Organized Order in Ornamentation - Supplemental Material

Lena Gieseke
Film University Babelsberg Konrad Wolf

Paul Asente
Adobe Research

Jingwan Lu
Adobe Research

Martin Fuchs
Stuttgart Media University

ACM Reference format:

Lena Gieseke, Paul Asente, Jingwan Lu, and Martin Fuchs. 2017. Organized Order in Ornamentation - Supplemental Material. In *Proceedings of CAe'17, Los Angeles, CA, USA, July 28-29, 2017,* 3 pages.

DOI: 10.1145/3092912.3092913

1 PLACEMENT STRATEGY

To support a wide variety of placement strategies, we use a general "placement function", $p : \mathbb{R}^2 \to \mathbb{R}$, which takes higher values at preferred placement locations.

p is updated after every element placement so that its values decrease. Once $\max\{p\}$ reaches 0 or falls below, we are done.

1.1 Notation

We formulate algebraic expressions of individual functions representing user input and the current system state with the usual mathematical operations on functions, with the following additions:

For functions $f_a, f_b : \mathbb{R}^2 \to \mathbb{R}$, $f_c : \mathbb{R}^2 \to \mathbb{R}^2$, we notate

$$f_a \odot f_b(x, y) := f_a(x, y) \cdot f_b(x, y) \tag{1}$$

for the point-wise multiplication, and

$$f_a \circ f_c := (f_a \circ f_c)(x, y) = f_a(f_c(x, y)) \tag{2}$$

for concatenation. For a set A of functions of the same type, we construct the point-wise minimum as

$$\operatorname{Min} A := \min_{f \in A} f(x, y). \tag{3}$$

The function $E : (\mathbb{R}^2 \to \mathbb{R}) \to \mathbb{R}, \quad f_a \mapsto E(f_a)$ denotes the Euclidean distance transform of f_a, that is, the function that maps every coordinate (x, y) in the plane to its minimal distance to the zero set of f_a. As examples of this notation, the characteristic function of all pre-placed element locations within the stencil is $I_s \odot c$, and the distance map to the union of all placed elements and the stencil border is $\operatorname{Min}\{E(1-g), E(I_s)\}$ (see the following section for the definitions).

1.2 Components

We construct p out of the following building blocks:

User-specified input: the artists choose a **binary stencil** $I_s : \mathbb{R}^2 \to \{0, 1\}$ that defines the area the ornament will occupy, with permissable areas marked with 1.

They may provide a **gray-level image** I_e, which can be used to control aspects like an element's size.

With **scalar constants** (greek letters in the formulae), they can control trade-offs, for example between equidistant and symmetric distributions.

Desired **symmetries** are specified using a symmetry group T of planar isomorphisms $t : \mathbb{R}^2 \to \mathbb{R}^2$, which, for each point in the plane, map to the location of a corresponding point according to the symmetry (e.g. reflection across a straight line or rotation). The artists can optionally provide a map I_a of preferred locations that emphasize the symmetry (e.g. a symmetry axis).

Internal data structures: we store the locations of placed elements in two ways:

- as the characteristic function of their centers of mass $c : \mathbb{R}^2 \to \{0, 1\}$ (that is, $c(x, y) = 1$ iff an element was placed there) and
- a map $g : \mathbb{R}^2 \to \{0, 1\}$ that keeps rasterized geometric proxies for the element. In order to be robust to resampling, c is in practice implemented with 3×3-sized splats.

The distinction between c and g is necessary, as c can be used to drive symmetric placement of elements of different sizes and shapes, while g drives the balanced distribution of arbitrarily-shaped elements.

In the next sections, we will describe how to combine these building blocks to express different design goals, e.g. an equilibrium between desired symmetry and required density.

1.3 Balanced Element Distribution Inside the Stencil

Our placement function p supports the simplest geometric constraint of filling an artist-defined space as defined by a stencil I_s.

Wong et al. show that a balanced element distribution can be obtained by maximizing the distances between elements and between elements and the stencil border. We formulate this as

$$\begin{aligned} p_{\text{dist}} := &\operatorname{Min}\{E(I_s), E(1-c)\} \\ &\odot \operatorname{Min}\{1, E(1-g) - \delta_{\min}\} \end{aligned} \tag{4}$$

A placement is considered optimal if it maximizes the distance to the union of the stencil and previously placed element locations, provided we stay a minimal distance of $\delta_{\min}$ pixels away from already-placed elements. As a result of the greedy search for maximal spaces, larger elements are placed before smaller elements. This creates the visible order hierarchy that is characteristic for many ornamental styles, but fails to produce symmetry consistently in the presence of pre-placed elements.

1.4 Creating Symmetry

A placement is considered symmetric if a placed element can be mapped to itself by an isomorphism t belonging to a nontrivial symmetry group T, e.g., a geometric transformation such as reflection across an axis, or rotation by a rational fraction of the unit circle. For instance, the locations of elements are symmetric with respect to t if $c = c \circ t$, i.e. $c(x, y) = c(t(x, y)), \forall\ (x, y) \in \mathbb{R}^2$.

To approximate that goal, we define a placement function p_{sym} that implements a set of heuristics,

$$\begin{aligned} p_{\text{sym}} := (&\alpha_{\text{sym}} \cdot \sum_{t \in T} c \circ t \\ &+ \operatorname{Min}_{t \in T} \{E(I_s), E(1 - c \circ t)\} \\ &+ \beta_{\text{sym}} \cdot I_a) \\ &\odot \operatorname{Min}\{1, E(1-g) - \delta_{\min}\} \odot I_s \end{aligned} \tag{5}$$

The heuristics work as follows, row by row of Equation 5:

(1) If an element was placed at some location (x, y), we need to preferentially place new elements in locations as dictated by the symmetry operation (for instance, at reflected locations) – in more formal terms, fill its orbit under the symmetry group. Hence, p_{sym} should have the highest values at transformed locations of previously placed elements, motivating the expression in the first line $\alpha_{\text{sym}} \cdot \sum_{t \in T} c \circ t$. The parameter α_{sym} is chosen so that it dominates the other terms. In all our results, $\alpha_{\text{sym}} = 500$.

(2) If placement at such a location is not possible (for instance, because the set of placed elements is already closed under t), we have some freedom. In addition to keeping a maximal distance to the stencil, we prefer placing the next element at a location that permits future symmetric placements in an optimal way, i.e., we want maximal distance not only to existing element proxies, but to the set union of their transformation under t. This is achieved with the summand $\operatorname{Min}_{t \in T}\{E(I_s), E(1 - c \circ t)\}$.

(3) To deal with cases where the latter summand has many equally-high maxima, we optionally add a tie-breaker $\beta_{\text{sym}} \cdot I_a$. In the axis reflection results shown (Figure 5), we have set β_{sym} to 10 on the symmetry axes of T, in the other results, we have set it to 0.

(4) Finally, we exclude placements that are too close to existing values or outside the stencil, so we multiply by $\operatorname{Min}\{1, E(1 - g) - \delta_{\min}\} \odot I_s$.

For symmetry patterns that involve several simultaneous transformations (e.g. four-way symmetry with two axes of reflection), we construct T as the union of the respective symmetry groups.

1.5 Controlling Placements with Image Data

In order to control the placement with an artist-specified example image I_e, we specify control parameters α_{raster} and β_{raster}, so that we can optimize the placement according to

$$\begin{aligned} p_{\text{raster}} := \operatorname{Min}\{&\alpha_{\text{raster}} \cdot I_e - \beta_{\text{raster}}, \\ &E(1-g), E(I_s)\} \end{aligned} \tag{6}$$

We start placing larger elements at places in the input bitmap with high intensity values. We then fill the rest of the target region with smaller elements. The affine transformation of the values of the input image by α_{raster} and β_{raster} controls the scale of the largest placed element and the cut-off value for placement size, leaving the black parts of the example empty. In order to fill the space more densely, the placement function uses the distance to the actually placed elements $E(1 - g)$, instead of their centers $E(1 - c)$.

2 ASYMPTOTIC PLACEMENT PERFORMANCE

Let n be the number of pixels in the stencil we need to fill, m the number of placed elements. In the worst case, we use a placement function that requests elements to be placed in scanline order, and we need to update $O(n \cdot m)$ entries to fill the pattern. However, the average case is much more amenable. We will show this for the two most expensive data structures, the Euclidean distance maps and the maximum pyramid P.

Consider the Euclidean distance: while inserting the i-th element, we need to update the pixels in the Voronoi cell around it, which, on average, covers n/i pixels. The total cost of updating the distance maps for inserting all elements then is

$$O\left(\sum_i \frac{n}{i}\right) = O\left(n \sum_i \frac{1}{i}\right) = O(n \cdot \log n) \tag{7}$$

Now consider the maximum pyramid P. In a single step, updating an area covering a pixels on the lowest region incurs update costs on $\log n$ layers, in total

$$O\left(\sum_{k=0}^{\log n} \frac{a}{4^k}\right) = O\,(a + \log n) \tag{8}$$

In the worst possible pattern, we fill the entire plane with single pixel-sized elements, so $a = 1$ in each of $m = n$ steps, and we incur costs of $O(n \log n)$ (for higher values of a, $m \cdot a$ cannot exceed n, so this is still the worst case).

Next, we will put both costs together. We can expect to update, on average, $O(\frac{n}{i})$ pixels, so every step costs $O(\frac{n}{i} + \log n)$. In total, this causes costs of

$$\begin{aligned} & O\left(\sum_{i=1}^{m}\left(\frac{n}{i} + \log n\right)\right) \\ =& O\left(n \cdot \sum_{i=1}^{m} \frac{1}{i} + m \log n\right) \end{aligned} \tag{9}$$

In the worst case, again, every pixel is filled with a single element, $m = n$, and we obtain total costs of $O(n \log n)$.