Mandelbrot- and Julia-Like Rendering of Polynomiographs

Krzysztof Gdawiec

Institute of Computer Science, University of Silesia
Będzińska 39, 41-200, Sosnowiec, Poland
kgdawiec@ux2.math.us.edu.pl

Abstract. Polynomiography is a method of visualization of complex polynomial root finding process. One of the applications of polynomiography is generation of aesthetic patterns. In this paper, we present two new algorithms for polynomiograph rendering that allow to obtain new diverse patterns. The algorithms are based on the ideas used to render the well known Mandelbrot and Julia sets. The results obtained with the proposed algorithms can enrich the functionality of the existing polynomiography software.

Keywords: polynomiography, rendering, Julia set, Mandelbrot set, computer art

1 Introduction

One of the most elusive goals in computer aided design is artistic design and pattern generation. Pattern generation involves diverse aspects: analysis, creativity, development [11]. We must deal with all the three aspects in order to obtain an interesting pattern that could be later used in jewellery design, carpet design, as a texture etc.

Many methods of pattern generation exist in the literature, but we will mention only those which are needed in the paper. First such method is polynomiography. The method was introduced by Kalantari and it is based on the root-finding methods of complex polynomials [4]. Other known methods of pattern generation are Mandelbrot and Julia sets [3]. The methods are based on the iteration of a complex function, usually a quadratic function. In this paper, we combine concepts taken from the rendering methods of the Mandelbrot and Julia sets with the polynomiography, obtaining in this way new methods of artistic pattern generation. The patterns obtained with the help of our algorithms could find similar applications as the standard polynomiography, i.e., creating paintings, carpet design, tapestry design, animations etc. [5].

The paper is organized as follows. In Sec. 2 we introduce some basic informations about polynomiography and a standard algorithm for polynomiograph rendering. Then, in Sec. 3 we present two algorithms of polynomiograph rendering that are based on the ideas used in the rendering of Mandelbrot and Julia sets. We present some exemplary polynomiographs obtained with the proposed algorithms in Sec. 4. Finally, in Sec. 5 we give some concluding remarks.

2 Polynomiography

Polynomiography is the art and science of visualization in approximation of the zeros of complex polynomials, via fractal and non-fractal images created using the mathematical convergence properties of iteration functions [4]. A single image created using the mentioned methods is called a polynomiograph.

In the polynomiography we can use different polynomial root finding methods, e.g., Newton method [9], Traub-Ostrowski method [9], Harmonic Mean Newton's method [1], Halley method [1], Whittaker method [9] etc. Because in the literature there is such multiplicity of root finding methods in the paper we will limit to the so-called parametric basic family [6].

Let us consider a polynomial $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ of the form:

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0.$$

To define the parametric basic family we need to introduce a sequence of functions $D_m : \mathbb{C} \to \mathbb{C}$. For $z \in \mathbb{C}$ the D_m function is defined as follows [6]:

$$\begin{aligned} D_0(z) &= 1, \\ D_m(z) &= \det \begin{pmatrix} p'(z) & \frac{p''(z)}{2!} & \ldots & \frac{p^{(m-1)}(z)}{(m-1)!} & \frac{p^{(m)}(z)}{m!} \\ p(z) & p'(z) & \ddots & \ddots & \frac{p^{(m-1)}(z)}{(m-1)!} \\ 0 & p(z) & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{p''(z)}{2!} \\ 0 & 0 & \ldots & p(z) & p'(z) \end{pmatrix} \end{aligned} \tag{1}$$

for $m \geq 1$.

Now, the parametric basic family is a sequence of functions $B_{m,\alpha} : \mathbb{C} \to \mathbb{C}$ for $m = 2, 3, \ldots$ and $\alpha \in \mathbb{C}$ of the following form [6]:

$$\forall_{z \in \mathbb{C}} \quad B_{m,\alpha}(z) = z - \alpha p(z) \frac{D_{m-2}(z)}{D_{m-1}(z)}. \tag{2}$$

When we take $m = 2$ and 3 it turns out that B_2 and B_3 are the parametric Newton method and the parametric Halley method (respectively):

$$B_{2,\alpha}(z) = z - \alpha \frac{p(z)}{p'(z)}, \tag{3}$$

$$B_{3,\alpha}(z) = z - \alpha \frac{2p'(z)p(z)}{2p'(z)^2 - p''(z)p(z)}. \tag{4}$$

To render a single polynomiograph we can use Algorithm 1. In the algorithm we use the so-called iteration colouring, i.e., colour is determined according to the number of iteration in which we have left the while loop. Other colouring methods exist in the literature, e.g., basins of attraction, mixed colouring [6].

Algorithm 1: Polynomiograph rendering

Input: $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – polynomial, $A \subset \mathbb{C}$ – area, k – number of iterations, ε – accuracy, $m \geq 2$ – number for $B_{m,\alpha}$, $\alpha \in \mathbb{C}$ – parameter for $B_{m,\alpha}$, $colours[0..k]$ – colourmap.

Output: Polynomiograph for the area A.

```
for z_0 ∈ A do
    i = 0
    while i ≤ k do
        z_{i+1} = B_{m,α}(z_i)
        if |z_{i+1} − z_i| < ε then
            break
        i = i + 1
    Print z_0 with colours[i] colour
```

3 Algorithms of Mandelbrot- and Julia-Like Rendering of Polynomiographs

When we generate Julia and Mendelbrot sets, similar to the polynomiography, for each point in the area $A \subset \mathbb{C}$ we make some iterative process. For the Mandelbrot set this iterative process is following:

$$z_{i+1} = z_i^2 + c, \tag{5}$$

where constant c is equal to the considered point, and $z_0 = 0$. If z_{i+1} fulfils the escape criteria, i.e., $|z_{i+1}| > 2$, then the point do not belongs to the Mandelbrot set and we draw it with colour corresponding to the number of iteration.

We can transfer the concept with the constant c from the Mandelbrot algorithm to the polynomiography. In the original polynomiography algorithm we replace the standard iteration of $B_{m,\alpha}$ with:

$$z_{i+1} = B_{m,\alpha}(z_i) - c. \tag{6}$$

The constant c, unlike in the Mandelbrot algorithm, is taken as a value of a mapping $f : \mathbb{C} \to \mathbb{C}$ in the considered point. Moreover, at the end of each iteration constant c is transformed with an additional mapping $g : \mathbb{C} \to \mathbb{C}$. Algorithm 2 presents the complete pseudocode of the proposed algorithm.

The iterative process for the Julia sets is the same as for the Mandelbrot set, but this time c is constant for all the points in A, and the starting point is equal to the considered point. Moreover, in the escape criteria we have different threshold value: $\max\{2, |c|\}$.

In the Julia-like version of the rendering algorithm for polynomiography we change the standard iteration process with:

$$z_{i+1} = B_{m,\alpha}(z_i) + c, \tag{7}$$

Algorithm 2: Mandelbrot-like Polynomiograph rendering

Input: $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – polynomial, $A \subset \mathbb{C}$ – area, k – number of iterations, ε – accuracy, $m \geq 2$ – number for $B_{m,\alpha}$, $\alpha \in \mathbb{C}$ – parameter for $B_{m,\alpha}$, $f, g : \mathbb{C} \to \mathbb{C}$ – mappings, $colours[0..k]$ – colourmap.

Output: Polynomiograph for the area A.

```
for z_0 ∈ A do
    c = f(z_0)
    i = 0
    while i ≤ k do
        z_{i+1} = B_{m,α}(z_i) − c
        if |z_{i+1} − z_i| < ε then
            break
        c = g(c)
        i = i + 1
    Print z_0 with colours[i] colour
```

and at the end of each iteration we use mapping $f : \mathbb{C} \to \mathbb{C}$ to transform the constant c. Algorithm 3 presents the complete pseudocode of the proposed algorithm.

Algorithm 3: Julia-like Polynomiograph rendering

Input: $c \in \mathbb{C}$ – parameter, $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – polynomial, $A \subset \mathbb{C}$ – area, k – number of iterations, ε – accuracy, $m \geq 2$ – number for $B_{m,\alpha}$, $\alpha \in \mathbb{C}$ – parameter for $B_{m,\alpha}$, $f : \mathbb{C} \to \mathbb{C}$ – mapping, $colours[0..k]$ – colourmap.

Output: Polynomiograph for the area A.

```
for z_0 ∈ A do
    i = 0
    while i ≤ k do
        z_{i+1} = B_{m,α}(z_i) + c
        if |z_{i+1} − z_i| < ε then
            break
        c = f(c)
        i = i + 1
    Print z_0 with colours[i] colour
```

In both the proposed algorithms we use a standard test for the convergence of the iteration process, i.e., $|z_{i+1} - z_i| < \varepsilon$, but we can use different convergence tests as was proposed in [2] for the standard polynomiography rendering algorithm.

4 Examples

In this section, we present some examples of polynomiographs obtained using algorithms proposed in Sect. 3.

We start with examples of polynomiographs obtained using the Mandelbrot-like rendering algorithm. The polynomiographs are presented in Fig. 1, and the parameters used were following:

(a) $p(z) = z^3 - 1$, $A = [-2, 2]^2$, $k = 40$, $\varepsilon = 0.001$, $m = 2$, $\alpha = 0.75$, $f(z) = 0.1 \sin z + 0.33 \cos z$, $g(z) = \cos z$,
(b) $p(z) = z^3 - 1$, $A = [-2, 2]^2$, $k = 40$, $\varepsilon = 0.001$, $m = 3$, $\alpha = 0.75$, $f(z) = 0.1 \sin z + 0.33 \cos z$, $g(z) = \cos z$,
(c) $p(z) = z^3 - 1$, $A = [-2, 2]^2$, $k = 40$, $\varepsilon = 0.001$, $m = 2$, $\alpha = 0.75$, $f(z) = z$, $g(z) = \log(\cos z)$,
(d) $p(z) = z^4 + 4$, $A = [-2, 2]^2$, $k = 40$, $\varepsilon = 0.001$, $m = 3$, $\alpha = 0.75 - 0.8\mathbf{i}$, $f(z) = 2z$, $g(z) = z$.

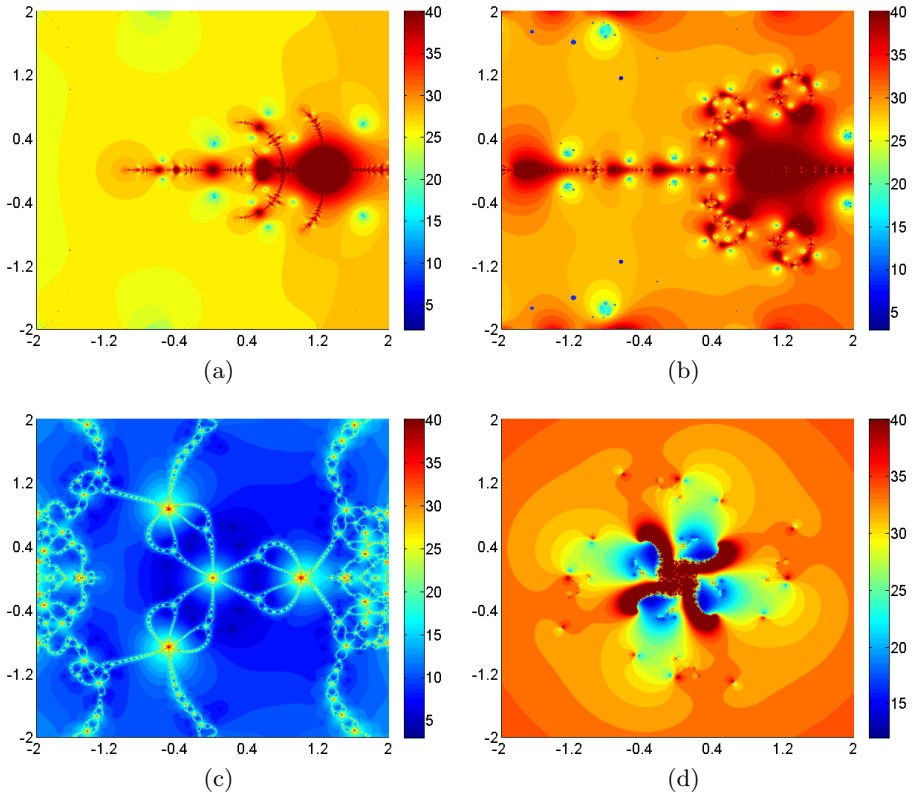

Fig. 1. Examples of Mandelbrot-like rendering of polynomiographs.

Fig. 2 presents some examples of polynomiographs obtained with the Julia-like rendering algorithm. The parameters used to obtain the images were following:

(a) $c = 0.285$, $p(z) = z^3 - 1$, $A = [-3.2, 0.7] \times [-2, 2]$, $k = 25$, $\varepsilon = 0.001$, $m = 2$, $\alpha = 1.0$, $f(z) = \sin(\cos(z))$,
(b) $c = 0.285$, $p(z) = z^3 - 1$, $A = [-3.2, 0.7] \times [-2, 2]$, $k = 25$, $\varepsilon = 0.001$, $m = 3$, $\alpha = 1.0$, $f(z) = \sin(\cos(z))$,
(c) $c = -0.8$, $p(z) = z^4 + 4$, $A = [-2, 2]^2$, $k = 30$, $\varepsilon = 0.001$, $m = 2$, $\alpha = 0.85$, $f(z) = 0.2 \cos z$,
(d) $c = -5.8$, $p(z) = z^4 + 4$, $A = [-2, 2]^2$, $k = 30$, $\varepsilon = 0.001$, $m = 2$, $\alpha = 0.85$, $f(z) = 0.2 \cos z$.

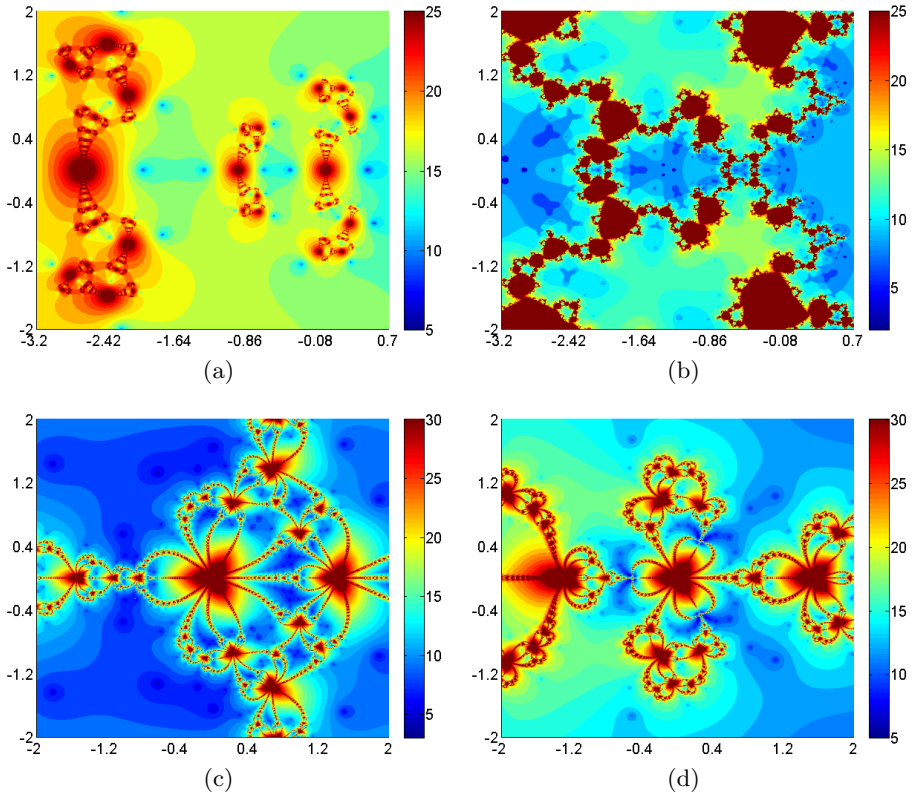

Fig. 2. Examples of Julia-like rendering of polynomiographs.

In the last example (Fig. 3) we show examples of using different convergence tests in the Mandelbrot- and Julia-like rendering algorithms. To see the difference between standard and non-standard convergence tests we used the same

parameters as in Figs. 1(d), 1(a), 2(d), 2(b) (respectively), and the tests were following:

(a) $||z_{i+1}|^2 - |z_i|^2| < \varepsilon$,
(b) $|0.01(z_{i+1} - z_i)| + |0.0285|z_{i+1}|^2 - 0.029|z_i|^2| < \varepsilon$,
(c) $|0.01(z_{i+1} - z_i)| + |0.029|z_{i+1}|^2 - 0.03|z_i|^2| < \varepsilon$,
(d) $|0.08\Re(z_{i+1} - z_i)| < \varepsilon \vee |0.08\Im(z_{i+1} - z_i)| < \varepsilon$,

where $\Re(z)$, $\Im(z)$ denote the real and imaginary part of z (respectively).

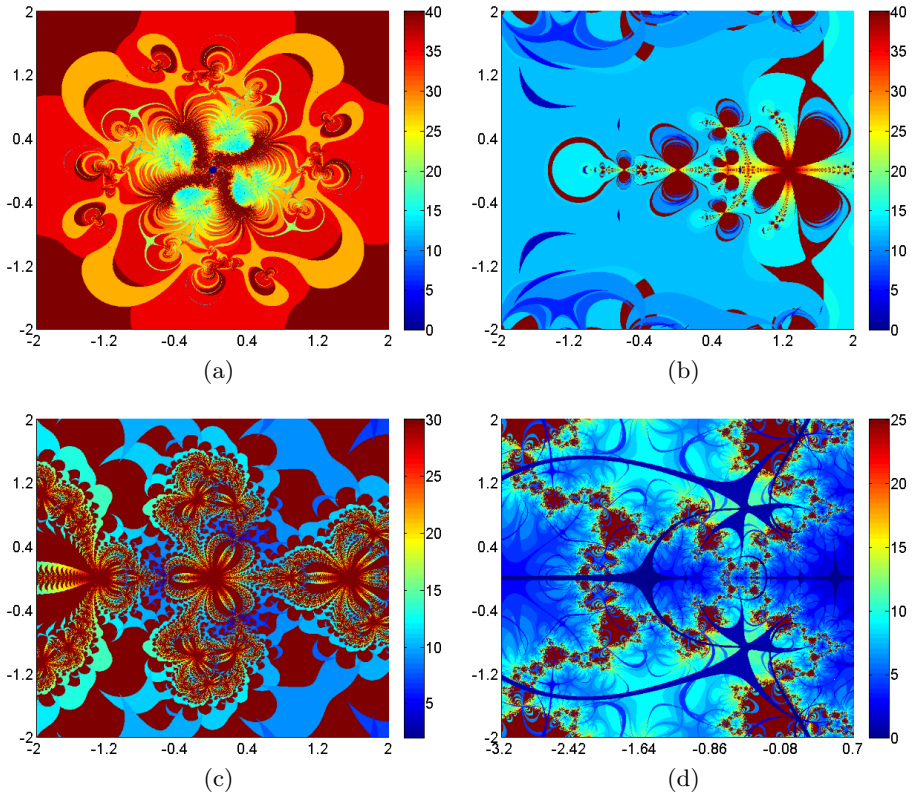

Fig. 3. Examples of Mandelbrot- and Julia-like rendering of polynomiographs with different convergence tests.

5 Conclusions

In this paper, we presented two algorithms for the rendering of polynomiographs that are based on the ideas used in the rendering of the well known Mandelbrot and Julia sets. The presented examples show that using the proposed algorithms

we are able to obtain very interesting and diverse patterns, and that these patterns are different from those obtained with the standard rendering method of the polynomiographs.

Polynomiography is based on the complex polynomials. In the literature we can find methods of obtaining interesting patterns using instead of the complex numbers the q-systems numbers [8] and bicomplex numbers [10]. Moreover, in the standard polynomiography we can use different iteration processes, e.g. Mann, Ishikawa [7]. The use of q-system and bicomplex numbers in the polynomiography together with the different iteration schemes can probably further enrich the obtained patterns, what would be examined in our future work.

References

1. Ardelean, G.: Comparison Between Iterative Methods by Using the Basins of Attraction. Applied Mathematics and Computation 218(1), 88–95 (2011)
2. Gdawiec, K.: Polynomiography and Various Convergence Tests. In: WSCG 2013 Communication Proceedings, pp. 15–20 (2013)
3. Herrmann, D.: Algorithmen für Chaos und Fraktale. Addison-Wesley, Bonn (1994)
4. Kalantari, B.: Polynomiography and Applications in Art, Education and Science. Computers & Graphics 28(3), 417–430 (2004)
5. Kalantari, B.: Two and Three-dimensional Art Inspired by Polynomiography. In: Proceedings of Bridges, Banff, Canada, pp. 321–328 (2005)
6. Kalantari, B.: Polynomial Root-Finding and Polynomiography. World Scientific, Singapore (2009)
7. Kotarski, W., Gdawiec, K., Lisowska, A.: Polynomiography via Ishikawa and Mann Iterations. In: Bebis, G., et al. (eds.) ISVC 2012, Part I. LNCS, vol. 7431, pp. 305–313. Springer, Heidelberg (2012)
8. Levin, M.: Discontinuous and Alternate Q-System Fractals. Computer & Graphics 18(6), 873–884 (1994)
9. Varona, J.L.: Graphics and Numerical Comparison Between Iterative Methods. The Mathematical Intelligencer 24(1), 37–46 (2002)
10. Wang, X.-Y., Song, W.-J.: The Generalized M-J Sets for Bicomplex Numbers. Nonlinear Dynamics 72(1-2), 17–26 (2013)
11. Wannarumon, S., Unnanon, K., Bohez, E.L.J.: Intelligent Computer System for Jewelry Design Support. Computer-Aided Design & Applications 1(1-4), 551–558 (2004)