AA Weaving

Abdalla G. M. Ahmed
abdalla_gafar@hotmail.com

Abstract

A novel algorithmic method for weaving design is presented, inspired by the recently discovered AA Fractals. The method is capable of producing a weaving pattern from any binary sequence or pair of sequences. All generated patterns can be woven in the simplest (4-shaft) dobby looms. An algorithm for generating input binary sequences is also presented, and the sequences produced by this algorithm generate patterns which resemble AA Patterns.

1 Introduction

The fact that woven fabrics are uniform meshes of threads grabs the interest of mathematicians. There have been many attempts for employing mathematical concepts to develop algorithmic weaving design techniques. For example, Ada Dietz in her 1949 monograph [8] described a method for using binomial expansions to write the threading draft. Great efforts were made by Ralph E. Griswold, both in collecting literature of this kind [11], and in introducing new mathematical methods for generating weaving patterns. He suggested methods based on Cellular Automata, L-Systems, Fractals, Chaos, and some other known mathematical patterns. He started collecting his work in a book [10], but he left this world before completing his effort.

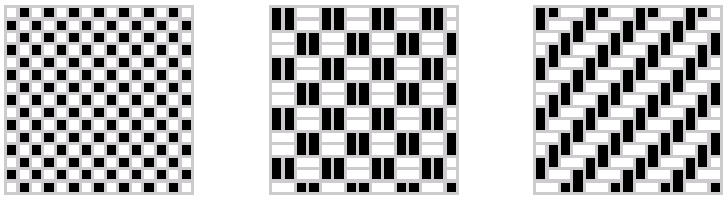
In this paper we introduce a new algorithmic technique for designing weaving patterns, inspired by AA Outlines: a recently discovered fractal form related to AA Patterns [7]. Unlike the fractal sequence methods suggested by Griswold [12], the technique we are going to see works 'natively' in simple looms, without any adaptations, and is still capable of producing rich weaving patterns thanks to the inherent fractal structure of the underlying concept. We start by providing an essential background about weaving in Section 2. In Section 3 we describe AA Outlines, and build on them a new concept, AA Bitmaps, which will serve the basis of the new weaving technique discussed in Section 4. Finally, we give a conclusion in Section 5, along with recommendations for future work.

2 Weaving Essentials

A woven fabric is made of threads running 'along', called 'warps', and threads running 'across' called 'wefts'. At each intersection of threads, looking from one side of the fabric, either the warp or the weft runs above. A device called 'loom' is used to weave the fabric, row by row. The loom has a mechanism to raise all warp threads meant to run above in the current row, lower those which are to stay below, and pass the weft thread across. If a "warp is above" is encoded as '0', and a "weft is above" encoded as '1', then the whole fabric can be abstracted as a monochrome bitmap, which is essentially a 2D matrix with binary entries. All (single) woven fabrics can be seen as monochrome bitmaps, as illustrated in Figure 1.

But are all monochrome bitmaps 'weavable'? Here we face two practical problems:

1. Weaving an arbitrary bitmap means that the loom needs to control each individual warp thread in every row. When this is translated into mechanical/electronic control, it means very sophisticated, and therefore very expensive looms. The loom capable of controlling individual warps is called "Jacquard loom".



(a) (b) (c)

Figure 1 : *Bitmaps of common kinds of woven fabrics: (a) plain weave (b) Oxford weave, and (c) twill weave. The number of distinct rows and columns are 2, 2, and 4, respectively.*

2. A long run of zeros or ones in the bitmap would translate into a thread that 'floats' on the surface for a substantial length without interweaving with the crossing threads. Such a thread is evidently prone to snagging and tearing, and causes some looseness in the fabric. One way to solve this problem is using a technique called 'double weaving', where two sets of warps and wefts are used, as if two fabrics are interwoven together. Once again, this technique is costly, and generates thick fabrics (two layers) which are not suitable for all purposes.

Given the first problem, it is desirable to keep the input bitmaps 'simple'. What do we mean by simple? A good measure of simplicity is the number of distinct rows and columns in the bitmap, as these are directly reflected in the design complexity of looms, and the effort required to setup and operate them. The idea is to control warp threads in groups, rather than individually, and the corresponding loom is called "dobby loom". The number of distinct rows indicates (but is not equal to) the number of groups of warps; or mechanically speaking, the number of 'shafts' in the loom. Obviously, the more shafts we have, the more expensive is the loom, and the more labor intensive is its setup.

So simplicity of weaving bitmaps is desirable from the manufacturer point of view; but visual appeal and richness are also highly desirable from the end user point of view, and between these two ends emerges the art of 'weaving design'. It can be described as the process of finding or creating bitmaps which exhibit visual beauty and/or any other desirable features, while still keeping the number of distinct rows and/or columns to the minimum. This is not a simple job, indeed, and as far as I know there is no currently available deterministic procedure to coordinate these two objectives. Thus, this function remained primarily a matter of art; although there are some helpful software tools.

It is worth mentioning that the traditional weaving design involves the very same bitmap concept we explained here, referred to as a 'draft'; but it also involves two narrow horizontal and vertical bitmaps, and one more small bitmap at the corner, as shown in the examples in Figure 2. The three auxiliary bitmaps represent 'handles' that can be tuned, totally independently, to control the draft; which is just another clue about the complexity of weaving design. There in another design of looms which replaces the treadles and the tie-up with a drum fitted with wedges to select which shafts are lifted in each row; much like a music box. For further details about traditional weaving design the interested reader is referred to the articles by Andrew Glassner [9] as a good starting point. These articles also explain how to develop software tools for weaving design.

3 AA Outlines

"AA Outlines" is a recently discovered fractal form related to AA Patterns [7]. Like most other fractal forms, they are defined in a very simple way [13]; as described in Algorithm 1 in the following page. Depending on the input binary sequences $\{\mathbf{X}, \mathbf{Y}\}$, Algorithm 1 can generate very complex structures. Note that the same binary sequence can be used to label rows and columns.

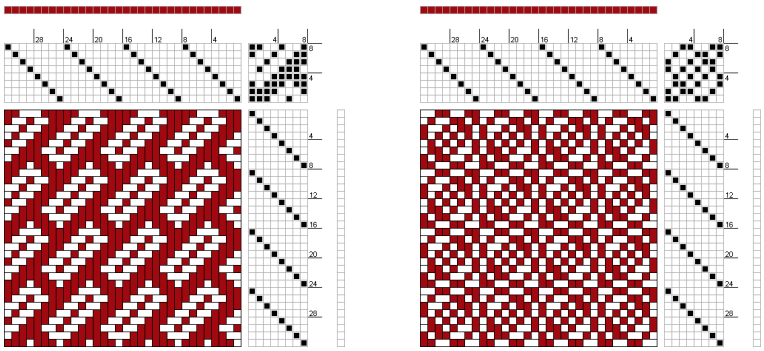
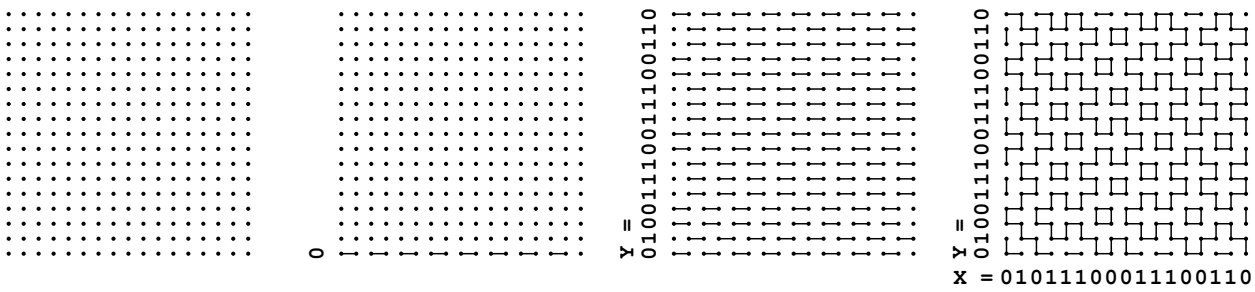


Figure 2 : *Two examples of weaving drafts, along with their threading plans (above), treadling sequences (at the right), and tie-ups (at top-right corner). Retrieved from www.handweaving.net [1, 2]. The gray pixels (red in electronic version of this paper) represent warp threads, which run vertically in the diagram. Each row of the threading plan represents a physical shaft in the loom, and each warp thread is attached to a single shaft, via a so-called heddle, as indicated by the set pixel in the corresponding column of the threading plan. Each shaft is connected to one or more treadles (pedals), and the 'tie-up' indicates which shafts are connected to which treadles. The treadling sequence indicates, for each row, which treadle is pressed to lift the connected shafts. Lifting shafts will raise all threads attached to them, opening a 'shed' for passing the weft thread.*

Algorithm 1 Drawing AA Outlines.

1. Start with a uniform 2D grid of points. See Figure 3(a).
2. On the first row join every other pair of points, creating a dashed line. See Figure 3(b). We label this row 0 if the first and second points are joined and 1 if the second and third points are joined.
3. Use a binary sequence $\mathbf{Y}$ to label each row 0 or 1 and create the appropriate dashed line. See Figure 3(c).
4. Similarly use a binary sequence $\mathbf{X}$ to label each column 0 or 1 and create the appropriate dashed line. See Figure 3(d).



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

Figure 3 : *Steps to build an AA Outline. Numbers indicate whether each dashed line begins with a dash (0) or a gap (1) relative to the edge.*

One important property of AA Outlines is that at every point in the underlying grid there is exactly one dash running vertically and one running horizontally. See Figure 3. Dashes form chains which either surround closed regions, or run until they meet the edges of the pattern. In both cases each chain of dashes splits the pattern area into two distinctly separated regions, which makes the overall outline lend itself to the even-odd style of painting [4], as illustrated in Figure 4. These even-odd paintings are binary by nature, and

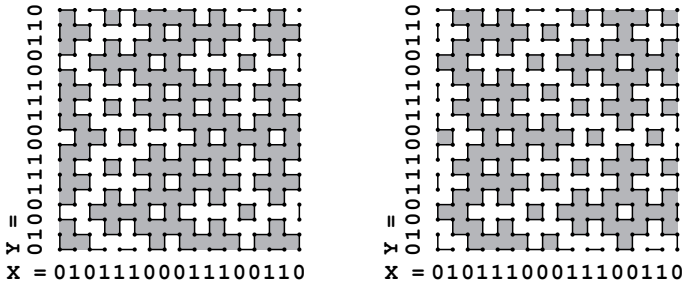


Figure 4 : *The outlines of Figure 3(d) painted in an even-odd fashion to make an AA Bitmap.*

can be represented by monochrome bitmaps, with pixels having their corners at the underlying grid points. We will call such bitmaps AA Bitmaps.

Let us analyze AA Bitmaps to find out how they can be generated algorithmically. The fact that we are dealing with binary values suggests using modulo 2 arithmetics, in which value transitions are easy to detect by addition:

$$0 + 0 = 1 + 1 = 0\,, \tag{1}$$

$$0 + 1 = 1 + 0 = 1\,; \tag{2}$$

so adding similar values yields 0 and adding different values yields 1. Inspecting the first (bottom) row of pixels in Figure 4 reveals that vertical dashes in the painted outline, or 0's in sequence $\mathbf{X}$, mark color transitions in the bitmap, and 1's make runs of pixels of the same color. In modulo-2 this translates into

$$\mathbf{X}[i] = \mathbf{R1}[i-1] + \mathbf{R1}[i] + 1\,, \tag{3}$$

where $\mathbf{R1}$ is a binary vector representing pixels in the first row. But by design the same set of dashes appear in every other row, which means that all odd rows are either copies or negatives of the first row. Similarly, all even rows are copies or negatives of the second row. The whole bitmap is therefore made up of only four distinct rows. Further, note that where there is a dash in odd rows there is always a gap in even rows, and vise-versa; so a color transition in one row matches no-transition in the adjacent rows, and vise-versa. We conclude that the four distinct rows can easily be generated given only one of them. Which of the four reference rows is used in each row of the final bitmap depends only on whether the vertical index of the bitmap row is odd or even, and on the color of the first pixel in that row. By symmetry, all the arguments made about rows apply to columns as well, so

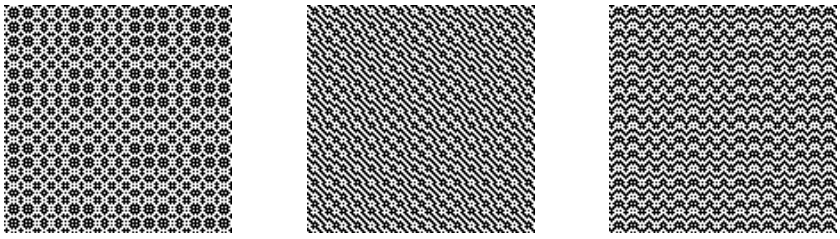
$$\mathbf{Y}[i] = \mathbf{C1}[i-1] + \mathbf{C1}[i] + 1\,, \tag{4}$$

where $\mathbf{C1}$ is a binary vector representing pixels in the first column, and so on.

The foregoing analysis can easily be translated into steps for generating AA Bitmaps, given the first row and and the first column. This time (1) and (2) will be used to apply value transitions besides detecting them. Algorithm 2 has the details, and Figure 5 shows example results. The relations (3) and (4) make it also possible to use $\{\mathbf{X}, \mathbf{Y}\}$ directly to generate AA Bitmaps, as shown in Algorithm 3. The simple copy-and-invert operation in Algorithm 3 makes it possible to work this algorithm by hand. It also gives us new insight in the structure of AA Bitmaps; that on moving from row to row half of pixels are inverted and half of them remain unchanged.

Algorithm 2 Generating an AA Bitmap $\mathbf{M}$ from two binary vectors $\{\mathbf{R1}, \mathbf{C1}\}$ which represent the first row and the first column. Indexes start at 1, and all additions are modulo 2. Note that $\mathbf{R2}$, $\mathbf{R3}$, and $\mathbf{R4}$ are not necessarily the second, third, and fourth row in the bitmap.

```
1. Populate the other three distinct rows:
    Set R2[1] = R1[1]
    Set R3[1] = R4[1] = R1[1] + 1
    For j = 2 to length of R1
        Set R2[j] = R2[j-1] + (R1[j-1] + R1[j]) + 1
        Set R3[j] = R1[j] + 1
        Set R4[j] = R2[j] + 1
2. Populate bitmap M with the appropriate rows:
    For i = 1 to length of C1
        If C1[i] equals C1[1] then
            If i is odd then Set M[i] = R1
            else set M[i] = R2
        else
            If i is odd then Set M[i] = R3
            else set M[i] = R4
```



(a) (b) (c)

Figure 5 : *AA Bitmaps generated by Algorithm 2 taking input from (a) Fibonacci word (sequence A003849 in [3]), (b) Thue-Morse sequence (sequence A010060 in [3]), and (c) Fibonacci word for* $\mathbf{R1}$ *and Thue-Morse for* $\mathbf{C1}$.

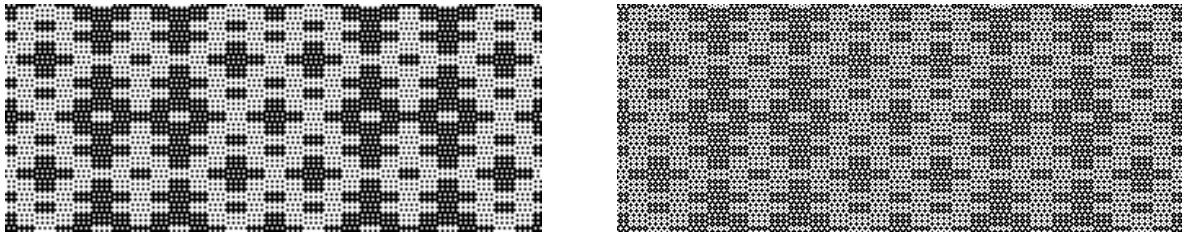
4 AA Weaving

The concept we have seen in the previous section is capable of converting any binary sequence (or pair of sequences) into a monochrome bitmap; hence a weavable pattern. How good are these bitmaps as weaving designs, considering the aspects we talked about in Section 2? Let us first address the simplicity issue. As we mentioned in the previous section, all AA Bitmaps have exactly 4 distinct rows and 4 distinct columns. This makes them about the simplest weavable patterns, save a few common patterns like the ones illustrated in Figure 1. Even better, a specialized loom can be designed to take advantage of the fact that the 4 rows are interrelated, and so are the 4 columns. Such a loom should be easier to setup and to operate; for example, for each row or column the loom or the labor has to choose between only 2 alternatives instead of four, since the even/odd alternation can be preset or automated.

Next we consider the long floats issue, which might appear with some input sequences. the key for solving this problem in AA weaving is to take advantage of the fractal (nested) nature of the underlying AA Outlines, which allows adding more details to the pattern while still maintaining its existing details, as shown in Figure 6. Evidently adding details means changing the size of the pattern and/or the density of threads, but the point is that it makes it possible to weave any arbitrary binary sequence. The details need

Algorithm 3 Generating an AA Bitmap $\mathbf{M}$ from vectors $\{\mathbf{X}, \mathbf{Y}\}$ used to label columns and rows in the underlying AA Outline (see Algorithm 1). Indexes start at 1, and all additions are modulo 2.

```
1. Use X to populate the first row of bitmap M:
    Arbitrarily Set M[1][1] to 0 or 1
    For j = 2 to width of M
        Set M[1][j] = M[1][j-1] + X[j] + 1
2. Use Y to populate consecutive rows of M:
    For i = 2 to height of M
        For j = 1 to width of M
            Set M[i][j] = M[i-1][j] + Y[i] + j
```



(a) (b)

Figure 6 : *An example of adding finer details to solve the long floats issue: (a) a pattern containing 9 threads floats, and (b) A substitute pattern which maintains all existing details, but limit the maximum float to 3 threads. Note that the substitute pattern has to be denser and/or larger.*

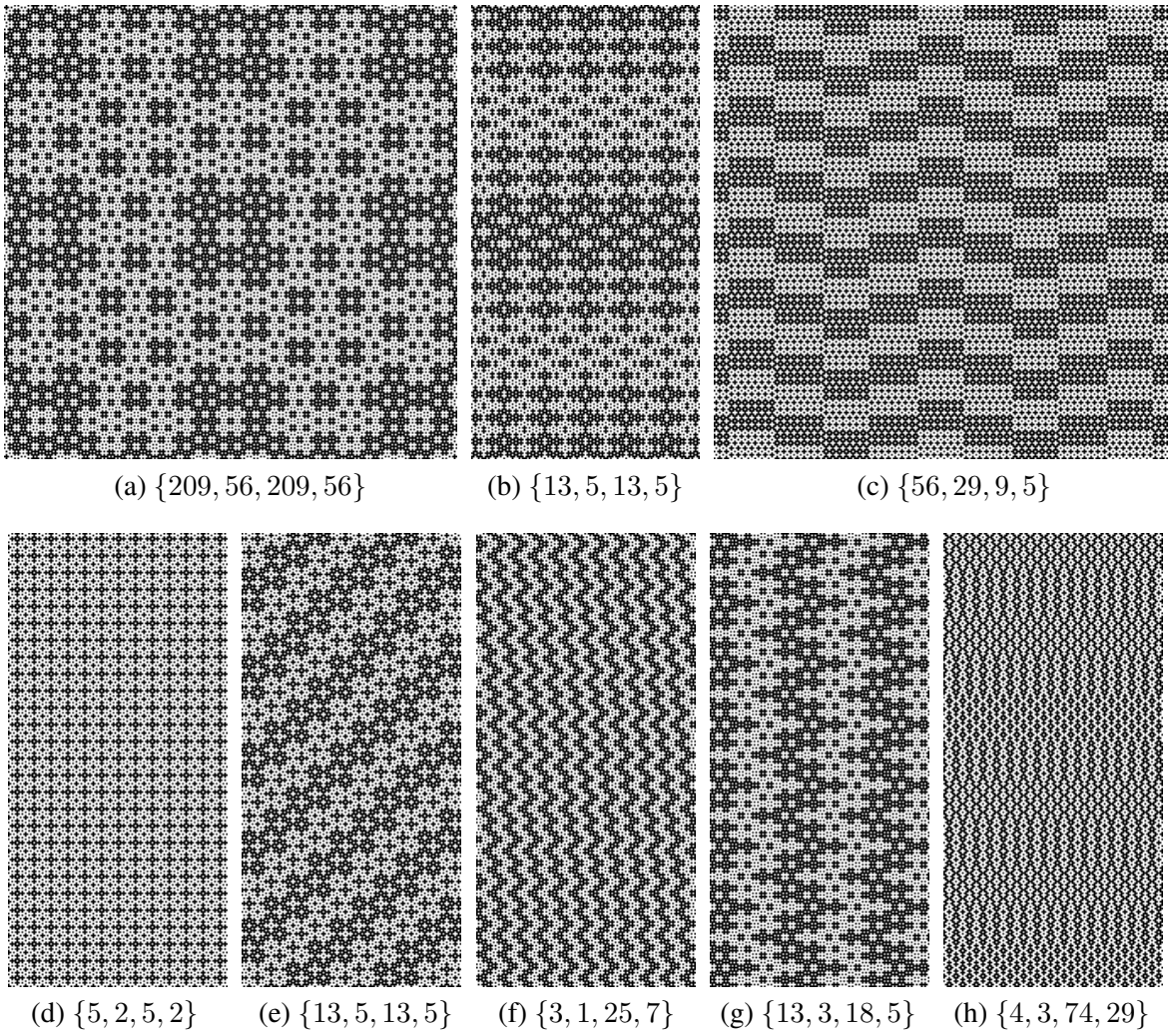
better understanding of AA Fractals [7], which is still a work in progress at the time of this writing.

Moving to the visual appeal issue, it is a fact that any binary sequence can generate an AA bitmap, but as you can expect, not all such bitmaps are visually appealing. The internal structure of the binary sequence is reflected in the generated bitmap, so a random sequence is expected to generate an equally random bitmap.

It would therefore be a good idea to have a way to obtain binary sequences that generate the desirable features in the bitmap. The original AA Patterns now come in handy: Algorithm 4 describes a way to generate binary sequences for AA Bitmaps using 2 parameters $\{q, r\}$. Sequences generated by Algorithm 4, which we will call AA Sequences, are closely related to AA Patterns, and they generate outlines of the original AA Patterns [6] when fed as $\{\mathbf{X}, \mathbf{Y}\}$ into Algorithm 1. Consequently, using AA Sequences as inputs to Algorithm 3 produces AA Bitmaps which closely resemble AA Patterns, and enjoy many of their properties. Specifically, entries of the continued fraction expansion of $(q - r)/r$ directly control levels of details in the generated bitmap in a similar manner to that described in [5] for AA Patterns. Figure 7 shows example AA Bitmaps generated from AA Sequences.

Algorithm 4 Generating an AA Sequence $\mathbf{A}$ from two co-prime integer parameters $\{q, r\}$. Parameters and variables in this algorithm are related to parameters and variables with similar names in [6]. Note that the sequence will be periodic, and its period is $2q$.

```
For x = 1 to required length
    Set dX = ((2q - r) * x) modulo 2q
    if (dX < q) Set A[x] = 0
    else Set A[x] = 1
```



(a) $\{209, 56, 209, 56\}$ (b) $\{13, 5, 13, 5\}$ (c) $\{56, 29, 9, 5\}$

(d) $\{5, 2, 5, 2\}$ (e) $\{13, 5, 13, 5\}$ (f) $\{3, 1, 25, 7\}$ (g) $\{13, 3, 18, 5\}$ (h) $\{4, 3, 74, 29\}$

Figure 7: *Example AA Bitmaps from AA Sequences, with horizontal and vertical parameters* $\{q_h, r_h, q_v, r_v\}$ *written beneath each sample. Note the nested structure in (a), the diagonal twill style in (e), the slow slope in (f), the herringbone style in (g), the illusion of bent lines in (c), and the illusion of curved lines in (h).*

5 Conclusion

In this paper we introduced AA Weaving, a new algorithmic method for weaving design, capable of generating a weaving pattern (AA Bitmap) from any binary sequence or pair of sequences! We have also supplemented the technique with an algorithm for generating input binary sequences (AA Sequences) from any pair of co-prime integer parameters. We consider the findings of this paper a great breakthrough in weaving design, because AA Bitmaps from AA Sequences are easy to generate, easy to tune, easy to weave, and yet they are visually rich and appealing thanks to their fractal (nested) structure. Typical visually rich weaving patterns used to have complex bitmaps, as can be seen, for example, in the thousands of patterns available at `http://www.handweaving.net` (as of Apr. 21, 2013). When it comes to AA Bitmaps, however, we do not ask about complexity at all, as it is always 4 distinct rows by 4 distinct columns, no matter how rich the pattern is. To summarize, AA Bitmaps represent a fractal form dedicated to weaving.

For future work we recommend (1) studying the properties of AA Patterns and AA Fractals to build

a guide for selecting the appropriate parameters to work with, (2) searching for other families of binary sequences to be used in AA Weaving, (3) developing a weaving design software tool based on the methods described in this paper, or integrating these methods into existing tools, and (4) designing specialized looms which can implement these methods mechanically. The author has already moved a step into (3) and built a small Java application, supplied with the electronic version of this paper, to demonstrate AA Bitmaps from AA Sequences.

Acknowledgments. The author thanks Mr Awad Fathelrahman, Mr El Na'eem Yousif, and Mr Ibrahim El Baloula for their cooperation in producing woven samples of some of the patterns in this paper.

References

[1] Atlas de 4000 Armures, Louis Serrure, France (Arahne) Draft #35168. Available online at `http://www.handweaving.net/PatternDisplay.aspx?PATTERNID=35168`, as of Apr. 3, 2013.

[2] Baldwin's Textile Designer Vol. 1 No. 7, Brasher Falls, New York, U.S.A. Draft #53946, July 1888. Available online at `http://www.handweaving.net/PatternDisplay.aspx?PATTERNID=53946`, as of Apr. 3, 2013.

[3] The On-Line Encyclopedia of Integer Sequences, 2010. published electronically at `http://oeis.org`, as of Apr. 3, 2013.

[4] CORPORATE Adobe Systems Inc. *PostScript language reference (3rd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[5] Abdalla G. M. Ahmed. Mathematical Hints for Parameter Selection for AA Patterns. In *Bridges Coimbra*, pages 271–278, Coimbra, Portugal, July 2011. The Bridges Organization.

[6] Abdalla G. M. Ahmed. Pixel Patterns from Quantization Artifacts of Forward Affine Mapping. *Journal of Graphics, GPU, and Game Tools*, 15(2):73–94, 2011.

[7] Abdalla G. M. Ahmed. On the Fractal Behaviour of AA Patterns. In Hamish Carr and Silvester Czanner, editor, *Theory and Practice of Computer Graphics*, pages 93–97, Rutherford, United Kingdom, September 2012. Eurographics Association.

[8] Ada K. Dietz. *Algebraic Expressions in Handwoven Textiles*. Little Loomhouse, 1949.

[9] A.S. Glassner. *Morphs, mallards & montages: computer-aided imagination*. Ak Peters Series. A K Peters, 2004.

[10] Ralph E. Griswold. *Mathematical and Computational Topics in Weaving*. Available online at: `http://www.cs.arizona.edu/patterns/weaving/webdocs/mo.pdf`, as of Apr. 3, 2013.

[11] Ralph E. Griswold. On-Line Digital Archive of Documents on Weaving and Related Topics. `http://www.cs.arizona.edu/patterns/weaving/webdocs.html`, as of Apr. 3, 2013.

[12] Ralph E. Griswold. Designing with Fractal Sequences. 2004. Available online at: `http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_fctl.pdf`, as of Apr. 3, 2013.

[13] Kenneth Falconer. *Fractal Geometry - Mathematical Foundations and Applications*. Wiley, 2 edition, 2003.