where the shape boundary is described using symbols from the alphabet $\sum_{F8} = \{0, 1, 2, 3, 4, 5, 6, 7\}$. The angle of movement for each symbol $c \in \Sigma_{F8}$ in the raster space (according to the x-axis) is calculated as $\phi = 45° \times c$ (see Fig. 1A). The

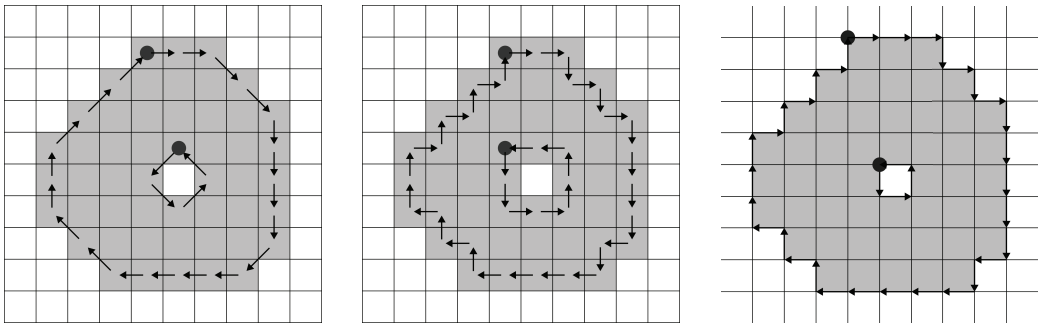

Figure 2 Encoding of a shape by Freeman chain codes: (A) F8, (B) F4 (centre), (C) F4 (crack). The start of each chain code sequence is marked by a filled circle.
Full-size DOI: 10.7717/peerj-cs.3084/fig-2

four-directional Freeman chain code (F4) uses the same principles. It, however, consists of a smaller alphabet containing just four symbols: $\sum_{F4} = \{0, 1, 2, 3\}$. Consequently, the angle of movement for symbol $c \in \sum_{F4}$ is calculated as $\phi = 90° \times c$. Possible movements of the F4 chain code are shown in Fig. 1B.

While the F8 chain code traverses through centres of boundary pixels (see Fig. 2A), the F4 chain code can be used in two ways. The first option moves along the centres of boundary pixels (Fig. 2B), while the second operates along boundary pixels' edges. It is often referred to as a crack code (*Wilson, 1997*) (see Fig. 2C).

Closed shapes can contain holes, therefore, they are described by one or more chain code sequences. The outer boundary of the shape is often referred to as a loop, while the other, inner boundaries, are considered rings (*Mortenson, 1997*). An open shape, however, consists of only one chain, also known as a path. The proposed method is not limited in regards to characteristics of shapes. It works with either open or closed, self-intersecting, or non-self-intersecting shapes, while preserving their original topological characteristics (number of rings and number of self-intersections).

The main contributions of the article are:

- The first method for noise injection directly into eight-and four-directional Freeman chain codes, preserving the topological characteristics of chain code sequences.
- The design of alteration rulesets for both types of Freeman chain codes, which define noise injection with local perturbations of chain code sequences.
- Experimental work, which demonstrates the efficacy of the proposed method on a set of open, self-intersecting, and simple geometric shapes that may contain holes, and the analysis of the injected noise with fractal dimension and mean distance from original (*Rogelj, Hudej & Petric, 2013*).

METHOD

Let $\mathscr{S}$ be a geometric shape described by a set of chains $\mathscr{C} = \{C_i\};\ 0 \leq i < |\mathscr{C}|$. $|\mathscr{C}|$ denotes the number of chain code sequences, required to describe the boundary of $\mathscr{S}$. If $\mathcal{S}$ is a

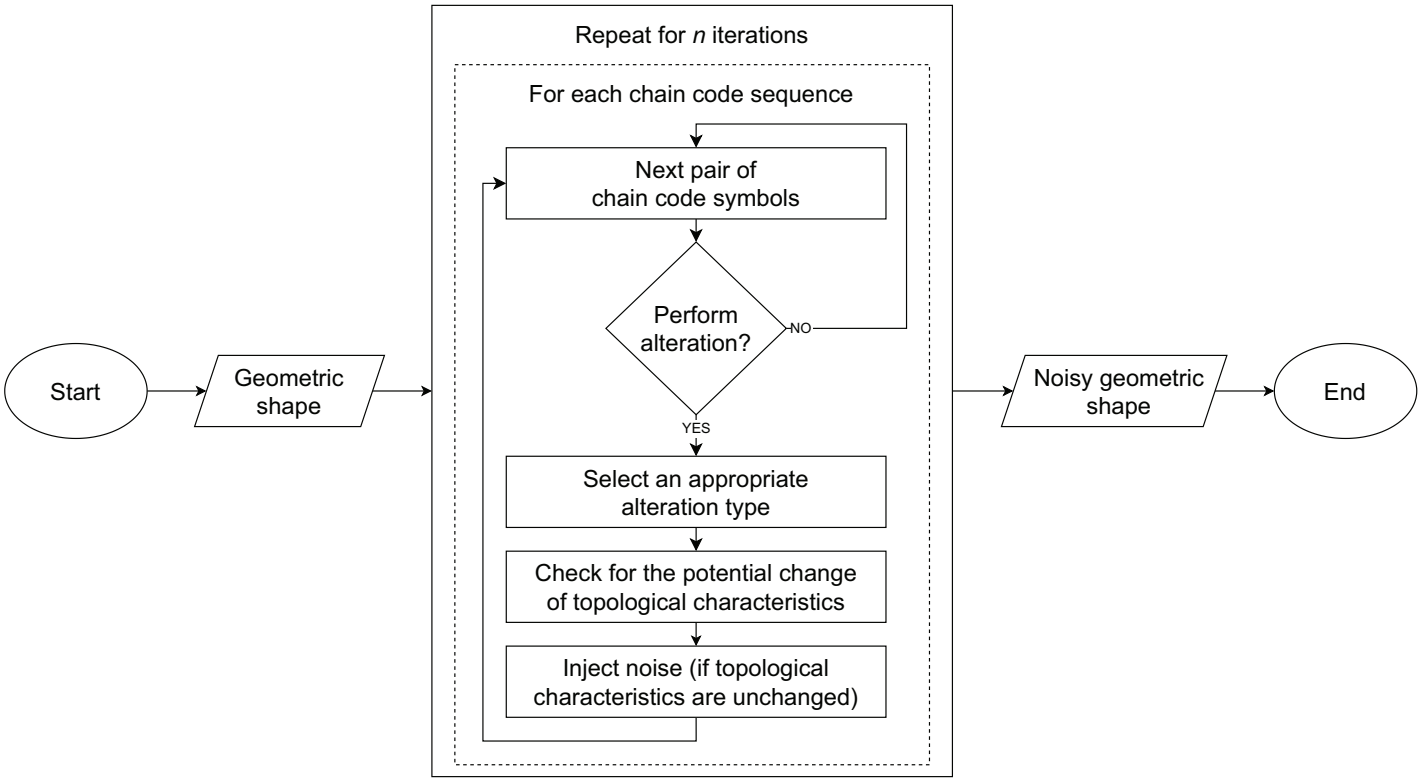

Figure 3 **Block diagram of the noise injection method.** Full-size DOI: 10.7717/peerj-cs.3084/fig-3

closed shape, $C_0 \in \mathscr{C}$ represents a loop, while $C_i \in \mathscr{C};\ 1 \leq i \leq |\mathscr{C}|$ represent rings (*i.e.*, holes). Otherwise, if $\mathcal{S}$ is an open shape, it consists of only $C_0 \in \mathscr{C}$, which represents a path. Each chain consists of a sequence of chain code symbols $C_i = \langle c_{i,j} \rangle;\ c_{i,j} \in \sum_{\{\mathrm{F8/F4}\}};\ 1 \leq j \leq |C_i|$. If an F4 chain code sequence is given as a crack code sequence, in our case, it is first converted to a format that encodes movements between centres of boundary pixels. The block diagram that outlines the sequence of steps during noise injection is shown in Fig. 3.

Noise is introduced separately to each C_i, where randomly chosen chain codes are injected with noise according to a set of predefined rules. The rules ensure that a loop or a ring is not converted to a path, and vice versa. An example of noise injection to an F8 chain code sequence is shown in Fig. 4.

The noise injection method for a geometric shape described with Freeman chain code sequences is explained in Algorithm 1. As input, the algorithm receives a set of all chains $\mathscr{C}$, type of the provided chain code sequences, probability of an alteration μ (an arbitrary value from the interval $(0, 1]$), and a number of noise injection iterations n (an arbitrary positive integer value). μ and n are user-defined parameters. The loop in line 9 iterates through all chain code symbols $c_{i,j} \in C_i$. In line 12, the altered chain code sequence section A is obtained according to randomly selected chain code symbol $c_{i,j}$ and its descendant $c_{i,j+1}$. The alteration is performed according to replacement rules defined later in this section.

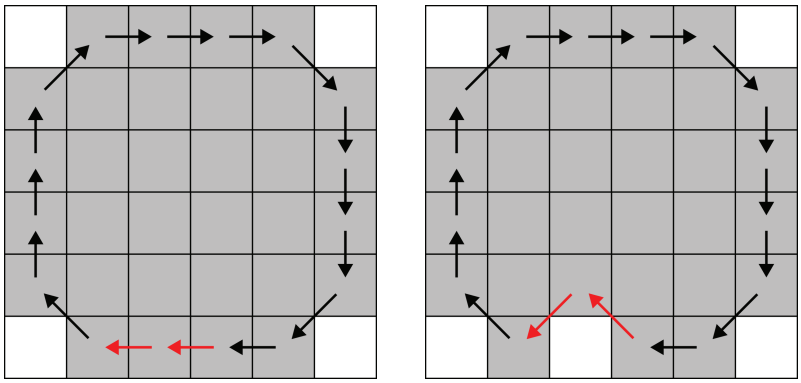

Figure 4 **Noise injection into the boundary of the shape denoted with an F8 loop: (A) before the noise injection (the red arrows indicate a pair of chain codes where noise shall be injected), (B) after the noise injection (the red arrows indicate the altered pairs of chain codes).**
Full-size DOI: 10.7717/peerj-cs.3084/fig-4

Algorithm 1 **Noise injection into S described by Freeman chain code sequences.**

function NOISE-INJECTION ($\mathscr{C}$, *type*, μ, n)
▷ $\mathscr{C}$: chain code sequences
▷ *type*: {F8/F4}
▷ $\mu \in (0, 1]$: probability of an alteration
▷ $n > 0$: number of iterations
▷ Returns: geometric shape with noisy chain code sequences
  for $l \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to $|\mathscr{C}|$ **do**
      for $j \leftarrow 1$ to $|C_i| - 1$ **do**
        $r \leftarrow$ Random(0, 1)
        if $r < \mu$ **then**
          $A \leftarrow$ Alteration($c_{i,j}$, $c_{i,j+1}$, *type*)
          if PreservedTopologicalCharacteristics($\mathscr{C}$, A) **then**
            C_i = InjectNoise(j, C_i, A)
          end if
        end if
      end for
    end for
  end for

  return $\mathscr{C}$
end function

Afterwards, if an alteration does not change the amount of self-intersections in the chain C_i or the amount of intersections with another chain $C_j \in \mathscr{C}$, $c_{i,j}$ and $c_{i,j+1}$ are replaced by A in line 14. The set $\mathscr{C}$ of chain code sequences with injected noise is obtained as the result of the algorithm after the function is repeated for all chain code sequences for n times.

Noise injection method operates using the same principles for both types of Freeman chain code sequences. The sole difference lies in alteration type selection, where separate alteration rulesets have been designed for F8 and F4 chain codes according to their specifics. The rulesets for both types of alterations were designed in a way that retains the property of openness of a chain code sequence after the noise is injected.

Alteration options for F8

There are seven different alteration options when injecting noise into F8 chain code sequences:

- *REMOVE*,
- *PUSH*,
- *WRAP*,
- *FLIP*,
- *PULL*,
- *CUT*,
- *ROTATE*.

The alterations are selected based on $c_{i,j}$ and $c_{i,j+1}$ and are formally given in Eq. (1). In the right column, there are various conditions that define the selection of the appropriate alteration in the left column.

$$f(c_{i,j}, c_{i,j+1}) = \begin{cases} f_{REMOVE}(); & |c_{i,j} - c_{i,j+1}| = 4 \\ f_{PUSH}(c_{i,j}); & (c_{i,j} = c_{i,j+1}) \wedge (c_{i,j} \bmod 2 = 0) \\ f_{WRAP}(c_{i,j}); & (c_{i,j} = c_{i,j+1}) \wedge (c_{i,j} \bmod 2 = 1) \\ f_{FLIP}(c_{i,j}, c_{i,j+1}); & (|c_{i,j} - c_{i,j+1}| = 1) \vee \\ & (|c_{i,j} - c_{i,j+1}| = 7) \\ f_{PULL}(c_{i,j}, c_{i,j+1}); & ((|c_{i,j} - c_{i,j+1}| = 2) \vee \\ & (|c_{i,j} - c_{i,j+1}| = 6)) \wedge \\ & (c_{i,j} \bmod 2 = 1) \\ f_{FLIP/CUT}(c_{i,j}, c_{i,j+1}); & ((|c_{i,j} - c_{i,j+1}| = 2) \vee \\ & (|c_{i,j} - c_{i,j+1}| = 6)) \wedge \\ & (c_{i,j} \bmod 2 = 0) \\ f_{ROTATE/CUT}(c_{i,j}, c_{i,j+1}); & (|c_{i,j} - c_{i,j+1}| = 3) \vee \\ & (|c_{i,j} - c_{i,j+1}| = 5) \end{cases}. \quad (1)$$

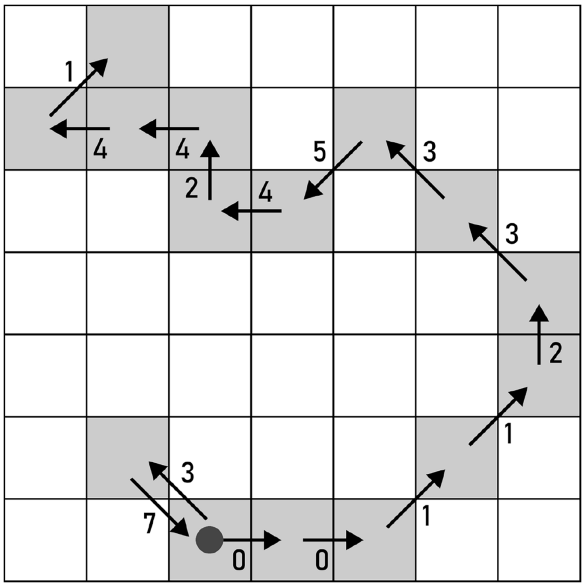

Figure 5 A path used for demonstrating the alterations for F8 chain codes. The start of the path is marked by a filled circle. Full-size DOI: 10.7717/peerj-cs.3084/fig-5

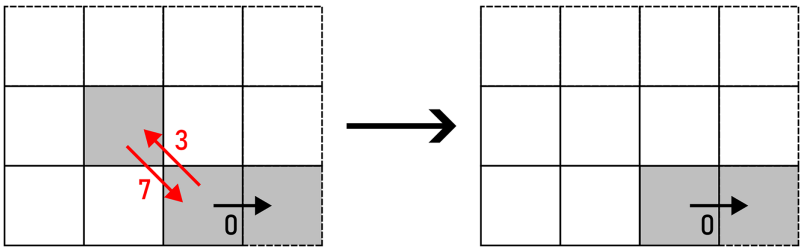

Figure 6 REMOVE applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 0011233542441. Full-size DOI: 10.7717/peerj-cs.3084/fig-6

The effects of all types of alterations will be demonstrated using the path in Fig. 5, whose C_i consists of the following chain code symbols: 370011233542441.

REMOVE erases a pair of $c_{i,j}$ and $c_{i,j+1}$, as displayed in Fig. 6.

$$f_{REMOVE}() = \langle\rangle. \tag{2}$$

PUSH operates on a pair of equal consecutive chain code symbols, representing a horizonal or a vertical segment. They are transformed in a way that a turn is obtained as given in Eq. (3) and shown in Fig. 7. A turn can be produced in two directions: up and down (horizontal segment), or left and right (vertical segment), chosen on a random basis.

$$f_{PUSH}(c_{i,j}) = \begin{cases} \langle (c_{i,j}+1) \bmod 8, (c_{i,j}+7) \bmod 8 \rangle \\ \langle (c_{i,j}+7) \bmod 8, (c_{i,j}+1) \bmod 8 \rangle. \end{cases} \tag{3}$$

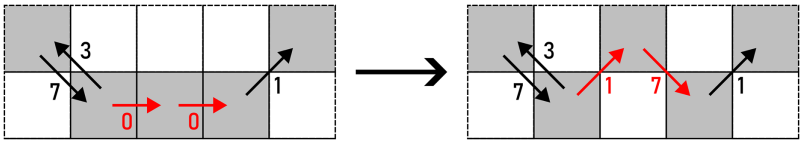

Figure 7 PUSH applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 371711233542441. Full-size DOI: 10.7717/peerj-cs.3084/fig-7

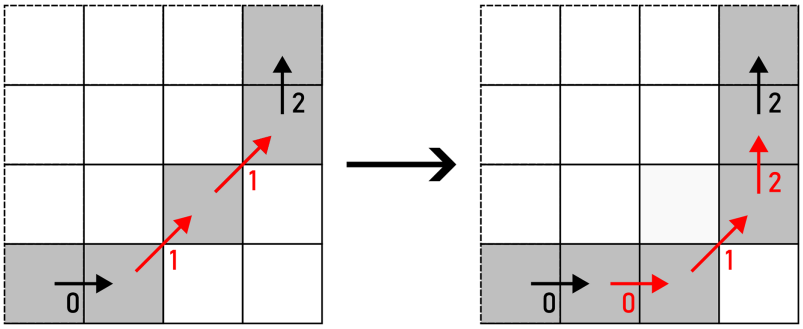

Figure 8 WRAP applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 3700012233542441. Full-size DOI: 10.7717/peerj-cs.3084/fig-8

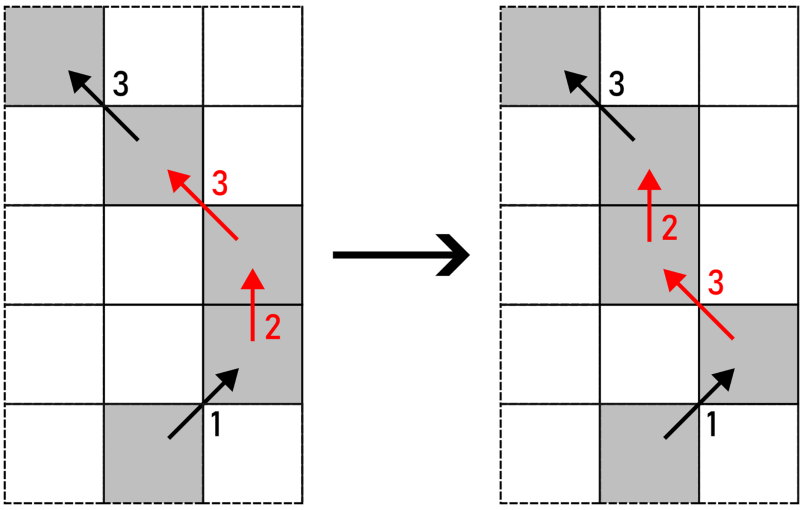

Figure 9 FLIP applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 370011323542441. Full-size DOI: 10.7717/peerj-cs.3084/fig-9

WRAP contains two options to apply an alteration to pairs of diagonal steps in the same direction: up and down, which are selected on a random basis. The action is given in Eq. (4) and shown in Fig. 8.

$$f_{WRAP}(c_{i,j}) = \begin{cases} \langle (c_{i,j}+1) \bmod 8, c_{i,j}, (c_{i,j}+7) \bmod 8 \rangle \\ \langle (c_{i,j}+7) \bmod 8, c_{i,j}, (c_{i,j}+1) \bmod 8 \rangle. \end{cases} \tag{4}$$

FLIP performs a transformation where $c_{i,j}$ and $c_{i,j+1}$ switch their roles, as given in Eq. (5). Consequently, a left turn is transformed into a right one and vice versa as displayed in Fig. 9.

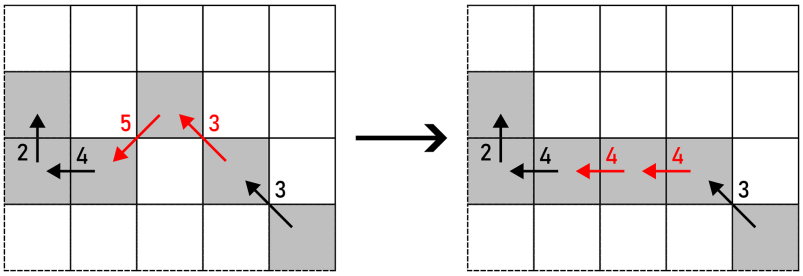

Figure 10 PULL applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 370011234442441. Full-size DOI: 10.7717/peerj-cs.3084/fig-10

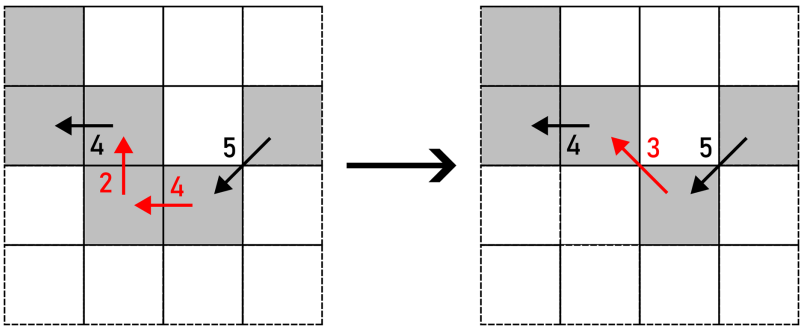

Figure 11 CUT applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 37001123353441. Full-size DOI: 10.7717/peerj-cs.3084/fig-11

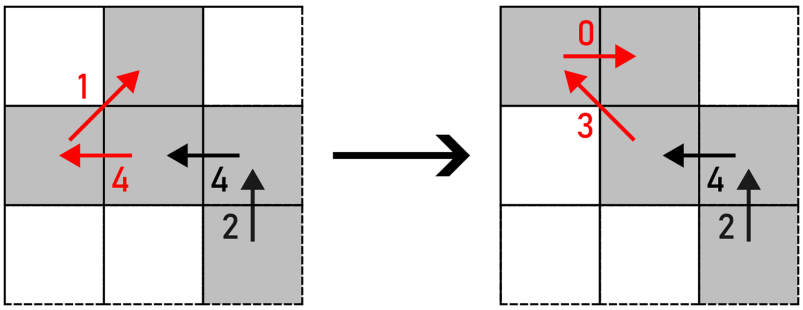

Figure 12 ROTATE applied to a pair of F8 chain code symbols (marked with red arrows). Example: 370011233542441 → 370011233542430. Full-size DOI: 10.7717/peerj-cs.3084/fig-12

$$f_{FLIP}(c_{i,j}, c_{i,j+1}) = \langle c_{i,j+1}, c_{i,j} \rangle. \tag{5}$$

PULL represents an opposite operation to *PUSH*, as two same consecutive chain code symbols are obtained from a turn. The latter is given in Eq. (6) and displayed in Fig. 10.

$$f_{PULL}(c_{i,j}, c_{i,j+1}) = \begin{cases} \left\langle \frac{c_{i,j}+c_{i,j+1}}{2} \bmod 8, \frac{c_{i,j}+c_{i,j+1}}{2} \bmod 8 \right\rangle; & |c_{i,j} - c_{i,j+1}| = 2 \\ \left\langle \frac{(c_{i,j}+c_{i,j+1}) \bmod 8}{2}, \frac{(c_{i,j}+c_{i,j+1}) \bmod 8}{2} \right\rangle; & |c_{i,j} - c_{i,j+1}| = 6. \end{cases} \tag{6}$$

CUT is a type of alteration that transforms $c_{i,j}$ and $c_{i,j+1}$ into a single connection according to specific conditions (right column) given in Eq. (7) (see Fig. 11).

$$f_{CUT}(c_{i,j}, c_{i,j+1}) = \begin{cases} \langle \frac{c_{i,j}+c_{i,j+1}}{2} \rangle; & |c_{i,j}, c_{i,j+1}| = 2 \\ \langle \lceil \frac{c_{i,j}+c_{i,j+1}}{2} \rceil \rangle; & |c_{i,j} - c_{i,j+1}| = 3 \wedge \\ & ((c_{i,j} + c_{i,j+1} = 3) \vee \\ & (c_{i,j} + c_{i,j+1} = 7) \vee \\ & (c_{i,j} + c_{i,j+1} = 11)) \\ \langle \lfloor \frac{c_{i,j}+c_{i,j+1}}{2} \rfloor \rangle; & |c_{i,j} - c_{i,j+1}| = 3 \wedge \\ & ((c_{i,j} + c_{i,j+1} = 5) \vee \\ & (c_{i,j} + c_{i,j+1} = 9)) \\ \langle (\max\{c_{i,j}, c_{i,j+1}\} + 1) \bmod 8 \rangle; & |c_{i,j} - c_{i,j+1}| \neq 3 \wedge \\ & ((c_{i,j} + c_{i,j+1} = 5) \vee \\ & ((c_{i,j} + c_{i,j+1} = 6) \vee \\ & (c_{i,j} + c_{i,j+1} = 9)) \\ \langle (\min\{c_{i,j}, c_{i,j+1}\} - 1) \bmod 8 \rangle; & |c_{i,j} - c_{i,j+1}| \neq 3 \wedge \\ & c_{i,j} + c_{i,j+1} = 7 \quad . \end{cases} \tag{7}$$

The final alteration of F8 chain code is *ROTATE*, which rotates $c_{i,j}$ and $c_{i,j+1}$ by 45°. An example of such action is given in Eq. (8) and shown in Fig. 12.

$$f_{ROTATE}(c_{i,j}, c_{i,j+1}) = \begin{cases} \langle (c_{i,j} + 1) \bmod 8, (c_{i,j+1} + 1) \bmod 8 \rangle; & (c_{i,j} + c_{i,j+1} = 3) \vee \\ & (c_{i,j} + c_{i,j+1} = 7) \vee \\ & (c_{i,j} + c_{i,j+1} = 11) \\ \langle (c_{i,j} + 7) \bmod 8, (c_{i,j+1} + 7) \bmod 8 \rangle; & (c_{i,j} + c_{i,j+1} = 5) \vee \\ & (c_{i,j} + c_{i,j+1} = 9) \quad . \end{cases} \tag{8}$$

Alteration options for F4

In contrast to F8, F4 has only three possible alterations:

- *WRAP*,
- *REMOVE*,
- *FLIP*.

The ruleset of all possible alterations is given in Eq. (9). In the continuation, the alterations are given and displayed on the path shown in Fig. 13.

$$f(c_{i,j}, c_{i,j+1}) = \begin{cases} f_{WRAP}(c_{i,j}); & c_{i,j} = c_{i,j+1} \\ f_{REMOVE}(); & |c_{i,j} - c_{i,j+1}| = 2 \\ f_{FLIP}(c_{i,j}, c_{i,j+1}); & \text{otherwise} \quad . \end{cases} \tag{9}$$

The effects of alterations will be demonstrated using the path in Fig. 13, whose C_i consists of the following chain code symbols: 000113.

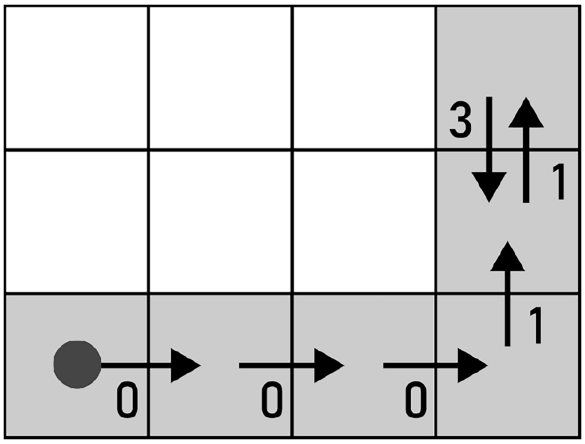

Figure 13 A path used for demonstrating the alterations for F4 chain codes. The start of the path is marked by a filled circle. Full-size DOI: 10.7717/peerj-cs.3084/fig-13

WRAP transforms a straight line into a U-turn as per Eq. (10). There are two options to perform this action: up and down (horizontal segment), or left and right (vertical segment), as shown in Fig. 14. The alteration is chosen on a random basis.

$$f_{WRAP}(c_{i,j}) = \begin{cases} \langle (c_{i,j}+1) \bmod 4, c_{i,j}, c_{i,j}, (c_{i,j}+3) \bmod 4 \rangle \\ \langle (c_{i,j}+3) \bmod 4, c_{i,j}, c_{i,j}, (c_{i,j}+1) \bmod 4 \rangle \end{cases} \tag{10}$$

As per its name, *REMOVE* removes $c_{i,j}$ and $c_{i,j+1}$ according to Eq. (11) (see Fig. 15).

$$f_{REMOVE}() = \langle\rangle \tag{11}$$

FLIP flips the roles of $c_{i,j}$ and $c_{i,j+1}$. As a result, a left turn is transformed into a right turn and vice versa, as shown in Eq. (12) and Fig. 16.

$$f_{FLIP}(c_{i,j}, c_{i,j+1}) = \langle c_{i,j+1}, c_{i,j} \rangle \tag{12}$$

Preservation of topological characteristics

In the final step before accepting an alteration, potential changes of the topological genus in the variated $\mathscr{S}$ are checked. The routine verifies whether the replacement of $c_{i,j}$ and $c_{i,j+1}$ with A would change the number of holes and self-intersections in the observed shape. In that case, such alteration is not accepted. Example of an alteration that changes the topological characteristics of $\mathscr{S}$ can be seen in Fig. 17.

In another case, if $\mathscr{S}$ already contains self-intersections before an alteration is performed, the same procedure is utilised when injecting noise into $\mathscr{S}$. The topological characteristics of $\mathscr{S}$ are preserved in this case as well. Self-intersections that are present in $\mathscr{S}$ in the beginning are preserved no matter the amount of noise injection iterations. An example of an alteration that changes the topological genus of a self-intersecting $\mathscr{S}$ is shown in Fig. 18.

Implementation-wise, chain code sequences $C_i \in \mathscr{C}$ are implicitly rasterised before noise injection since the topological characteristics are observed in raster space. The pixels

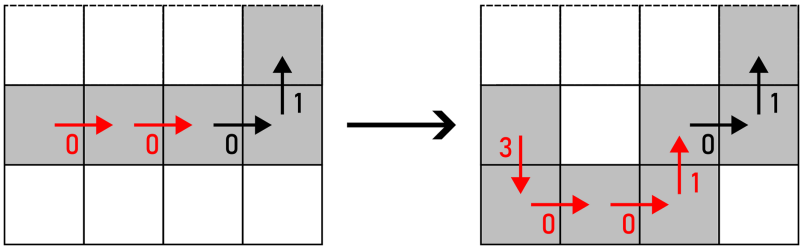

Figure 14 WRAP applied to a pair of F4 chain code symbols (marked with red arrows). Example: 000113→30010113. Full-size DOI: 10.7717/peerj-cs.3084/fig-14

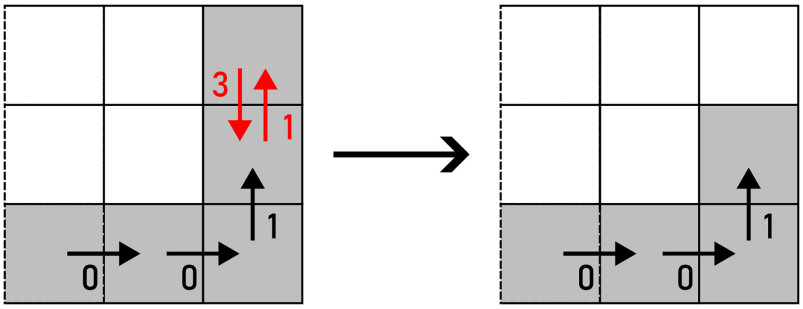

Figure 15 REMOVE applied to a pair of F4 chain code symbols (marked with red arrows). Example: 000113→0001. Full-size DOI: 10.7717/peerj-cs.3084/fig-15

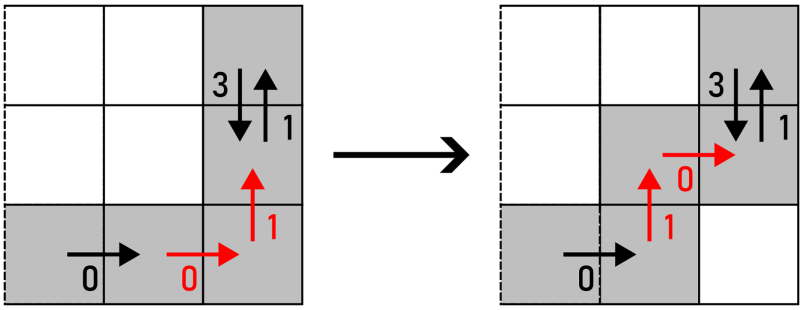

Figure 16 FLIP applied to a pair of F4 chain code symbols (marked with red arrows). Example: 000113→001013. Full-size DOI: 10.7717/peerj-cs.3084/fig-16

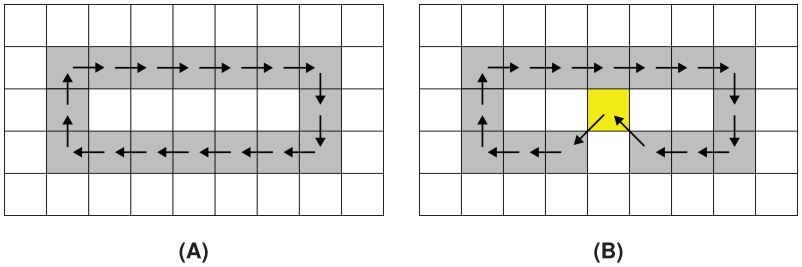

Figure 17 $\mathcal{S}$: (A) before the alteration, (B) after the alteration. The pixel that changes the topological characteristics of $\mathcal{S}$ is marked yellow. Full-size DOI: 10.7717/peerj-cs.3084/fig-17

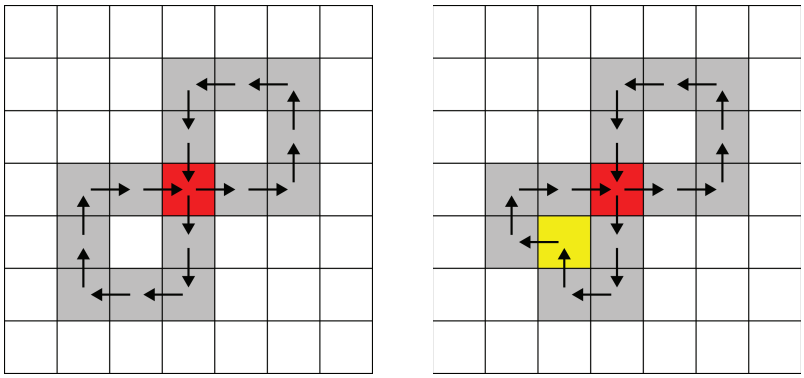

Figure 18 $\mathcal{S}$: **(A) before the alteration, (B) after the alteration.** The pixel that changes the topological characteristics of $\mathcal{S}$ is marked yellow while the self-intersection is marked red.
Full-size DOI: 10.7717/peerj-cs.3084/fig-18

belonging to the rasterised chain code sequences are stored in a hash table using the following hash function h:

$$h(x, y) = x_{max} \cdot y + x, \tag{13}$$

where x and y are rasterised pixel coordinates, and x_{max} is the maximum x coordinate. After an alteration is selected, the procedure checks whether any pixel in the altered chain code sequence borders any other pixel in segments of all chain code sequences apart from its neighbouring pixels. In such case, the alteration is rejected. Otherwise, the alteration is accepted and injected into C_i, while the hash table is updated accordingly.

Time and space complexity

Let L_0 represent the total number of chain code symbols of an input geometric shape $\mathscr{S}$, which is calculated using:

$$L_0 = \sum_{i=1}^{|\mathscr{C}|} |C_i|. \tag{14}$$

In each iteration of noise injection, the total number of chain code symbols that describe $\mathscr{S}$ can increase/decrease depending on the number of performed alterations. Each *REMOVE* operation decreases a chain code sequence length by 2, each *CUT* decreases it by 1, and each *WRAP* increases it by 1 (F8) or 2 chain code symbols (F4). Let $|REMOVE|_i$, $|CUT|_i$, and $|WRAP|_i$ represent the numbers of performed alterations *REMOVE*, *CUT*, and *WRAP* in the i-th noise injection iteration. For F8 chain codes, let k_i be defined as:

$$k_i = \frac{(|WRAP|_i - |CUT|_i - 2 \cdot |REMOVE|_i)}{L_i}. \tag{15}$$

where L_i represents the total number of chain code symbols after i noise injection iterations. Similarly, for F4 chain codes, let k_i be defined as:

$$k_i = \frac{(2 \cdot |WRAP|_i - 2 \cdot |REMOVE|_i)}{L_i}. \tag{16}$$

The total number of chain code symbols after the first iteration in the case of F8 chain codes is calculated as:

$$\begin{aligned} L_1 &= L_0 + (|WRAP|_0 - |CUT|_0 - 2 \cdot |REMOVE|_0) = \\ &= L_0 + k_0 L_0 = \\ &= L_0(1 + k_0). \end{aligned}$$

After the second noise injection iteration, the total number of chain code symbols is calculated as:

$$\begin{aligned} L_2 &= L_1 + (|WRAP|_1 - |CUT|_1 - 2 \cdot |REMOVE|_1) = \\ &= L_1 + k_1 L_1 = \\ &= L_0 + k_0 L_0 + k_1(L_0 + k_0 L_0) = \\ &= L_0(1 + k_0 + k_1 + k_0 k_1) = \\ &= L_0(1 + k_0)(1 + k_1). \end{aligned}$$

Continuing in this manner, the total length of chain codes can be expressed as:

$$L_i = L_0 \prod_{j=0}^{i-1} (1 + k_j). \tag{17}$$

For $k \ll 1$, this equation can be approximated by:

$$L_i = L_0(1 + \overline{k_i})^i, \tag{18}$$

where $(1 + \overline{k_i})$ is the geometric mean of terms in the product.

The time complexity of the proposed method depends on the total number of chain code symbols of the input geometric shape S. During the iteration through L_0 chain code symbols, each selection of a suitable alteration, detection of potential topological characteristic changes as well as local alteration of a chain code sequence are performed in $O(1)$. According to this and the previous derivation of L_i, the overall time complexity of the proposed method is $T(n) = L_0 \sum_{i=0}^{n-1} (1 + \overline{k_i})^i$. The space complexity follows the same pattern except that it includes the implicit rasterisation (for preservation of topological characteristics). Therefore, it is expressed as $S(n) = 2 \cdot L_0(1 + \overline{k_n})^n$.

RESULTS AND DISCUSSION

The proposed method was tested on a set of different geometric shapes, which can be accessed at *GeMMA (2025)* and represent a selection of shapes with different characteristics (number of chain code symbols, number of loops, *etc.*). Among those, Line Segment, Square, Twist, Ankh, Camel, and Butterfly were chosen for the demonstration purposes. Each shape is described with both F8 and F4 chain code sequences. Test shapes along with their basic properties (number of chains, total number of chain codes in F8 and F4 encodings) are collected in Table 1 while the shapes are displayed in Fig. 19.

Table 1 Properties of test geometric shapes.

Shape	$\lvert\mathscr{C}\rvert$	Number of chain code symbols (F8)	Number of chain code symbols (F4)
Line Segment	1	200	200
Square	1	400	400
Twist	1	1,600	3,200
Ankh	2	1,418	1,734
Camel	10	13,858	20,156
Butterfly	127	46,172	64,374

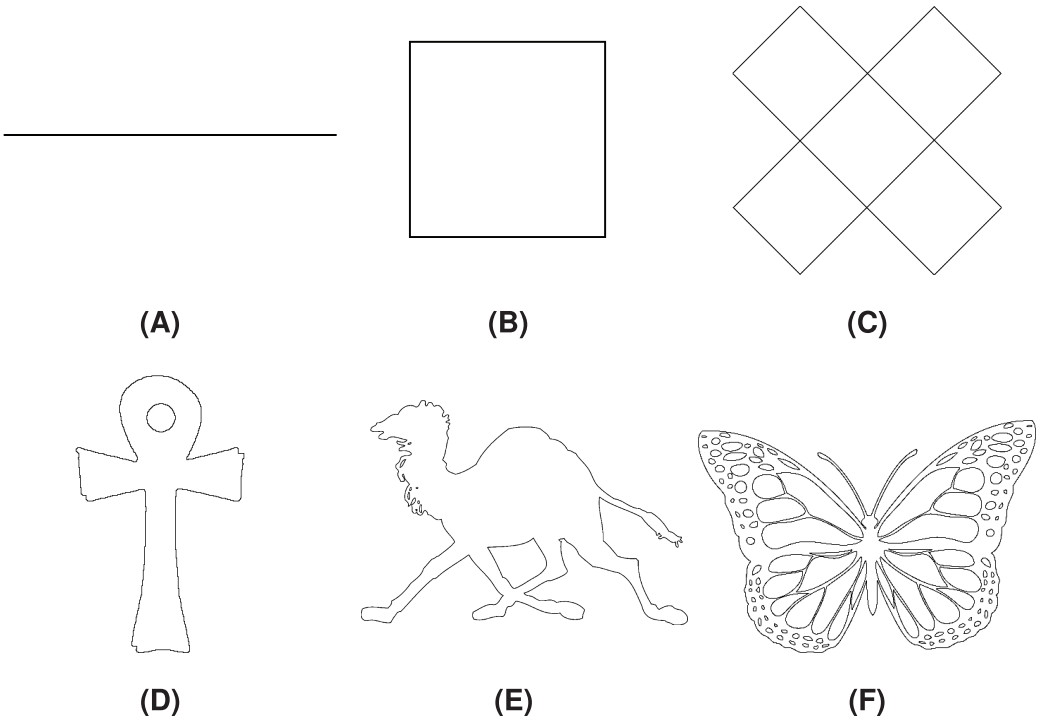

Figure 19 Test geometric shapes: (A) Line Segment, (B) Square, (C) Twist, (D) Ankh, (E) Camel, (F) Butterfly. Full-size DOI: 10.7717/peerj-cs.3084/fig-19

Fractal Dimension (*FD*) (*Mandelbrot, 1967*) is an index that indicates the ruggedness of a pattern and its detail complexity (*Allen, Brown & Miles, 1995*). It is given in Eq. (19):

$$FD = \frac{\ln N(\varepsilon)}{\ln \frac{1}{\varepsilon}} \tag{19}$$

where N denotes the number of pixels that lie on the shape's boundary, while ε represents the ratio between the pixel side and the size of geometrical space.

Mean Distance from Original (*MDfO*) measures the average distance between the pixels of the original and the perturbed geometric shape as (*Rogelj, Hudej & Petric, 2013*):

$$MDfO = \frac{1}{|\mathscr{C}|}\sum_{i=1}^{|\mathscr{C}|}\frac{1}{|C_i|}\sum_{j=1}^{|C_i|}\min_{c^0_{i,k}\in C^0_i}\left\|c_{i,j} - c^0_{i,k}\right\| \tag{20}$$

where C^0_i represents an original chain code sequence without injected noise.

Table 2 ***FD*** **and** ***MDfO*** **values after 100 iterations of noise injection into F8 test shapes using different values of** μ**.**

Shape	Metric	Noise injection probability				
		$\mu = 0.01$	$\mu = 0.05$	$\mu = 0.10$	$\mu = 0.20$	$\mu = 0.50$
Line Segment	*FD*	1.00	1.01	1.03	1.04	1.11
	MDfO	0.39	0.84	1.01	1.16	2.25
Square	*FD*	1.29	1.29	1.31	1.33	1.40
	MDfO	0.47	0.72	0.96	1.43	2.25
Ankh	*FD*	1.21	1.23	1.24	1.27	1.32
	MDfO	0.45	0.80	1.96	1.38	2.08
Twist	*FD*	1.28	1.31	1.33	1.34	1.40
	MDfO	0.49	0.88	1.12	1.35	2.03
Camel	*FD*	1.23	1.25	1.26	1.28	1.33
	MDfO	0.39	0.75	0.99	1.31	2.04
Butterfly	*FD*	1.39	1.40	1.41	1.43	1.48
	MDfO	0.44	0.79	1.01	1.34	2.08

Table 3 ***FD*** **and** ***MDfO*** **values after 100 iterations of noise injection into F4 test shapes using different values of** μ**.**

Shape	Metric	Noise injection probability				
		$\mu = 0.01$	$\mu = 0.05$	$\mu = 0.10$	$\mu = 0.20$	$\mu = 0.50$
Line Segment	*FD*	1.12	1.27	1.36	1.47	1.60
	MDfO	0.96	2.70	3.85	6.89	16.28
Square	*FD*	1.40	1.55	1.65	1.73	1.81
	MDfO	0.77	2.31	3.70	6.99	14.66
Ankh	*FD*	1.30	1.43	1.53	1.63	1.72
	MDfO	0.58	2.05	3.62	6.09	11.68
Twist	*FD*	1.38	1.48	1.53	1.67	1.80
	MDfO	0.59	1.91	3.32	5.98	13.51
Camel	*FD*	1.30	1.39	1.47	1.55	1.66
	MDfO	0.57	2.00	3.33	5.40	11.29
Butterfly	*FD*	1.45	1.54	1.62	1.69	1.78
	MDfO	0.57	1.98	3.37	5.91	11.00

Noise injection into test geometric shapes was evaluated with *FD* and *MDfO*. Tables 2 and 3 present the values of *FD* and *MDfO* after 100 noise injection iterations with different alteration probabilities μ. As expected, both *FD* and *MDfO* increase with higher values of μ. It can be observed that F4 test shapes are affected by noise injection more than their F8 counterparts. This difference can be attributed to the definition of the common alteration *WRAP* in F4, which increases the number of chain code symbols by 2, whereas the maximum increment of chain code symbols in a single alteration in F8 is only 1.

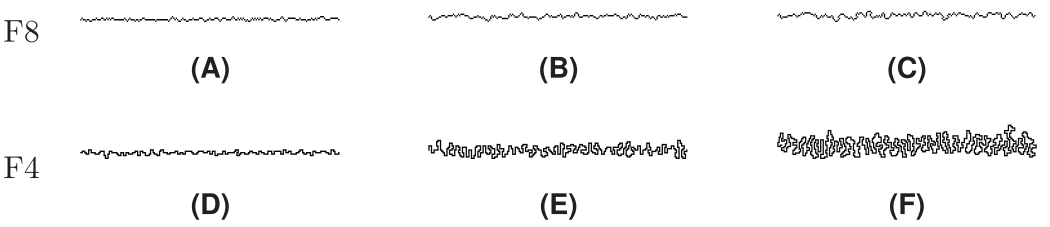

Figure 20 Line Segment after different numbers of noise injection iterations: (A, D) n = 10, (B, E) n = 50, (C, F) n = 100. Full-size DOI: 10.7717/peerj-cs.3084/fig-20

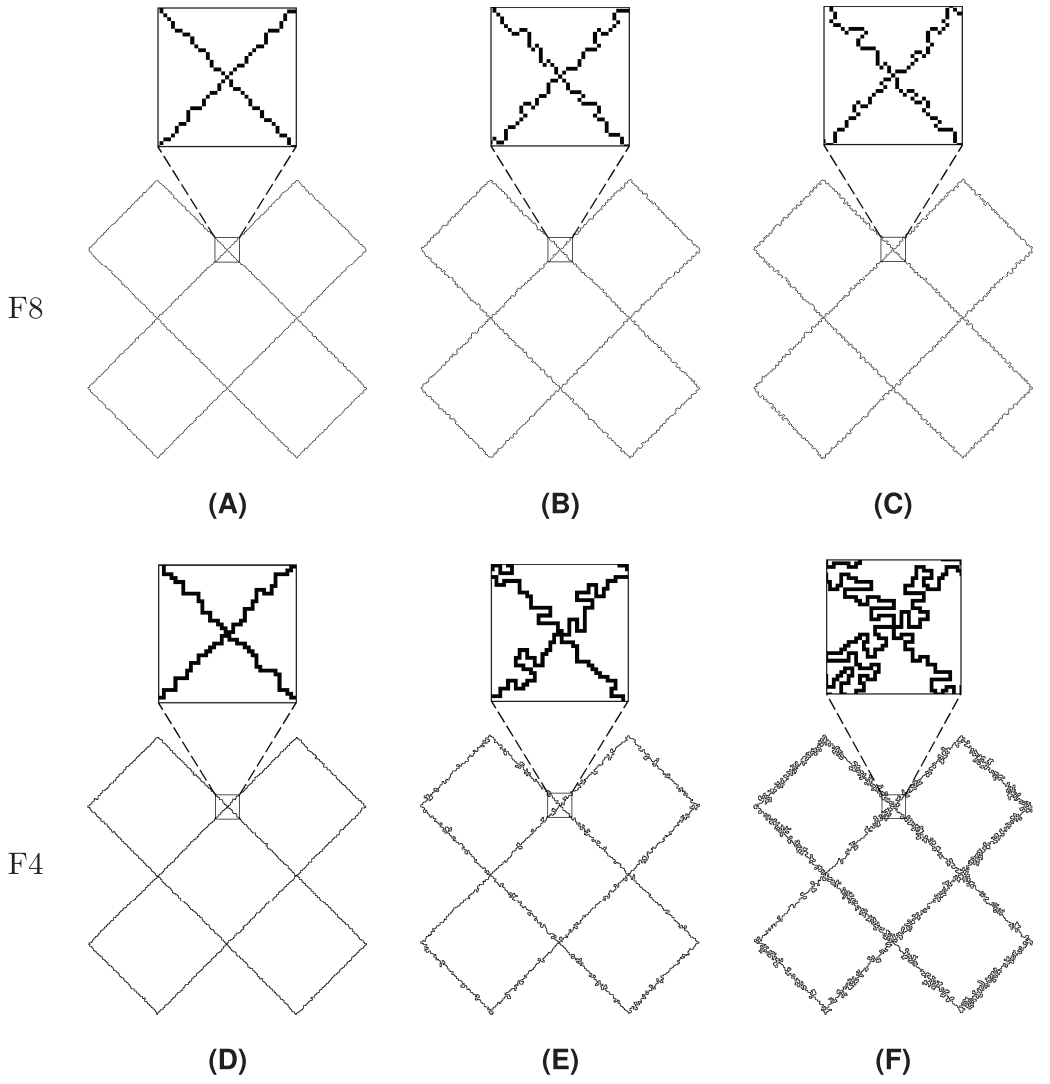

Figure 21 Twist after different numbers of noise injection iterations: (A, D) n = 10, (B, E) n = 50, (C, F) n = 100. Full-size DOI: 10.7717/peerj-cs.3084/fig-21

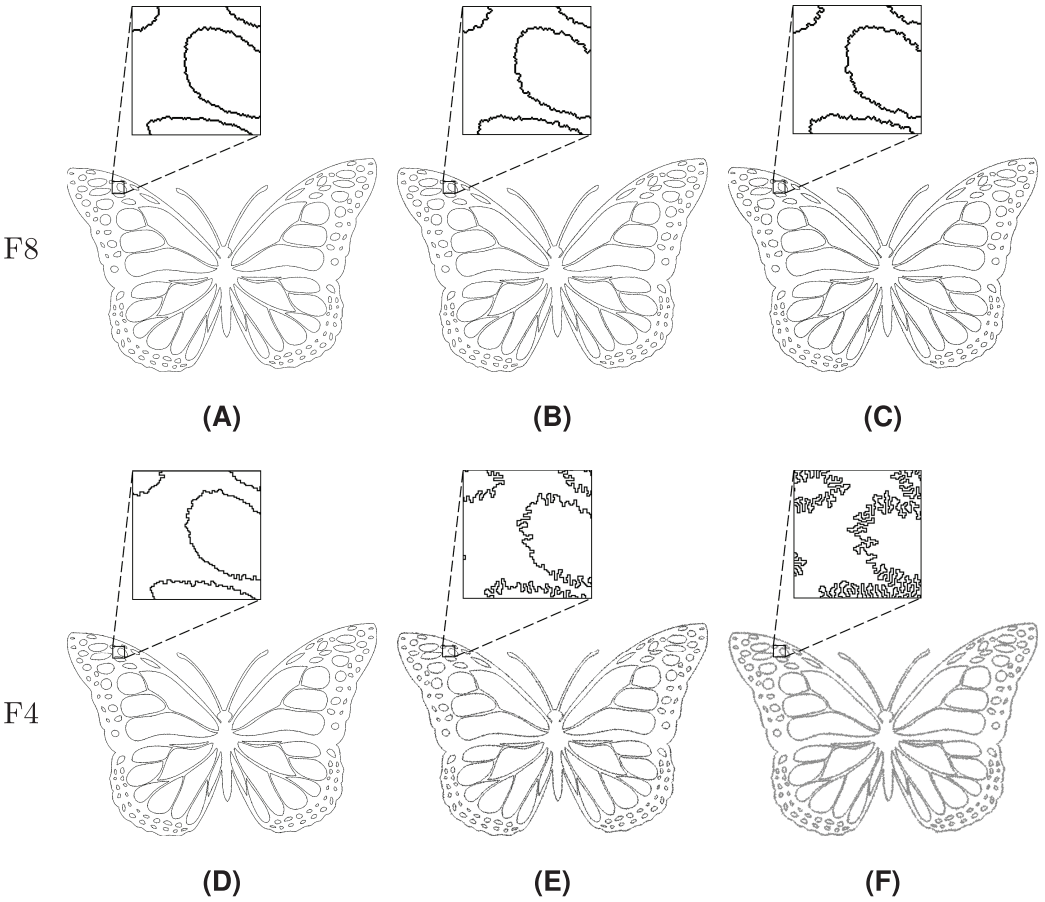

Figure 22 **Butterfly after different numbers of noise injection iterations: (A, D) $n = 10$, (B, E) $n = 50$, (C, F) $n = 100$.** Full-size DOI: 10.7717/peerj-cs.3084/fig-22

The value $\mu = 0.10$ was used in the rest of experiments in order to observe gradual effects of noise injection over an increasing number of iterations. Injection of noise into both F8 and F4 shapes is visually demonstrated in Figs. 20–22, where subfigures (A), (B), and (C) indicate F8 shapes, while (D), (E), and (F) indicate F4 shapes.

Noise injection into Line Segment is shown in Fig. 20. Among the test shapes, it is the only open shape. Therefore, it does not contain any loops or rings as it is described by a path instead. It should be noted that Line Segment remains open no matter the number of noise injection iterations.

Figure 21 represents Twist, a shape that is described by a single loop. Topologically, Twist is different from other test shapes since it contains four self-intersections. It can be seen that the injected noise does not change the shape's topological characteristics.

The last presented shape is Butterfly (Fig. 22), which consists of a loop and 126 rings. The noise injection method successfully handled the shape despite the fact that it is considerably complicated.

In Tables 4 and 5, *FD* and *MDfO* evaluation values are collected at different numbers of noise injection iterations for F8 and F4 geometric shapes, respectively. *FD* values generally

Table 4 ***FD*** **and** ***MDfO*** **values according to increasing numbers of noise injection iterations into F8 test shapes.**

Shape	Metric	Number of iterations					
		$n = 1$	$n = 10$	$n = 50$	$n = 100$	$n = 500$	$n = 1{,}000$
Line Segment	*FD*	1.00	1.00	1.01	1.03	1.17	1.30
	MDfO	0.07	0.46	0.71	1.01	2.54	4.35
Square	*FD*	1.30	1.29	1.29	1.31	1.44	1.56
	MDfO	0.07	0.40	0.67	0.96	2.80	4.62
Ankh	*FD*	1.21	1.21	1.23	1.24	1.37	1.47
	MDfO	0.08	0.49	0.75	0.96	2.51	4.13
Twist	*FD*	1.24	1.28	1.31	1.33	1.42	1.52
	MDfO	0.12	0.50	0.87	1.12	2.52	4.36
Camel	*FD*	1.23	1.23	1.25	1.26	1.35	1.44
	MDfO	0.09	0.46	0.74	0.99	2.50	4.15
Butterfly	*FD*	1.38	1.39	1.40	1.41	1.50	1.58
	MDfO	0.09	0.43	0.78	1.01	2.48	4.17

Table 5 ***FD*** **and** ***MDfO*** **values according to increasing numbers of noise injection iterations into F4 test shapes.**

Shape	Metric	Number of iterations					
		$n = 1$	$n = 10$	$n = 50$	$n = 100$	$n = 500$	$n = 1{,}000$
Line Segment	*FD*	1.03	1.11	1.25	1.36	1.62	1.72
	MDfO	0.23	0.85	2.40	3.85	18.43	38.34
Square	*FD*	1.32	1.40	1.55	1.65	1.80	1.83
	MDfO	0.21	0.84	2.32	3.70	16.61	34.29
Ankh	*FD*	1.26	1.30	1.43	1.53	1.73	1.77
	MDfO	0.12	0.58	1.99	3.62	12.62	19.85
Twist	*FD*	1.35	1.35	1.42	1.53	1.79	1.84
	MDfO	0.12	0.57	1.89	3.32	15.69	31.69
Camel	*FD*	1.28	1.30	1.39	1.47	1.68	1.75
	MDfO	0.09	0.57	1.86	3.33	12.68	23.59
Butterfly	*FD*	1.43	1.45	1.54	1.62	1.78	1.82
	MDfO	0.10	0.55	1.91	3.37	11.74	13.92

increase as more noise is applied to shapes (Fig. 23). Due to its limited function range, *FD* tends to converge towards the upper limit of the range (indicating a shape that is essentially dominated by noise). Hence, *FD* can also serve as a criterion for determining a stop condition for noise injection (*e.g.*, when after several iterations *FD* rises only marginally). On the other hand, *MDfO* exhibits a more continuous growth across iterations (Fig. 24). Unlike *FD*, the initial value of *MDfO* is zero for all geometric shapes, making it particularly suitable for estimating the amount of injected noise. This property also enables comparisons of noise levels embedded in different geometric shapes. Furthermore, it can

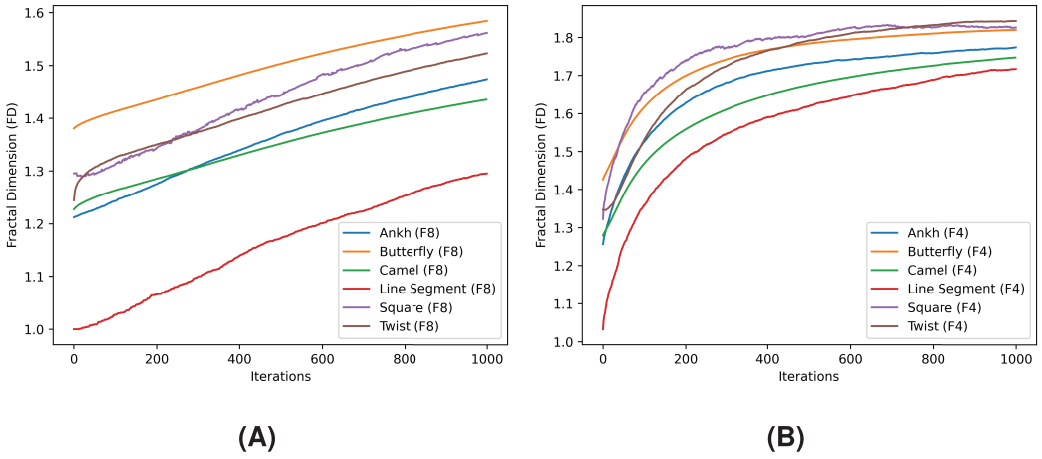

Figure 23 **Fractal dimension (FD) values for the test shapes according to different numbers of noise injection iterations: (A) F8, (B) F4.** Full-size DOI: 10.7717/peerj-cs.3084/fig-23

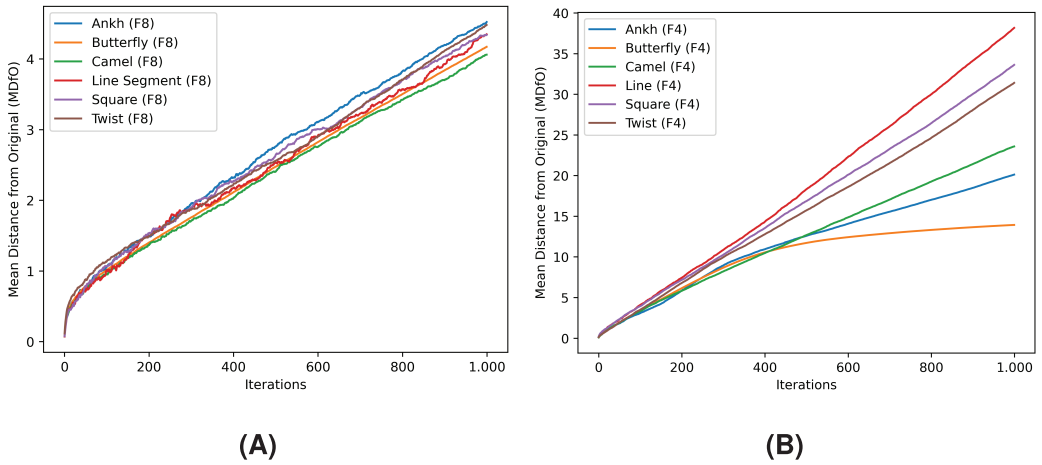

Figure 24 **Mean distance from original (MDfO) values for the test shapes according to different numbers of noise injection iterations: (A) F8, (B) F4.** Full-size DOI: 10.7717/peerj-cs.3084/fig-24

be seen that the growth of *MDfO* slows down beyond a certain number of noise injection iterations, particularly for geometric shapes with many rings. The reason for this is that the raster space in the interior of a geometric shape gets occupied and the geometric shape can be expanded only outwards, causing many alterations to be rejected.

CONCLUSIONS

This article introduces a method for injection of noise into shape boundaries described by Freeman chain codes. Shapes retain their property of being closed or open after noise is injected into their chain code sequences due to the design of the proposed method. Furthermore, during the noise injection, the procedure prevents alterations that would cause changes in shape's topological characteristics. Different test shapes were used to demonstrate noise injection into chain codes. Although the difference of their properties is

immense, the method managed to process all shapes successfully regardless of their complexity. The evaluation metrics fractal dimension and mean distance from original were used for estimating the degree of injected noise in a shape. The proposed method provides an efficient process of noise injection performed at the chain code level. To the best of our knowledge, it is the first method that enables direct noise injection into chain codes without the need to rasterise and re-encode the geometric shapes.

Noise injection into chain codes could serve as a preprocessing phase of different methods by producing noisy datasets. Especially in the field of deep neural networks, predominantly dealing with chain codes (*Sanchiz, Iñesta & Pla, 1996*; *Subri, Haron & Sallehuddin, 2006*), adding noise to the training data could be a crucial step in order to reduce the influence of phenomena that occur during the training phase of a neural network. Therefore, data augmentation in the form of providing noisy datasets would be of the utmost importance for enhancing the performance of neural networks.

There are several directions for future work. The most obvious improvement would be to adapt the method to process other types of chain codes, such as VCC, 3OT, UMCC, chain codes based on biological computing, or multi-resolution chain codes. Apart from the encoding type, an adjustment could be made to the method in order to enable the noise injection into 3D chain codes (*Sánchez-Cruz, López-Valdez & Cuevas, 2014*; *Strnad et al., 2019*). Furthermore, the method could be applied to machine learning (ML), where the performance of ML models could be evaluated under varying degrees of noise injected into their datasets.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This research was funded by the Slovenian Research and Innovation Agency under Research Project J2-4458 and Research Programme P2-0041, and the Czech Science Foundation under Research Project 23-04622L. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Slovenian Research and Innovation Agency: J2-4458.
Research Programme: P2-0041.
Czech Science Foundation: 23-04622L.

Competing Interests

The authors have a patent with the number EP24150479.4 pending to University of Maribor.

Author Contributions

- Luka Lukač conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, and approved the final draft.

- Andrej Nerat conceived and designed the experiments, performed the experiments, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Damjan Strnad analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Ivana Kolingerová analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Borut Žalik conceived and designed the experiments, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.

Patent Disclosures

The following patent dependencies were disclosed by the authors:

Patent application number: EP24150479.4

Name: A method for noise injection into 2D geometric shapes described by Freeman chain codes

Status: pending

Organization: University of Maribor

Authors: Luka Lukač, Damjan Strnad, Andrej Nerat, Borut Žalik

Date of filing: 5. 1. 2024

Data Availability

The following information was supplied regarding data availability:

The dataset that contains the testing examples is available at: https://gemma.feri.um.si/chaincodes/ (accessed on 13 February 2025).

The source code of the method is available at Github and Zenodo:

- https://github.com/luckyLukac/FreemanChainCodeNoiseInjection.

- luckyLukac. (2025). luckyLukac/FreemanChainCodeNoiseInjection: Version 1.00 (release). Zenodo. https://doi.org/10.5281/zenodo.15877994.

REFERENCES

Aldemir E, Arturo Tapia Dueñas O, Emre Kavur A, Tohumoglu G, Sánchez-Cruz H, Alper Selver M. 2023. Chain code strategy for lossless storage and transfer of segmented binary medical data. *Expert Systems with Applications* **216(4)**:119449 DOI 10.1016/j.eswa.2022.119449.

Ali MN, El-Dahshan E-SA, Yahia AH. 2017. Denoising of heart sound signals using discrete wavelet transform. *Circuits, Systems, and Signal Processing* **36(7)**:4482–4497 DOI 10.1007/s00034-025-03000-4.

Allen M, Brown GJ, Miles NJ. 1995. Measurement of boundary fractal dimensions: review of current techniques. *Powder Technology* **84(1)**:1–14 DOI 10.1016/0032-5910(94)02967-s.

Althobaiti H, Lu C. 2017. A survey on Arabic optical character recognition and an isolated handwritten Arabic character recognition algorithm using encoded Freeman chain code. In: *Proceedings of the 2017 51st Annual Conference on Information Sciences and Systems (CISS)*. Piscataway: IEEE, 1–6.

Boukharouba A, Bennia A. 2017. Novel feature extraction technique for the recognition of handwritten digits. *Applied Computing and Informatics* **13(1)**:19–26 DOI 10.1016/j.aci.2015.05.001.

Bribiesca E. 1999. A new chain code. *Pattern Recognition* **32(2)**:235–251 DOI 10.1016/s0031-3203(98)00132-0.

Chen Z, Duan F, Chapeau-Blondeau F, Abbott D. 2022. Training threshold neural networks by extreme learning machine and adaptive stochastic resonance. *Physics Letters A* **432(21)**:128008 DOI 10.1016/j.physleta.2022.128008.

Dhou K. 2020. A new chain coding mechanism for compression stimulated by a virtual environment of a predator–prey ecosystem. *Future Generation Computer Systems* **102(1)**:650–669 DOI 10.1016/j.future.2019.08.021.

Dhou K, Cruzen C. 2019. An innovative chain coding technique for compression based on the concept of biological reproduction: an agent-based modeling approach. *IEEE Internet of Things Journal* **6(6)**:9308–9315 DOI 10.1109/jiot.2019.2912984.

Dhou K, Cruzen C. 2021. A highly efficient chain code for compression using an agent-based modeling simulation of territories in biological beavers. *Future Generation Computer Systems* **118(4)**:1–13 DOI 10.1016/j.future.2020.12.016.

Duan L, Ren Y, Duan F. 2022. Adaptive stochastic resonance based convolutional neural network for image classification. *Chaos, Solitons & Fractals* **162(2)**:112429 DOI 10.1016/j.chaos.2022.112429.

Fanti Z, Braumann U-D, Rauscher FG, Ebert T, Bribiesca E, Martinez-Perez ME. 2025. Slope chain code-based scale-independent tortuosity measurement on retinal vessels. *Experimental Eye Research* **254(23)**:110286 DOI 10.1016/j.exer.2025.110286.

Freeman H. 1961. On the encoding of arbitrary geometric configurations. *IEEE Transactions on Electronic Computers* **EC-10(2)**:260–268 DOI 10.1109/tec.1961.5219197.

GeMMA. 2025. Chaincodes—GeMMA lab. *Available at https://gemma.feri.um.si/chaincodes/* (accessed 17 January 2025).

Honi DG, Ali AH, Abduljabbar ZA, Ma J, Nyangaresi VO, Mutlaq KA-A, Umran SM. 2022. Towards fast edge detection approach for industrial products. In: *Proceedings of the 2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*. Piscataway: IEEE, 239–244.

Jeromel A, Žalik B. 2020. An efficient lossy cartoon image compression method. *Multimedia Tools and Applications* **79(1–2)**:433–451 DOI 10.1007/s11042-019-08126-7.

Jothimani S, Premalatha K. 2022. MFF-SAug: multi feature fusion with spectrogram augmentation of speech emotion recognition using convolution neural network. *Chaos, Solitons & Fractals* **162(1)**:112512 DOI 10.1016/j.chaos.2022.112512.

Kabir S. 2015. A compressed representation of mid-crack code with Huffman code. *International Journal of Image, Graphics and Signal Processing* **7(10)**:11–18 DOI 10.5815/ijigsp.2015.10.02.

Mandelbrot B. 1967. How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science* **156(3775)**:636–638 DOI 10.1126/science.156.3775.636.

Mortenson ME. 1997. *Geometric modeling*. New York: John Wiley & Sons, Inc.

Nerat A, Strnad D, Žalik KR, Žalik B. 2024. An efficient multi-resolution chain coding. *IEEE Access* **12**:54721–54731 DOI 10.1109/access.2024.3389062.

Rogelj P, Hudej R, Petric P. 2013. Distance deviation measure of contouring variability. *Radiology and Oncology* **47(1)**:86–96 DOI 10.2478/raon-2013-0005.

Sánchez-Cruz H, Bribiesca E, Rodríguez-Dagnino RM. 2007. Efficiency of chain codes to represent binary objects. *Pattern Recognition* **40(6)**:1660–1674 DOI 10.1016/j.patcog.2006.10.013.

Sánchez-Cruz H, López-Valdez HH, Cuevas FJ. 2014. A new relative chain code in 3D. *Pattern Recognition* **47(2)**:769–788 DOI 10.1016/j.patcog.2013.08.010.

Sánchez-Cruz H, Rodríguez-Dagnino RM. 2005. Compressing bi-level images by means of a 3-bit chain code. *Optical Engineering* **44(9)**:1–8 DOI 10.1117/1.2052793.

Sanchiz JM, Iñesta J, Pla F. 1996. A neural network-based algorithm to detect dominant points from the chain-code of a contour. In: *Proceedings of the 13th International Conference on Pattern Recognition*. Vol. 4. Piscataway: IEEE, 325–329.

Sharma A, Grau O, Fritz M. 2016. VConv-DAE: deep volumetric shape learning without object labels. In: Hua G, Jégou H, eds. *Computer Vision—ECCV, 2016 Workshops*. Amsterdam, The Netherlands: Springer International Publishing, 236–250.

Strnad D, Kohek Š, Nerat A, Žalik B. 2019. Efficient representation of geometric tree models with level-of-detail using compressed 3D chain code. *IEEE Transactions on Visualization and Computer Graphics* **26(11)**:3177–3188 DOI 10.1109/tvcg.2019.2924430.

Strnad D, Žlaus D, Nerat A, Žalik B. 2024. Efficient compressed storage and fast reconstruction of large binary images using chain codes. *Multimedia Tools and Applications* **13(4)**:1–19 DOI 10.1007/s11042-024-20199-7.

Subri SH, Haron H, Sallehuddin R. 2006. Neural network corner detection of vertex chain code. *AIML Journal* **6(1)**:37–43.

Tapia-Dueñas OA, López HH, Sánchez-Cruz H. 2025. A polyhedral reconstruction of a 3D object from a chain code and a low-density point cloud. *Multimedia Tools and Applications* **21(17)**:1–28 DOI 10.1007/s11042-025-20770-w.

Wang C, Principe JC. 1999. Training neural networks with additive noise in the desired signal. *IEEE Transactions on Neural Networks* **10(6)**:1511–1517 DOI 10.1109/72.809097.

Wang Y, Wang J. 2022. Fingertip detection based on Freeman chain code analysis. In: *Proceedings of the 2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. Wuhan, China, 651–654.

Wilson G. 1997. Properties of contour codes. *IEE Proceedings—Vision, Image, and Signal Processing* **144(3)**:145–149 DOI 10.1049/ip-vis:19971159.

Xu J, Wang Z, Tan C, Si L, Zhang L, Liu X. 2016. Adaptive wavelet threshold denoising method for machinery sound based on improved fruit fly optimization algorithm. *Applied Sciences* **6(7)**:199 DOI 10.3390/app6070199.

Yin S, Liu C, Zhang Z, Lin Y, Wang D, Tejedor J, Zheng TF, Li Y. 2015. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing* **2015(1)**:1–14 DOI 10.1186/s13636-014-0047-0.

Zahir S, Dhou K, George BP. 2007. A new chain coding based method for binary image compression and reconstruction. In: *Proceedings of the 26th Picture Coding Symposium (PCS2007)*. Lisbon, Portugal, 1321–1324.

Žalik KR. 2024. A multiresolution chain coding scheme for different resolution ratios. In: *Proceedings of the 2024 9th International Conference on Mathematics and Computers in Sciences and Industry (MCSI)*, Rhodes Island, Greece, 115–120.

Žalik B, Mongus D, Liu Y-K, Lukač N. 2016. Unsigned Manhattan chain code. *Journal of Visual Communication and Image Representation* **38(3)**:186–194 DOI 10.1016/j.jvcir.2016.03.001.

Žalik B, Mongus D, Lukač N, Žalik KR. 2018. Efficient chain code compression with interpolative coding. *Information Sciences* **439(8)**:39–49 DOI 10.1016/j.ins.2018.01.045.

Žalik B, Mongus D, Žalik KR, Podgorelec D, Lukač N. 2021. Lossless chain code compression with an improved binary adaptive sequential coding of zero-runs. *Journal of Visual Communication and Image Representation* **75(12)**:103050 DOI 10.1016/j.jvcir.2021.103050.

Zhang X, Yang F, Guo Y, Yu H, Wang Z, Zhang Q. 2023. Adaptive differential privacy mechanism based on entropy theory for preserving deep neural networks. *Mathematics* **11(2)**:330 DOI 10.3390/math11020330.

Zhao H-H, Liu H. 2020. Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. *Granular Computing* **5(3)**:411–418 DOI 10.1007/s41066-019-00158-6.