applied sciences

Article

Procedural Generation of Artistic Patterns Using a Modified Orbit Trap Method

Krzysztof Gdawiec [1,*] and Hezekiah Adewinbi [2]

[1] Institute of Computer Science, University of Silesia, Bedzinska 39, 41-200 Sosnowiec, Poland
[2] Department of Mathematical Sciences, Kent State University, Summit Street, Kent, OH 44242, USA; hezekiah@aims.edu.gh
* Correspondence: krzysztof.gdawiec@us.edu.pl

Abstract: In the literature, we can find various methods for generating artistic patterns. One of the methods is the orbit trap method. In this paper, we propose various modifications of a variant of the orbit trap method that generates patterns with wallpaper symmetry. The first modification relies on replacing the Picard iteration (used in the original method) with the S-iteration known from the fixed point theory. Moreover, we extend the parameters in the S-iteration from scalar to vector ones. In the second modification, we replace the Euclidean metric used in the orbit traps with other metrics. Finally, we propose three new orbit traps. The presented examples show that using the proposed method, we are able to obtain a great variety of interesting patterns. Moreover, we show that a proper selection of the orbit traps and the mapping used by the method can lead to patterns that possess a local fractal structure.

Keywords: orbit trap; S-iteration; artistic pattern; wallpaper symmetry

Citation: Gdawiec, K.; Adewinbi, H. Procedural Generation of Artistic Patterns Using a Modified Orbit Trap Method. *Appl. Sci.* **2022**, *12*, 2923. https://doi.org/10.3390/app12062923

Academic Editor: Antonio Fernández-Caballero

Received: 18 February 2022
Accepted: 10 March 2022
Published: 13 March 2022

1. Introduction

A pattern is a generic term for any type of repeated, often regular, arrangement [1]. It is also a design in which lines, shapes, forms, or colours are repeated. The repeated part is called a motif. A motif can be repeated and arranged in many ways to create different types of patterns. In regular patterns, the motif (or motifs) is repeated predictably [2]. It could be the same each time, or it could change in regularly repeating ways. Motifs can be arranged in many ways to create a regular pattern. An irregular pattern is one in which the motif changes or the way it is repeated is unpredictable [3]. How complicated a pattern is depends on what is repeated and the way in which it is repeated.

Before the invention of computers, patterns were generated by hand through drawings and many other skills. It takes a very long time to generate a single pattern, and sometimes, after all the labour, the pattern may not meet the requirements of the designer or the consumers. With the use of computers, it is easy to generate millions of different patterns in just a minute [4]. In a bid to satisfy human wants, researchers have found out that patterns can be generated using mathematical equations or artificial intelligence methods.

In the literature, we can find various methods for generating patterns that use mathematical equations. Among the methods, a popular approach is the use of fractals, e.g., inversion fractals [5] and Mandelbox [6]. Furthermore, other non-Euclidean geometries are used in the generation of patterns. For instance, in [7], the authors used spherical geometry, whereas in [8], hyperbolic geometry was used. In the generation of patterns, not only various types of non-Euclidean geometries are used. Mathematical objects such as whirls [9] or spirals [10,11] are commonly used in obtaining artistic patterns. Another group of methods are the methods that are based on the dynamics of discrete dynamical systems [12–14]. Not only the dynamics but also the orbits of discrete dynamical systems can generate aesthetic patterns [15]. Based on the results presented in [15] and the orbit trap method [16], Lu et al. in [17] introduced a method that is the main subject of this paper, namely, the orbit trap method that generates patterns with wallpaper symmetry.

In this paper, we introduce modifications of the orbit trap method presented by Lu et al. in [17]. The first modification relies on using the S-iteration [18] instead of the Picard one that is used in the original algorithm. We also extend the parameters in the S-iteration from scalar to vector ones. Moreover, we propose the use of various metrics for defining orbit traps and introduce some new orbit traps. Additionally, we present examples showing that we can obtain a local fractal structure in the generated pattern when using an appropriate mapping and orbit traps.

The paper is organised as follows. In Section 2, we briefly present the orbit trap method. Then, in Section 3, we introduce modifications to the orbit trap method from Section 2. Some examples showing the potential of the proposed modifications are presented in Section 4. Moreover, we present examples showing the local fractal structure of some of the patterns. Finally, in Section 5, we provide some concluding remarks and the directions for future work.

2. Orbit Trap Method

In [17], Lu et al. introduced a method for generating artistic images with wallpaper symmetries. The method was based on the mappings for which the chaotic attractor possesses a wallpaper symmetry [15] and the orbit trap method [16,19]. Moreover, they proposed a method for generating colour maps using an exponential function. In this section, we briefly present the method of Lu et al.

In general, in the method for each point in the considered area, we iterate a function starting from the point and check whether the transformed point falls into one of the orbit traps (orbit trap is a bounded area in the plane). When the point falls into one of the orbit traps, the point is considered to be trapped, and we end the iteration process. Next, a colour is assigned to the starting point based on the distance from the centre of the orbit trap to the trapped point. If the point is not trapped in the given number of iterations, then it gets a background colour.

To obtain images with wallpaper symmetry, we need to use a proper function that is iterated and some orbit traps. Lu et al. based their method on the functions introduced in [15]. Depending on the symmetry type, we need to use different functions.

For the symmetry type p2, pm, pg, cm, pmm, cmm, pmg, pgg, p4, p4m, and p4g, we use function $f : \mathbb{R}^2 \to \mathbb{R}^2$ of the following form [15]:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = [1, \cos x, \cos 2x, \sin x, \sin 2x] \begin{bmatrix} P_{00} & \dots & P_{04} \\ \vdots & \vdots & \vdots \\ P_{40} & \dots & P_{44} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \cos y \\ \cos 2y \\ \sin y \\ \sin 2y \end{bmatrix} \mod \begin{bmatrix} 2\pi \\ 2\pi \end{bmatrix}, \tag{1}$$

where $P_{00}, P_{01}, \dots, P_{44} \in \mathbb{R}^2$. In [15], Carter et al. derived conditions that the points $P_{00}, P_{01}, \dots, P_{44}$ must satisfy to obtain each of the symmetry type. For completeness, we gather the conditions in Appendix A.

To define functions for the p3, p3m1, p31m, p6, and p6m symmetries, let us firstly define the vectors [15]:

$$v_0 = \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{3}} \end{bmatrix}, \; v_1 = \begin{bmatrix} 0 \\ -\frac{2}{\sqrt{3}} \end{bmatrix}, \; v_2 = v_0 + v_1, \tag{2}$$

and the following mappings:

$$R_3\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3}$$

$$R_6\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4}$$

Let us note that R_3 is a counter-clockwise rotation by 120°, whereas R_6 is a counter-clockwise rotation by 60°.

Moreover, for $k \in \{3, 6\}$, let us define [15]

$$g_k(p, v, G_v) = \sum_{i=0}^{k-1} R_k^i(G_v)\cos(R_k^i(v) \cdot p), \tag{5}$$

$$h_k(p, v, H_v) = \sum_{i=0}^{k-1} R_k^i(H_v)\sin(R_k^i(v) \cdot p), \tag{6}$$

where $p, v, G_v, H_v \in \mathbb{R}^2$, and $\cdot$ is the dot product.

For each of the v_0, v_1, v_2 vectors, let us arbitrarily select $G_{v_0}, H_{v_0}, G_{v_1}, H_{v_1}, G_{v_2}, H_{v_2} \in \mathbb{R}^2$, respectively. Now, for each of the p3, p3m1, p31m, p6, p6m symmetries, we define the $f : \mathbb{R}^2 \to \mathbb{R}^2$ function in the following way [15]:

- p3 symmetry ($k = 3$) and p6 symmetry ($k = 6$)

$$f(p) = \sum_{j=0}^{2} \left(g_k(p, v_j, G_{v_j}) + h_k(p, v_j, H_{v_j}) \right) \mod \begin{bmatrix} 2\pi \\ 2\pi \end{bmatrix}, \tag{7}$$

- p3m1 and p31m symmetries ($k = 3$), and p6m symmetry ($k = 6$)

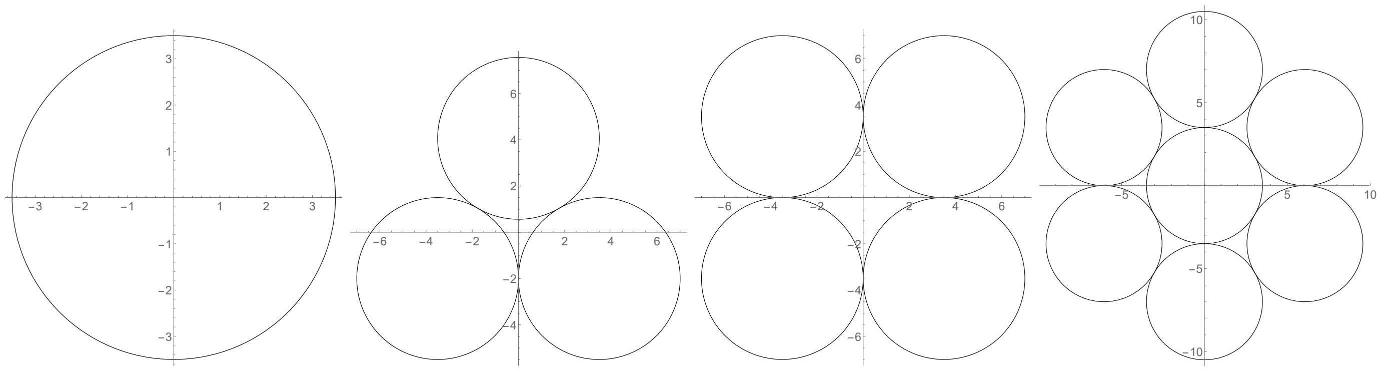
$$\begin{aligned} f(p) = \sum_{j=0}^{2} \Big(& g_k(p, v_j, G_{v_j}) + g_k(p, \sigma(v_j), \sigma(G_{v_j})) \\ & + h_k(p, v_j, H_{v_j}) + h_k(p, \sigma(v_j), \sigma(H_{v_j})) \Big) \mod \begin{bmatrix} 2\pi \\ 2\pi \end{bmatrix}, \end{aligned} \tag{8}$$

where

$$\sigma\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{cases} \begin{bmatrix} -x \\ y \end{bmatrix} & \text{for p3m1 symmetry,} \\ \begin{bmatrix} x \\ -y \end{bmatrix} & \text{for p31m, p6m symmetries.} \end{cases} \tag{9}$$

The next thing that we need in the method are the orbit traps. Lu et al. in [17] defined four orbit traps that were based on circles. We present them in Figure 1. The centres c_i and the radii r_i for the circles in each case are the following:

(a) $c_0 = [0, 0]^T, r_0 = 3.5$,
(b) $c_0 = [-3.5, -2]^T, c_1 = [3.5, -2]^T, c_2 = [0, -2 + 3.5\sqrt{3}]^T, r_0 = r_1 = r_2 = 3.5$,
(c) $c_0 = [-3.5, 3.5]^T$, $c_1 = [3.5, 3.5]^T$, $c_2 = [-3.5, -3.5]^T$, $c_3 = [3.5, -3.5]^T$, $r_0 = r_1 = r_2 = r_3 = 3.5$,
(d) $c_0 = [0, 0]^T, c_1 = [0, 7]^T, c_2 = [-3.5\sqrt{3}, 3.5]^T, c_3 = [-3.5\sqrt{3}, -3.5]^T, c_4 = [0, -7]^T$, $c_5 = [3.5\sqrt{3}, -3.5]^T, c_6 = [3.5\sqrt{3}, 3.5]^T, r_0 = r_1 = \ldots = r_6 = 3.5$.



(**a**) (**b**) (**c**) (**d**)

Figure 1. Orbit traps used in [17]. (**a**) one orbit trap; (**b**) three orbit traps; (**c**) four orbit traps; and (**d**) seven orbit traps.

The points in the orbit trap should be coloured according to some colour map. In [17], a method for generating colour maps that is based on an exponential function was proposed. In this method, we have a set of base colours $\{b_0, b_1, \ldots, b_N\}$, and for each of the base colours we assign a colour ratio $q_i \in [0,1]$ for $i = 0, 1, \ldots, N$ such that $q_0 < q_1 < \ldots < q_N$. Moreover, we give the number of colours N_c in the colour map. The method for generating the colour map is presented in Algorithm 1.

Algorithm 1: Colour map generation.

```
Input: $\{b_0, b_1, \ldots, b_N\}$—base colours; $\{q_0, q_1, \ldots, q_N\}$—colour ratios such that
       $q_0 < q_1 < \ldots < q_N$; $N_c$—the number of colours in the resulting colour map;
       $E$—exponent.
Output: $colourTable$—colour map.
1 $colourTable$ = empty array with $N_c$ elements
2 $k = 1$
3 for $i = 0, 1, \ldots, N_c - 1$ do
4     $q = \left(\frac{i}{1+i}\right)^E$
5     while $q > q_k$ do
6         $k = k + 1$
7     $t = \frac{q_k - q}{q_k - q_{k-1}}$
8     $colourTable[i] = tb_{k-1} + (1-t)b_k$
```

The pseudocode of the method introduced by Lu et al. in [17] is presented in Algorithm 2. In this algorithm, the following notation $p_{i,x}$ and $p_{i,y}$ is used to denote the x and y coordinates of a point p_i. Moreover, f is one of the functions given by (1) or (7) or (8). Lu et al. in their paper set $x_{min} = y_{min} = -2\pi$, $x_{max} = y_{max} = 2\pi$ for most of their examples.

When using the orbit trap method from Algorithm 2 to generate patterns with wallpaper symmetry, we need to point out one thing that was not mentioned in [17]. Namely, the symmetry type of f does not implicate that the generated pattern will be of the same symmetry type. We need to select a proper combination of the orbit traps and the type of the function f to get a desired symmetry type. In Section 4, we will present an example showing how the combination of the orbit traps and the function f affects the resulting pattern and changes its symmetry type.

Algorithm 2: Orbit trap method.

Input: $f : \mathbb{R}^2 \to \mathbb{R}^2$—mapping with the desired symmetry type; $C_0 = (c_0, r_0)$, $C_1 = (c_1, r_1), \ldots, C_n = (c_n, r_n)$—orbit traps, where c_i is the centre and r_i is the radius of the ith circle; $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$—area that should be drawn; $colourTable$—colour map with N_c colours generated using Algorithm 1; M—the maximum number of iterations; W, H—the width and height of the image; E—exponent.

Output: Image generated with the orbit trap method.

```
for $x = 0, 1, \ldots, W-1$ do
    for $y = 0, 1, \ldots, H-1$ do
        $x_0 = x_{min} + (x_{max} - x_{min})\frac{x}{W-1}$
        $y_0 = y_{min} + (y_{max} - y_{min})(1 - \frac{y}{H-1})$
        $p_0 = [x_0, y_0]^T$
        $trapped = false$
        for $k = 0, 1, \ldots, M-1$ do
            $p_{k+1} = f(p_k)$
            for $i = 0, 1, \ldots, n$ do
                $d = \sqrt{(p_{k+1,x} - c_{i,x})^2 + (p_{k+1,y} - c_{i,y})^2}$
                if $d < r_i$ then
                    $trapped = true$
                    break
            if $trapped$ then
                $v = 10^{\log_{10}(\frac{d}{r_i})/E}$
                $index = \min(\frac{v}{1-v}, N_c - 1)$
                colour $(x, y)$ with colour $colourTable[index]$
                break
        if $not\ trapped$ then
            colour $(x, y)$ with colour $colourTable[N_c - 1]$
```

3. Modifications of the Orbit Trap Method

When we look at Algorithm 2, we notice that it uses a feedback process of the following form:

$$p_{k+1} = f(p_k) \text{ for } k = 0, 1, \ldots, \tag{10}$$

where p_0 is a starting point. This type of feedback process is called the Picard iteration [20], and it is used in various algorithms, e.g., polynomial root-finding using numerical algorithms [21], generation of fractals [5,22], etc.

One of the most important applications of Picard's iteration is finding the fixed points of a contractive mapping using Banach Fixed Point Theorem [20]. For instance, this application allows for the generation of fractals given by iterated function systems [23]. In fixed point theory, we can find many other iterations that are used to approximately find fixed points not only for contractive mapping but also for pseudo-contractive or non-expansive mappings [20]. We will modify the orbit trap method presented in Section 2 by replacing the Picard iteration with the so-called S-iteration, which was defined by Agarwal et al. in 2007 [18].

Let (X, d) be a metric space, $T : X \to X$ be a mapping and $p_0 \in X$ be a starting point. Then, the S-iteration is defined in the following way [18]:

$$\begin{aligned} p_{k+1} &= (1 - \alpha_k)T(p_k) + \alpha_k T(u_k), \\ u_k &= (1 - \beta_k)p_k + \beta_k T(p_k), \end{aligned} \tag{11}$$

where $\alpha_k \in (0,1]$ and $\beta_k \in [0,1]$ for all $k \in \mathbb{N}$. Let us notice that the S-iteration reduces to the Picard iteration if $\alpha_k = 1$ and $\beta_k = 0$ for all $k \in \mathbb{N}$. Moreover, it reduces to the Picard–Mann iteration [24] if $\alpha_k = 1$ and $\beta_k \neq 0$ for all $k \in \mathbb{N}$.

To use (11) in the orbit trap method, we need to specify a metric space and the mapping T. We take the metric space $(\mathbb{R}^2, d)$, where d is any metric in $\mathbb{R}^2$, and as the mapping, we take $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by the formulas introduced in Section 2. In (11), we use sequences of parameters α_k and β_k, but for simplicity we will use constant sequences, i.e., $\alpha_k = \alpha$ and $\beta_k = \beta$. The conditions about the range of the parameters in (11) were introduced in fixed point theory to obtain convergence to a fixed point of the mapping. In the orbit trap method, we are not interested in finding fixed points. Instead, we are interested in generating interesting artistic patterns, so we can omit these conditions and take any real values. We can even go further and take α and β as two-dimensional points from $\mathbb{R}^2$, but in such a case, we need to define multiplication in order to be able to perform calculations in the S-iteration. Thus, we define the multiplication in a similar way as for complex numbers, i.e.,

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 x_2 - y_1 y_2 \\ x_1 y_2 + y_1 x_1 \end{bmatrix}. \tag{12}$$

The last thing that we need to consider—if we want to use S-iteration in the orbit trap method—is the formula for p_{k+1}. In the orbit trap method, we generate patterns with wallpaper symmetry, so in each form of the f function, we calculate the final result modulo 2π. Therefore, in the formula for p_{k+1} in (11), we need to do the calculations modulo 2π.

The modified S-iteration that we will use in the orbit trap method has the following form:

$$\begin{aligned} p_{k+1} &= ([1,0]^T - \alpha) \cdot f(p_k) + \alpha \cdot f(u_k) \mod [2\pi, 2\pi]^T, \\ u_k &= ([1,0]^T - \beta) \cdot p_k + \beta \cdot f(p_k), \end{aligned} \tag{13}$$

where $p_0 \in \mathbb{R}^2$ is a starting point; $\alpha, \beta \in \mathbb{R}^2$ are parameters; and $f : \mathbb{R}^2 \to \mathbb{R}^2$ is the mapping introduced in Section 2. Let us notice that when the parameters α and β are of the following form: $\alpha = [x_\alpha, 0]^T$, $\beta = [x_\beta, 0]^T$, where $x_\alpha, x_\beta \in \mathbb{R}$, then (13) reduces to the S-iteration with scalar parameters $\alpha = x_\alpha, \beta = x_\beta$. Therefore, for $\alpha = [1,0]^T$ and $\beta = [0,0]^T$, iteration (13) reduces to the standard Picard iteration.

Looking again at Algorithm 2 and the orbit traps presented in Figure 1, we see that the circles forming the orbit traps are defined in a metric space in which we use the Euclidean metric. When introducing the S-iteration to the orbit trap method, we assumed that we take the metric space $(\mathbb{R}^2, d)$, where d is any metric in $\mathbb{R}^2$. Thus, the circles will have different shapes for various metrics. In the rest of the paper, we will use three metrics defined in $\mathbb{R}^2$ [25]:

- l_p-metric, where $p \in [1, \infty)$

$$d_p([x_1, y_1]^T, [x_2, y_2]^T) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{\frac{1}{p}}, \tag{14}$$

- Chebyshev metric (or l_∞-metric)

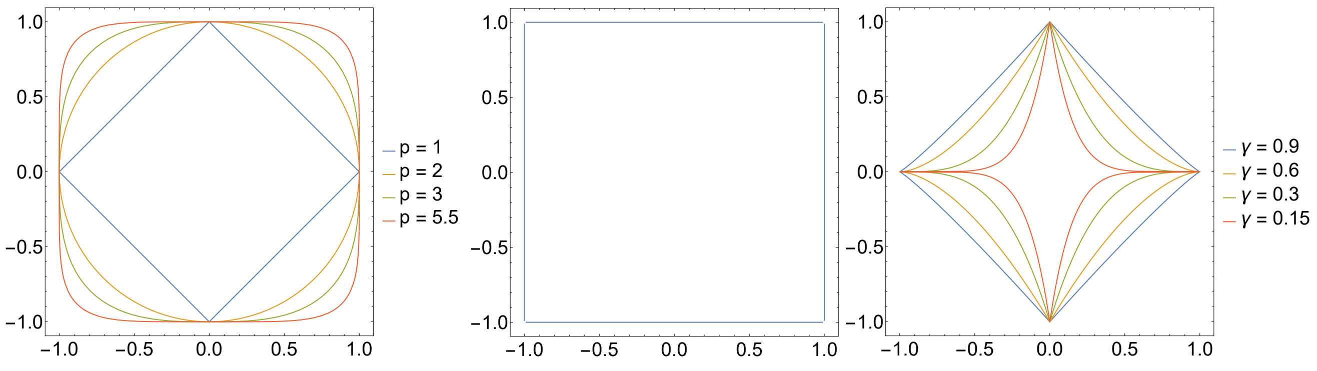
$$d_\infty([x_1, y_1]^T, [x_2, y_2]^T) = \max\{|x_1 - x_2|, |y_1 - y_2|\}, \tag{15}$$

- Rickman's rug metric

$$d_\gamma([x_1, y_1]^T, [x_2, y_2]^T) = |x_1 - x_2| + |y_1 - y_2|^\gamma, \tag{16}$$

where $\gamma \in (0,1)$.

Examples of circles with centre $[0,0]^T$ and radius 1 obtained with metrics (14)–(16) are presented in Figure 2.



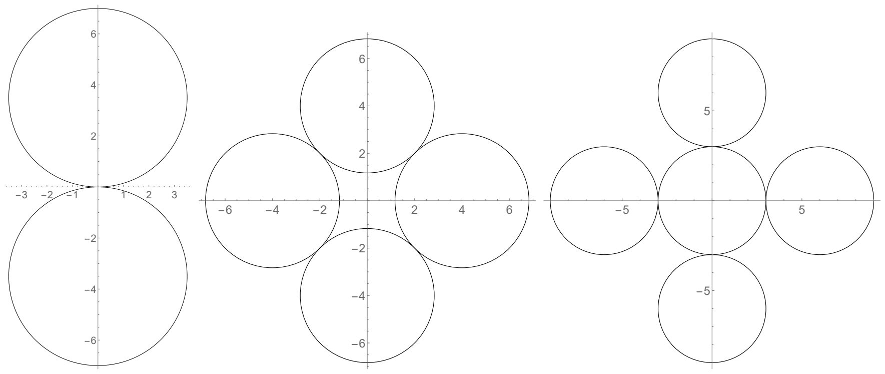
(**a**) (**b**) (**c**)

Figure 2. Circles with centre $[0,0]^T$ and radius 1 obtained with various metrics. (**a**) l_p; (**b**) Chebyshev; and (**c**) Rickman.

Besides the use of the S-iteration and various metrics for orbit traps, we also introduce three new orbit traps:

(a) $c_0 = [0, 3.5]^T, c_1 = [0, -3.5]^T, r_0 = r_1 = 3.5,$
(b) $c_0 = [4, 0]^T, c_1 = [-4, 0]^T, c_2 = [0, 4]^T, c_3 = [0, -4]^T, r_0 = r_1 = r_2 = r_3 = 2\sqrt{2},$
(c) $c_0 = [0, 0]^T, c_1 = [-6, 0]^T, c_2 = [6, 0]^T, c_3 = [0, -6]^T, c_4 = [0, 6]^T, r_0 = r_1 = \ldots = r_4 = 3.$

Figure 3 presents the new orbit traps obtained with the Euclidean metric.



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

Figure 3. New orbit traps obtained with the Euclidean metric. (**a**) two orbit traps; (**b**) four orbit traps; and (**c**) five orbit traps.

In Algorithm 3, we present the pseudocode for the modified version of the orbit trap method. We can implement this algorithm on the GPU (graphics processing unit) using shaders instead of the CPU (central processing unit) implementation presented in [17]. In the implementation, we render a quad that occupies the whole space in the window. Then, in the vertex shader, we calculate the coordinates of area corners and let the rasterizer calculate the coordinates of the starting point p_0, which will be an input variable for the fragment shader. In the fragment shader, we make all the calculations that start from line 6 in the algorithm. Moreover, to enhance the quality of the generated pattern, we can use multisampling to perform the anti-aliasing that was not used in the original method presented in [17].

Algorithm 3: Modified orbit trap method.

Input: $f : \mathbb{R}^2 \to \mathbb{R}^2$—mapping with the desired symmetry type; $C_0 = (c_0, r_0)$, $C_1 = (c_1, r_1), \ldots, C_n = (c_n, r_n)$—orbit traps, where c_i is the centre and r_i is the radius of the ith circle; $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$—area which should be drawn; *colourTable*—colour map with N_c colours generated using Algorithm 1; M—the maximum number of iterations; W, H—the width and height of the image; E—exponent; $d : \mathbb{R}^2 \times \mathbb{R}^2 \to [0, \infty)$—metric; $\alpha, \beta \in \mathbb{R}^2$—parameters for the S-iteration.

Output: Image generated with the modified orbit trap method.

```
for $x = 0, 1, \ldots, W-1$ do
    for $y = 0, 1, \ldots, H-1$ do
        $x_0 = x_{min} + (x_{max} - x_{min}) \frac{x}{W-1}$
        $y_0 = y_{min} + (y_{max} - y_{min})(1 - \frac{y}{H-1})$
        $p_0 = [x_0, y_0]^T$
        $trapped = false$
        for $k = 0, 1, \ldots, M-1$ do
            $u_k = ([1,0]^T - \beta) \cdot p_k + \beta \cdot f(p_k)$
            $p_{k+1} = ([1,0]^T - \alpha) \cdot f(p_k) + \alpha \cdot f(u_k) \mod [2\pi, 2\pi]^T$
            for $i = 0, 1, \ldots, n$ do
                $d = d(p_{k+1}, c_i)$
                if $d < r_i$ then
                    $trapped = true$
                    break
            if $trapped$ then
                $v = 10^{\log_{10}(\frac{d}{r_i})/E}$
                $index = \min(\frac{v}{1-v}, N_c - 1)$
                colour $(x, y)$ with colour $colourTable[index]$
                break
        if not $trapped$ then
            colour $(x, y)$ with colour $colourTable[N_c - 1]$
```

4. Examples

In this section, we present some examples of patterns obtained using the modified orbit trap method presented in Section 3. In all examples, we use multisampling with eight samples per pixel to perform anti-alising for enhancing the quality of the generated patterns. The algorithm for generating the patterns was written in OpenGL Shading Language (GLSL).

In the examples, we use four colour maps generated using Algorithm 1. The maps are presented in Figure 4, and the parameters used to generate them were the following: $E = 16$, $N_c = 3000$, and the base colours together with the colour ratios are gathered in Table 1.

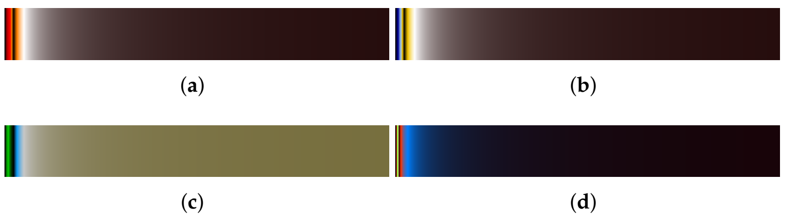


Figure 4. Colour maps used in the examples.

Table 1. Base colours and colour ratios used to generate colour maps in Figure 4.

Base Colour	Colour Ratio
(a)	
$(0, 0, 0)$	0.0
$(255, 0, 0)$	0.7
$(255, 128, 0)$	0.75
$(0, 0, 0)$	0.8
$(255, 128, 0)$	0.85
$(255, 255, 255)$	0.9
$(26, 0, 0)$	1.0
(b)	
$(0, 0, 0)$	0.0
$(0, 0, 150)$	0.5
$(161, 202, 241)$	0.7
$(243, 195, 0)$	0.75
$(0, 0, 0)$	0.8
$(243, 195, 0)$	0.85
$(255, 255, 255)$	0.9
$(26, 0, 0)$	1.0
(c)	
$(0, 0, 0)$	0.0
$(6, 204, 2)$	0.6
$(0, 0, 0)$	0.8
$(19, 158, 245)$	0.85
$(200, 200, 200)$	0.9
$(115, 106, 55)$	1.0
(d)	
$(0, 0, 0)$	0.0
$(254, 0, 0)$	0.25
$(0, 255, 0)$	0.3
$(255, 255, 127)$	0.47
$(25, 0, 0)$	0.6
$(255, 52, 0)$	0.7
$(0, 128, 255)$	0.85
$(25, 0, 0)$	1.0

In Section 2, we mentioned about the proper combination of the orbit traps and the type of the function f in order to get the desired type of wallpaper symmetry. Thus, the first example presents this property. In the example, we generate patterns using the same function f but with various orbit traps that we introduced in Figures 1 and 3. The obtained patterns are presented in Figure 5, and the parameters used to generate them were the following: colour map from Figure 4b, area $[-2\pi, 2\pi]^2$, $M = 100$, Picard iteration (i.e., $\alpha = [1, 0]^T$, $\beta = [0, 0]^T$), Euclidean metric, function f given by (8) with p3m1 symmetry and $G_{v_0} = [0.343244, 0.711418]^T$, $G_{v_1} = [-0.671499, -0.289951]^T$, $G_{v_2} = [-0.959405, -0.868905]^T$, $H_{v_0} = [-0.224996, 0.0844803]^T$, $H_{v_1} = [-0.0571376, 0.0250126]^T$, and $H_{v_2} = [0.0250892, 0.553883]^T$. From the images in Figure 5, we see that only Figure 5a,c,g have the same symmetry type as f, namely the p3m1 symmetry. All the other images have another type of symmetry, namely, the cm symmetry. When we look at the orbit traps used to obtain the patterns with p3m1 symmetry, then we notice that they have the same rotational symmetries and reflections as the p3m1 symmetry. Thus, to generate a pattern with a given symmetry type, the function f and the orbit traps should satisfy similar symmetry conditions. Moreover, from this example, we can observe one more interesting thing. When we look at the orbit traps in Figures 1c and 3b, we see that they are the same circles but rotated by 45°. However, if we look at the patterns generated using them with the same other parameters (see Figure 5d,e), then we notice that the patterns are completely different. Thus, a modification of the orbit traps obtained by rotation could lead to a significant shape change of the resulting pattern.

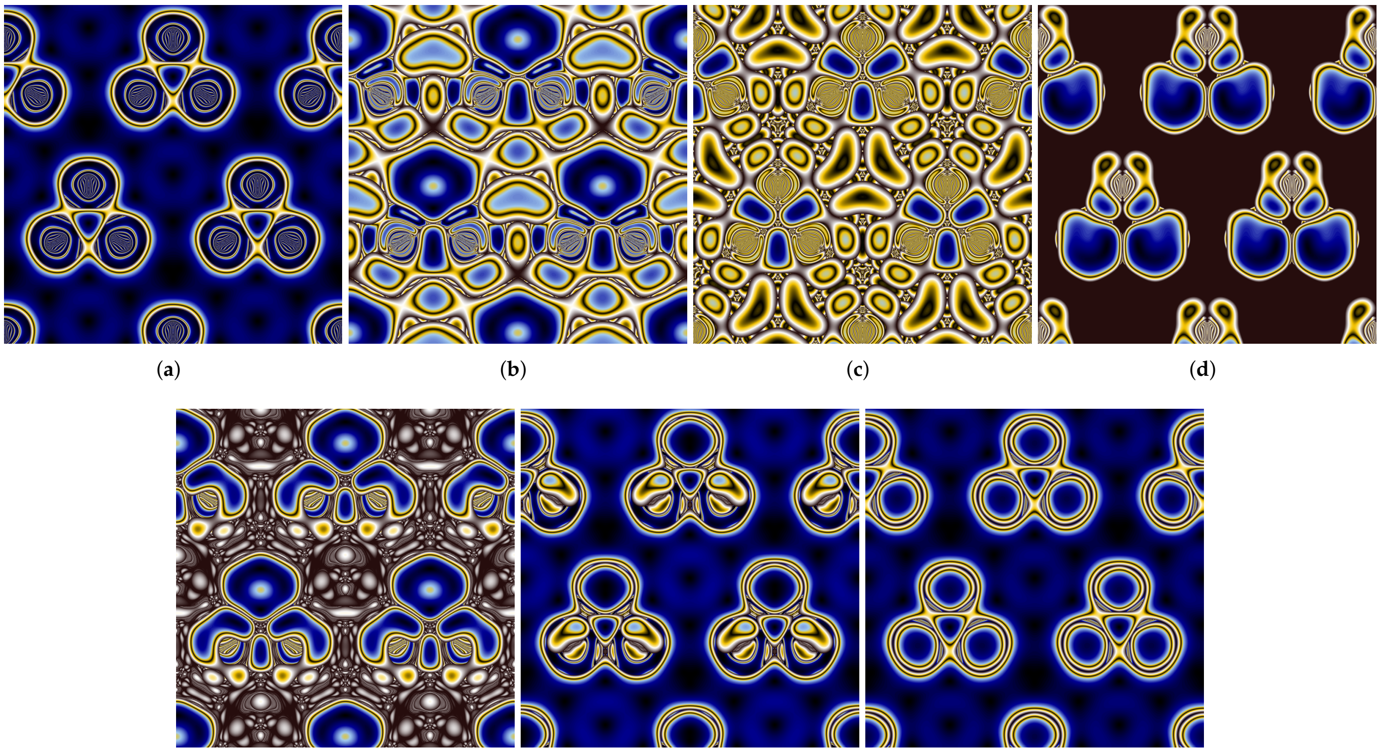

(**a**) (**b**) (**c**) (**d**) (**e**) (**f**) (**g**)

Figure 5. Patterns generated with function f with p3m1 symmetry and various orbit traps from Figures 1 and 3. (**a**) 1 orbit trap from Figure 1a; (**b**) 2 orbit traps from Figure 3a; (**c**) 3 orbit traps from Figure 1b; (**d**) 4 orbit traps from Figure 1c; (**e**) 4 orbit traps from Figure 3b; (**f**) 5 orbit traps from Figure 3c; and (**g**) 7 orbit traps from Figure 1d.

In the next example, we present the effect of changing the metric used in the definition of the orbit traps. To show this, we fix orbit traps and change the metric and its parameter, i.e., p for the l_p metric and γ for the Rickman's rug metric. In the example, we use orbit traps from Figure 3a, and the other parameters defining the pattern were the following: area $[-2\pi, 2\pi]^2$, $M = 100$, $E = 16$, Picard iteration ($\alpha = 1$, $\beta = 0$ in the S-iteration), colour map from Figure 4c, and function f given by (1) with the cmm symmetry defined by the coefficients (we give only the non-zero elements of the matrix that defines (1)) $P_{04} = [0, -0.890893]^T$, $P_{13} = [0, 0.594811]^T$, $P_{24} = [0, 0.166299]^T$, $P_{31} = [2.33836, 0]^T$, $P_{40} = [-1.53665, 0]^T$, and $P_{42} = [0.149503, 0]^T$. In Figure 6, we show patterns obtained with the l_p metric with the following values of p: (a) 1.0, (b) 1.5, (c) 2.0, (d) 2.5, (e) 3.0, and (f) ∞, whereas in Figure 7, we see patterns generated using Rickman's rug metric with the following values of γ: (a) 0.9, (b) 0.7, (c) 0.5, and (d) 0.1. The image from Figure 6c was obtained with the Euclidean metric that was used in the original method presented in [17]. It will serve as a reference point. When we increase the value of p over 2.0 (Figure 6d–f), we see that some of the areas of the pattern are gradually smooth out. This change is continuous, and the smoothing becomes bigger with the increase of the p value, obtaining in the limit the pattern presented in Figure 6f. When we decrease p below 2.0 and tend to 1.0 (Figure 6a,b), we see that more details appear in the pattern compared to the pattern obtained with the original method. If we look at the images obtained with the Rickman's rug metric, then we can also notice a smoothing effect with the decrease of the γ value. However, some other details are smoothed out compared to the l_p metric. Therefore, we see that with the use of various metrics, we can obtain very interesting patterns that were not possible to obtain with the Euclidean metric.

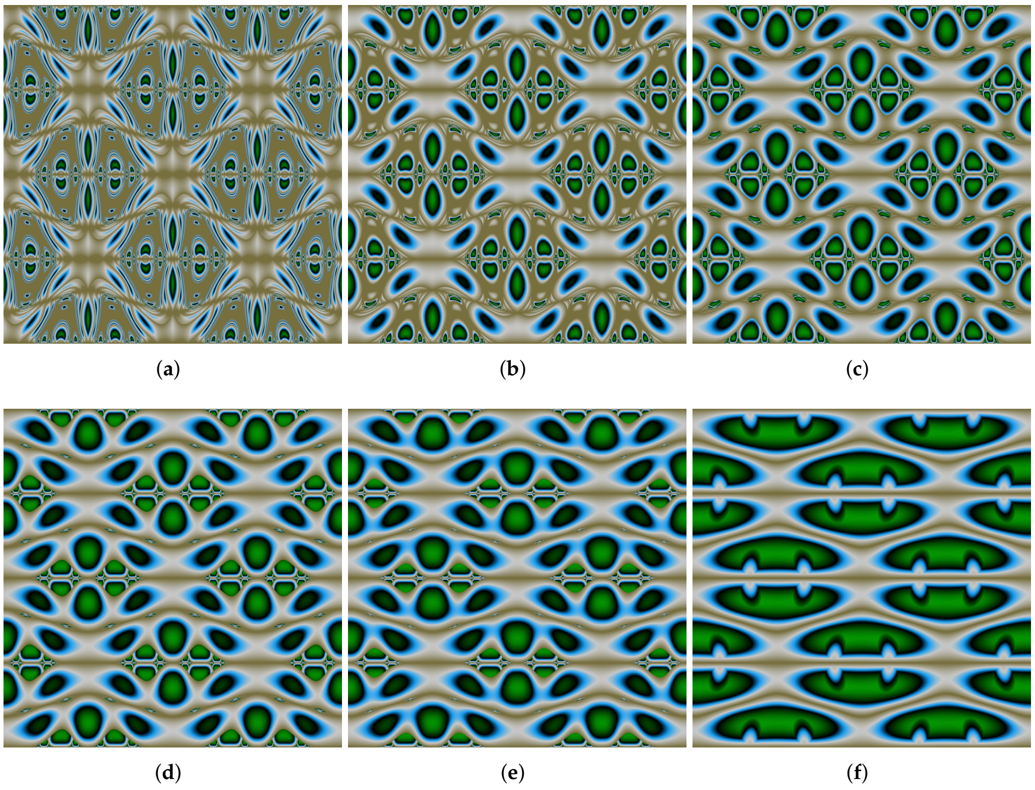


Figure 6. Patterns generated with function f with cmm symmetry and orbit traps from Figure 3a with various values of p in the l_p metric. (**a**) $l_{1.0}$; (**b**) $l_{1.5}$; (**c**) $l_{2.0}$; (**d**) $l_{2.5}$; (**e**) $l_{3.0}$; and (**f**) l_{∞}.

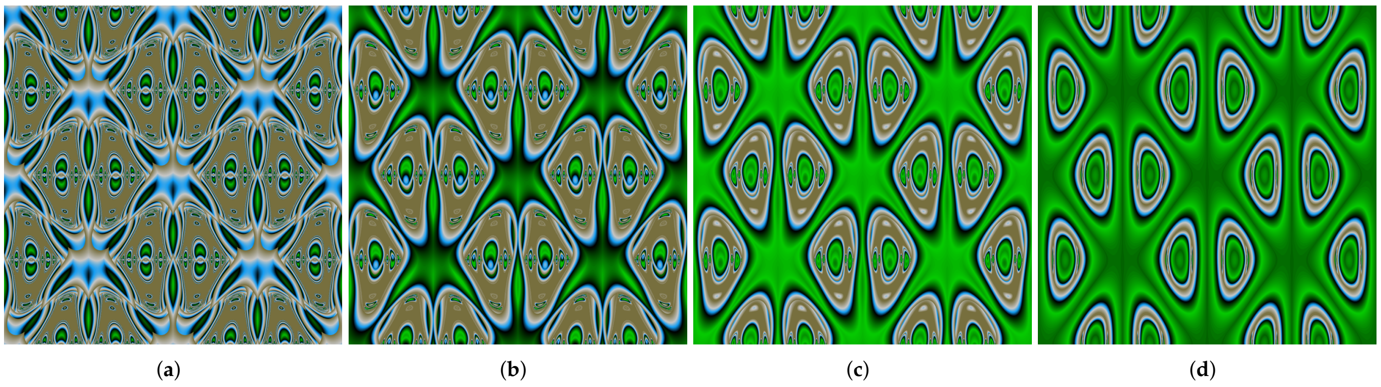


Figure 7. Patterns generated with function f with cmm symmetry and orbit traps from Figure 3a with various values of γ in the Rickman's rug metric. (**a**) $\gamma = 0.9$; (**b**) $\gamma = 0.7$; (**c**) $\gamma = 0.5$; and (**d**) $\gamma = 0.1$.

The next two examples show the use of the S-iteration. In the first example, we use S-iteration with scalar parameters α and β, which corresponds to the case in which we vary the x coordinate and take $y = 0$ of the parameters. In the second example, we use S-iteration, in which the β parameter is fixed and the α parameter has fixed x coordinate and varying y coordinate.

In the first example with the S-iteration, we generate patterns for scalar values of the iteration's parameters, i.e., the parameters are of the form $\alpha = [x_\alpha, 0]^T$, $\beta = [x_\beta, 0]^T$, where $x_\alpha, x_\beta \in \mathbb{R}$. The generated patterns are presented in Figure 8, where the following parameters were used in the generation process: orbit traps from Figure 3b with Euclidean metric, area $[-2\pi, 2\pi]^2$, $M = 100$, $E = 16$, colour map from Figure 4a, and function f given by (1) with the p4m symmetry defined by the coefficients (again we give only the non-zero elements of the matrix) $P_{03} = [0, -0.234602]^T$, $P_{04} = [0, -0.231376]^T$, $P_{13} = [0, -0.931333]^T$, $P_{14} = [0, -0.982975]^T$, $P_{23} = [0, 0.429163]^T$, $P_{24} = [0, 0.363704]^T$, $P_{30} = [-0.234602, 0]^T$, $P_{31} = [-0.931333, 0]^T$, $P_{32} = [0.422963, 0]^T$, $P_{40} = [-0.231376, 0]^T$, $P_{41} = [-0.982975, 0]^T$, and $P_{42} = [0.363704, 0]^T$. From the images, we see that by changing the scalar values of the α and β parameters in the S-iteration, we are able to generate a great variety of patterns. The shapes of the patterns can be very simple, as in Figure 8a, or very complex, as in Figure 8d.

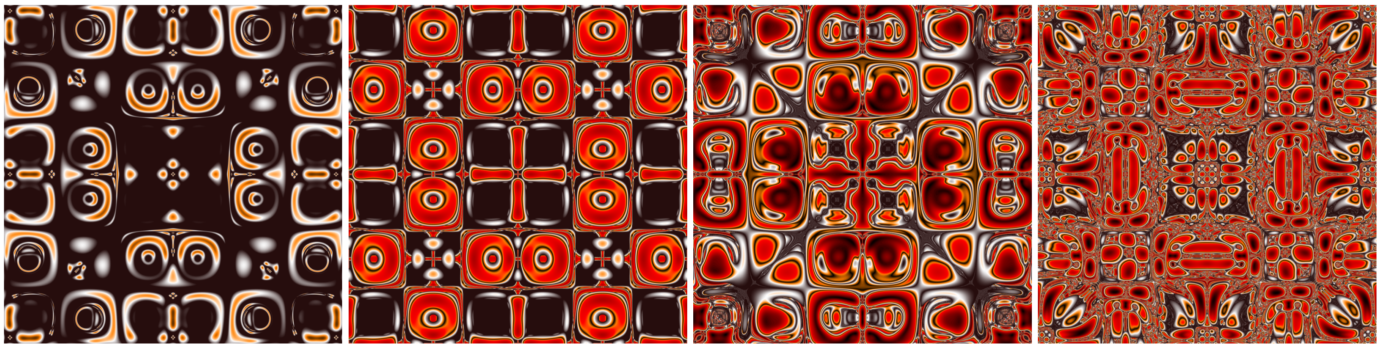

(**a**) (**b**) (**c**) (**d**)

Figure 8. Patterns generated with function f with p4m symmetry, orbit traps from Figure 3b, and S-iteration with varying scalar α and β parameters. (**a**) $\alpha = [1.1, 0]^T$, $\beta = [1.1, 0]^T$; (**b**) $\alpha = [1.5, 0]^T$, $\beta = [1, 0]^T$; (**c**) $\alpha = [-1.5, 0]^T$, $\beta = [1.3, 0]^T$; and (**d**) $\alpha = [2, 0]^T$, $\beta = [-1.1, 0]^T$.

In the second example, with the S-iteration, we generate patterns in which we vary the y coordinate of the α parameter, and the other parameters are fixed. In this way, we show that not only scalar parameters generate interesting patterns but also the vector parameters. In the example, we used the following parameters: $\beta = [0.5, 0]^T$ in the S-iteration, orbit traps from Figure 1c with Euclidean metric, area $[-2\pi, 2\pi]^2$, $M = 100$, $E = 16$, colour map from Figure 4d, and function f given by (1) with the p4 symmetry defined by the coefficients (again we give only the non-zero elements of the matrix) $P_{03} = [0.215989, -0.150289]^T$, $P_{04} = [0.0748772, 0.670663]^T$, $P_{13} = [0.741719, 0.543639]^T$, $P_{14} = [0.709028, -0.822211]^T$, $P_{23} = [0.708072, -0.814562]^T$, $P_{24} = [0.100835, 0.40511]^T$, $P_{30} = [-0.150289, -0.215989]^T$, $P_{31} = [0.543639, -0.741719]^T$, $P_{32} = [-0.814562, -0.708072]^T$, $P_{40} = [0.670663, -0.0748772]^T$, $P_{41} = [-0.822211, -0.709028]^T$, and $P_{42} = [0.40511, -0.100835]^T$. In Figure 9, we see patterns generated with the following values of the α parameter: (a) $[1.1, 0.5]^T$, (b) $[1.1, 0.8]^T$, (c) $[1.1, -0.8]^T$, and (d) $[1.1, 4]^T$. Similar to the scalar case, when varying the y coordinate of the α parameter, we are able to obtain a great variety of patterns. The generated patterns vary not only in shape but also in colouring. In Figure 9, we see a greater variety of colours than in Figure 8.

In the end, we present an interesting observation. When using orbit traps in which we have an empty space in the neighbourhood of the origin that is surrounded by the orbit traps (see orbit traps in Figures 1b,c and 3b) and function f with compatible symmetry type, then in the area near the origin of the generated pattern, we find self-similar structures. Thus, the patterns have a local fractal structure. To observe this, in Figure 10, we present magnification of the central part of the patterns from Figures 5c, 8c and 9a. If we magnify these areas further, then we get the same images repeatedly, so we clearly see the local self-similarity property.

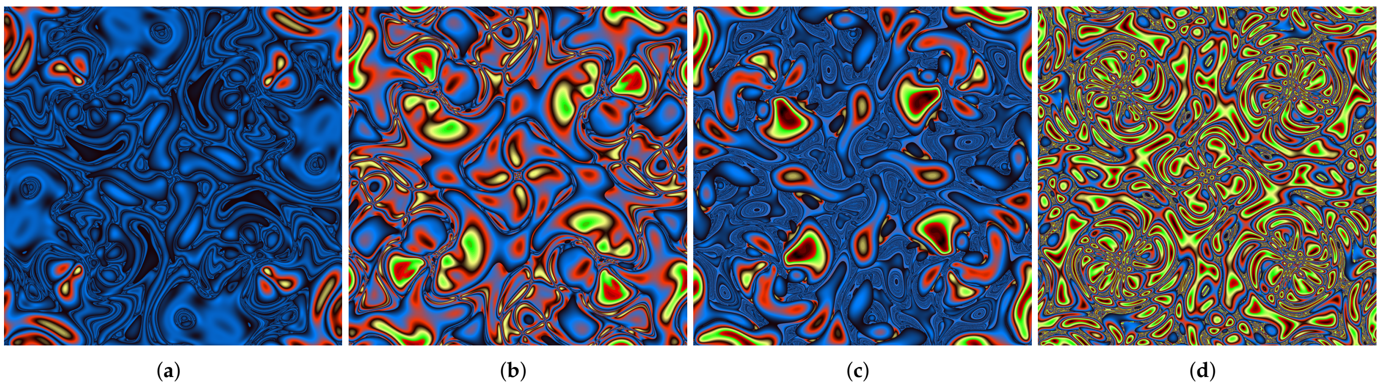


Figure 9. Patterns generated with function f with p4 symmetry, orbit traps from Figure 1c, and S-iteration with $\beta = [0.5, 0]^T$ and varying α parameter. (**a**) $\alpha = [1.1, 0.5]^T$; (**b**) $\alpha = [1.1, 0.8]^T$; (**c**) $\alpha = [1.1, -0.8]^T$; and (**d**) $\alpha = [1.1, 4]^T$.

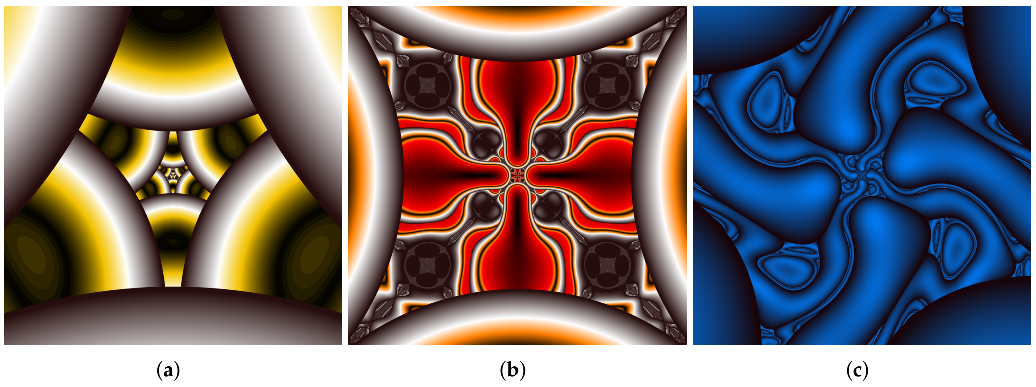


Figure 10. Magnifications of the areas near the origin of patterns from Figures 5c, 8c and 9a. These magnifications show the local self-similarity property of the patterns. (**a**) Figure 5c; (**b**) Figure 8c; and (**c**) Figure 9a.

5. Conclusions

In this paper, we proposed extensions of the orbit trap method introduced in [17]. The extensions rely on the use of (1) new orbit traps, (2) various metrics instead of the Euclidean one, and (3) the S-iteration. Moreover, we extended the S-iteration from scalar to vector parameters. The presented examples showed that we are able to generate patterns, which we had not been able to obtain previously.

In our extensions, we used only the S-iteration. In the literature, there are many other iteration methods. For example, in [26], we can find a review of 17 different iterations and their dependencies. Moreover, there are iteration methods that use several mappings [14]. Thus, an interesting direction for further study would be the use of other iteration methods.

Not all values of the parameters defining the f function that satisfy the appropriate symmetry condition give rise to interesting and artistic patterns [15]. To find the values that will give artistic patterns, Carter et al. in [15] used a simple Monte Carlo method, so the evaluation of the pattern was made by the authors. Therefore, another direction for future studies would be developing an automatic method for finding patterns with artistic features. For instance, we could try to use genetic algorithms for this task because this type of method has proven itself in various methods that generate artistic images [27–29].

Author Contributions: Conceptualization, K.G. and H.A.; methodology, K.G.; software, K.G.; investigation, K.G.; writing—original draft preparation, K.G.; writing—review and editing, K.G. and H.A.; visualization, K.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Natural Science Foundation of China grant number 62062042.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Appendix A. Symmetry Conditions

In this appendix, we gathered the conditions that function f given by (1) must satisfy to generate a pattern with one of the symmetry types: p2, pm, pg, cm, cmm, pmm, pmg, pgg, p4, p4m, and p4g.

The symmetry of the considered function f depends on the matrix:

$$P = \begin{bmatrix} P_{00} & \dots & P_{04} \\ \vdots & \vdots & \vdots \\ P_{40} & \dots & P_{44} \end{bmatrix},$$

where $P_{00}, P_{01}, \dots, P_{44} \in \mathbb{R}^2$. For simplicity, we present the conditions in the form of a matrix M (a mask) that has the same dimensions as P. To obtain a matrix P' that will define f with symmetry type given by the mask M, we need to calculate $P' = P \circ M$, where $\circ$ is element-wise product of matrices and vectors.

Let us introduce the following notation:

$$\mathbf{0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \mathbf{1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; \uparrow = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; \downarrow = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The masks for the p2, pm, pg, cm, cmm, pmm, pmg, and pgg symmetries are the following [15]:

$$M_{p2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \; M_{pm} = \begin{bmatrix} \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \end{bmatrix}, \; M_{pg} = \begin{bmatrix} \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \downarrow & \downarrow & \downarrow & \uparrow & \uparrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \\ \downarrow & \downarrow & \downarrow & \uparrow & \uparrow \\ \uparrow & \uparrow & \uparrow & \downarrow & \downarrow \end{bmatrix}$$

$$M_{cm} = \begin{bmatrix} \uparrow & \mathbf{0} & \uparrow & \mathbf{0} & \downarrow \\ \mathbf{0} & \uparrow & \mathbf{0} & \downarrow & \mathbf{0} \\ \uparrow & \mathbf{0} & \uparrow & \mathbf{0} & \downarrow \\ \mathbf{0} & \uparrow & \mathbf{0} & \downarrow & \mathbf{0} \\ \uparrow & \mathbf{0} & \uparrow & \mathbf{0} & \downarrow \end{bmatrix}, \; M_{cmm} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow \\ \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \uparrow & \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} \end{bmatrix}, \; M_{pmm} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow & \downarrow \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow & \downarrow \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \downarrow & \downarrow \\ \uparrow & \uparrow & \uparrow & \mathbf{0} & \mathbf{0} \\ \uparrow & \uparrow & \uparrow & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$M_{pmg} = \begin{bmatrix} \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \downarrow \\ \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \downarrow \\ \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \downarrow \\ \uparrow & \mathbf{0} & \uparrow & \downarrow & \mathbf{0} \\ \uparrow & \mathbf{0} & \uparrow & \downarrow & \mathbf{0} \end{bmatrix}, \; M_{pgg} = \begin{bmatrix} \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \downarrow \\ \downarrow & \mathbf{0} & \downarrow & \uparrow & \mathbf{0} \\ \mathbf{0} & \uparrow & \mathbf{0} & \mathbf{0} & \downarrow \\ \mathbf{0} & \downarrow & \mathbf{0} & \mathbf{0} & \uparrow \\ \uparrow & \mathbf{0} & \uparrow & \downarrow & \mathbf{0} \end{bmatrix}$$

In the case of the p4, p4m, and p4g symmetries, we cannot give the conditions in the form of masks because the coefficients of P must satisfy some symmetry conditions that cannot be placed in the mask. Therefore, for these three symmetry types, we directly give the form of the matrix P [15]:

$$P_{p4} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} m_{00} \\ m_{01} \end{bmatrix} & \begin{bmatrix} m_{10} \\ m_{11} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} m_{20} \\ m_{21} \end{bmatrix} & \begin{bmatrix} m_{30} \\ m_{31} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} m_{40} \\ m_{41} \end{bmatrix} & \begin{bmatrix} m_{50} \\ m_{51} \end{bmatrix} \\ \begin{bmatrix} m_{01} \\ -m_{00} \end{bmatrix} & \begin{bmatrix} m_{21} \\ -m_{20} \end{bmatrix} & \begin{bmatrix} m_{41} \\ -m_{40} \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} m_{11} \\ -m_{10} \end{bmatrix} & \begin{bmatrix} m_{31} \\ -m_{30} \end{bmatrix} & \begin{bmatrix} m_{51} \\ -m_{50} \end{bmatrix} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$P_{p4m} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 \\ m_{01} \end{bmatrix} & \begin{bmatrix} 0 \\ m_{11} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 \\ m_{21} \end{bmatrix} & \begin{bmatrix} 0 \\ m_{31} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 \\ m_{41} \end{bmatrix} & \begin{bmatrix} 0 \\ m_{51} \end{bmatrix} \\ \begin{bmatrix} m_{01} \\ 0 \end{bmatrix} & \begin{bmatrix} m_{21} \\ 0 \end{bmatrix} & \begin{bmatrix} m_{41} \\ 0 \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} m_{11} \\ 0 \end{bmatrix} & \begin{bmatrix} m_{31} \\ 0 \end{bmatrix} & \begin{bmatrix} m_{51} \\ 0 \end{bmatrix} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$P_{p4g} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 \\ m_{11} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} m_{20} \\ 0 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 \\ m_{51} \end{bmatrix} \\ \mathbf{0} & \begin{bmatrix} 0 \\ -m_{20} \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} m_{11} \\ 0 \end{bmatrix} & \mathbf{0} & \begin{bmatrix} m_{51} \\ 0 \end{bmatrix} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $m_{00}, m_{01}, m_{10}, m_{11}, \ldots, m_{50}, m_{51} \in \mathbb{R}$.

References

1. Oxford English Dictonary Online. 2022. Available online: http://www.oed.com (accessed on 20 August 2017).
2. Liu, Y.; Lin, W. *Deformable Texture: The Irregular-Regular-Irregular Cycle*; Technical Report; Carnegie Mellon University: Pittsburgh, PA, USA, 2003.
3. Ebert, D.; Musgrave, F.; Peachey, D.; Perlin, K.; Worley, S. *Texturing and Modeling: A Procedural Approach*, 3rd ed.; Morgan Kaufmann: San Francisco, CA, USA, 2002.
4. Gieseke, L.; Asente, P.; Mech, R.; Benes, B.; Fuchs, M. A Survey of Control Mechanisms for Creative Pattern Generation. *Comput. Graph. Forum* **2021**, *40*, 585–609. [CrossRef]
5. Gdawiec, K. Inversion Fractals and Iteration Processes in the Generation of Aesthetic Patterns. *Comput. Graph. Forum* **2017**, *36*, 35–45. [CrossRef]
6. Helt, G. Extending Mandelbox Fractals with Shape Inversions. In Proceedings of the Bridges 2018: Mathematics, Art, Music, Architecture, Education, Culture; Torrence, E., Torrence, B., Séquin, C., Fenyvesi, K., Eds.; Tessellations Publishing: Phoenix, AZ, USA, 2018; pp. 547–550. Available online: http://archive.bridgesmathart.org/2018/bridges2018-547.pdf (accessed on 1 February 2022).
7. Liu, S.; Leng, M.; Ouyang, P. The Visualization of Spherical Patterns with Symmetries of the Wallpaper Group. *Complexity* **2018**, *2018*, 7315695. [CrossRef]
8. Ouyang, P.; Fathauer, R.; Chung, K.; Wang, X. Automatic Generation of Hyperbolic Drawings. *Appl. Math. Comput.* **2019**, *347*, 653–663. [CrossRef]

9. Mitchell, K. Fun with Whirls. In *Proceedings of the Bridges 2015: Mathematics, Music, Art, Architecture, Culture*; Delp, K., Kaplan, C., McKenna, D., Sarhangi, R., Eds.; Tessellations Publishing: Phoenix, AZ, USA, 2015; pp. 175–182. Available online: http://archive.bridgesmathart.org/2015/bridges2015-175.html (accessed on 1 February 2022).
10. Ouyang, P.; Tang, X.; Chung, K.; Yu, T. Spiral Patterns of Color Symmetry from Dynamics. *Nonlinear Dyn.* **2018**, *94*, 261–272. [CrossRef]
11. Qiu, C.; Li, X.; Pang, J.; Ouyang, P. Visualization of Escher-like Spiral Patterns in Hyperbolic Space. *Symmetry* **2022**, *14*, 134. [CrossRef]
12. Gdawiec, K. Procedural Generation of Aesthetic Patterns from Dynamics and Iteration Processes. *Int. J. Appl. Math. Comput. Sci.* **2017**, *27*, 827–837. [CrossRef]
13. Lu, J.; Ye, Z.; Zou, Y. Automatic Generation of Colorful Patterns with Wallpaper Symmetries from Dynamics. *Vis. Comput.* **2007**, *23*, 445–449. [CrossRef]
14. Gdawiec, K. Fractal Patterns from the Dynamics of Combined Polynomial Root Finding Methods. *Nonlinear Dyn.* **2017**, *90*, 2457–2479. [CrossRef]
15. Carter, N.; Eagles, R.; Grimes, S.; Hahn, A.; Reiter, C. Chaotic Attractors with Discrete Planar Symmetries. *Chaos Solitons Fractals* **1998**, *9*, 2031–2054. [CrossRef]
16. Carlson, P. Two Artistic Orbit Trap Rendering Methods for Newton M-set Fractals. *Comput. Graph.* **1999**, *23*, 925–931. [CrossRef]
17. Lu, J.; Ye, Z.; Zou, Y.; Ye, R. Orbit Trap Rendering Methods for Generating Artistic Images with Crystallographic Symmetries. *Comput. Graph.* **2005**, *29*, 787–794. [CrossRef]
18. Agarwal, R.; O'Regan, D.; Sahu, D. Iterative Construction of Fixed Points of Nearly Asymptotically Nonexpansive Mappings. *J. Nonlinear Convex Anal.* **2007**, *8*, 61–79.
19. Pickover, C. *Computers and the Imagination*; St. Martin's Press: New York, NY, USA, 1992.
20. Berinde, V. *Iterative Approximation of Fixed Points*; Springer: Berlin/Heidelberg, Germany, 2007.
21. Kalantari, B. *Polynomial Root-Finding and Polynomiography*; World Scientific: Singapore, 2009. [CrossRef]
22. Chen, N.; Chen, Y.; Chung, K. Fractals from Nonlinear IFSs of the Complex Mapping Family $f(z) = z^n + c$. *Fractals* **2018**, *26*, 1850044. [CrossRef]
23. Barnsley, M. *Fractal Everywhere: New Edition*; Dover Publications: Mineola, NY, USA, 2012.
24. Khan, S. A Picard-Mann Hybrid Iterative Process. *Fixed Point Theory Appl.* **2013**, *2013*, 69. [CrossRef]
25. Deza, M.; Deza, E. *Encyclopedia of Distances*, 4th ed.; Springer: Berlin/Heidelberg, Germany, 2016.
26. Gdawiec, K.; Kotarski, W. Polynomiography for the Polynomial Infinity Norm via Kalantari's Formula and Nonstandard Iterations. *Appl. Math. Comput.* **2017**, *307*, 17–30. [CrossRef]
27. Lu, S.; Mok, P.; Jin, X. From Design Methodology to Evolutionary Design: An Interactive Creation of Marble-like Textile Patterns. *Eng. Appl. Artif. Intell.* **2014**, *32*, 124–135. [CrossRef]
28. Lv, J.; Zhu, M.; Pan, W.; Liu, X. Interactive Genetic Algorithm Oriented toward the Novel Design of Traditional Patterns. *Information* **2019**, *10*, 36. [CrossRef]
29. Pang, W.; Hui, K. Interactive Evolutionary 3D Fractal Modeling. *Vis. Comput.* **2010**, *26*, 1467–1483. [CrossRef]