Brushing Element Fields

Chen-Yuan Hsu
Bournemouth University

Li-Yi Wei
Adobe Research

Lihua You
Bournemouth University

Jian Jun Zhang
Bournemouth University

(a) 2D graphic design (b) surface artistic collage (c) volumetric aggregate modeling

Figure 1: *Oriented element synthesis*. Our brushing system can be applied for different domains such as 2D planes (a), 3D surfaces (b), and 3D volumes (c); diverse elements such as rigid (a) and deformable (c); and various applications such as design (a), collage (b) and modeling (c). Users can mix a variety of elements with distinct shapes (a) or types (rigid and deformable) (b) from the interface. The system can automatically complete the remaining work based on the partially user-specified strokes (a) (inset), enable the users to interactively author the elements (b) or directly synthesize the elements along a given field (c).

ABSTRACT

Aggregate elements following certain directions have a variety of applications in graphics, design, and visualization. However, authoring oriented elements in various output domains, especially in 3D, remains challenging. We propose a novel brushing system to facilitate interactive authoring of aggregate elements with diverse properties over given output domains via an element synthesis approach. To increase output quality and reduce input workload, we further propose *element fields* that can automatically orient the entire elements in better alignments over the output domains according to partially user-specified strokes. The proposed system can effectively synthesize distinct types of elements within various output domains in higher quality and efficiency and offer more user friendliness than existing practices. Our method can be applied to practical applications such as graphic design, artistic collage, and aggregate modeling.

SA '18 Technical Briefs , December 4–7, 2018, Tokyo, Japan

ACM ISBN 978-1-4503-6062-3/18/12…$15.00
https://doi.org/10.1145/3283254.3283274

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**;
• **Computing methodologies** → **Texturing**;

KEYWORDS

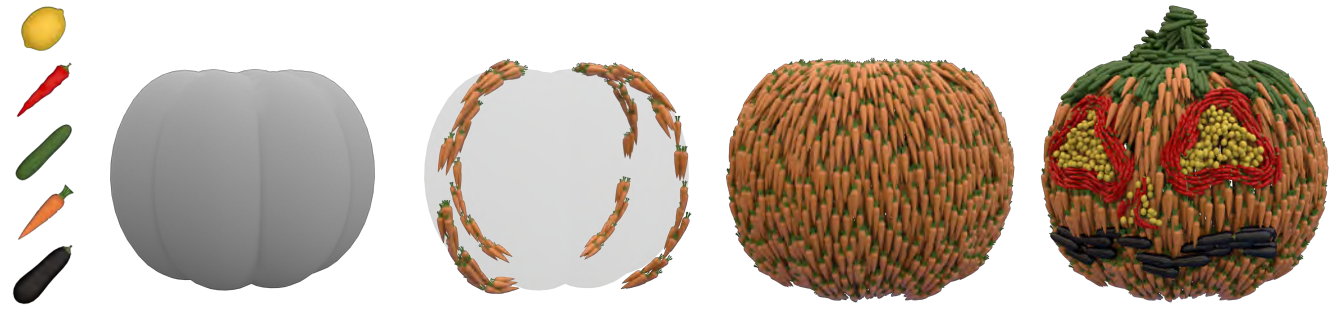
element, field, synthesis, packing, anisotropy, interface, modeling

ACM Reference format:
Chen-Yuan Hsu, Li-Yi Wei, Lihua You, and Jian Jun Zhang. 2018. Brushing Element Fields. In *Proceedings of SIGGRAPH Asia 2018 Technical Briefs , Tokyo, Japan, December 4–7, 2018 (SA '18 Technical Briefs)*, 4 pages.
https://doi.org/10.1145/3283254.3283274

1 INTRODUCTION

Aggregate elements following directions are ubiquitous in natural and man-made objects. Authoring such elements remains challenging, especially in 3D [Cho et al. 2007]. There exist methods for interactively authoring 2D elements (e.g. [Kazi et al. 2012]) or synthesizing 3D elements via batch processing (e.g. [Ma et al. 2011]), but there is a lack of friendly user interfaces for interactive authoring of 3D elements with general shapes, distributions and alignments [Roveri et al. 2015].

Interactive brushing interfaces have been widely adopted for digital painting and pattern authoring [Lu et al. 2014; Zhou et al. 2014]. We propose a novel brushing system for interactive authoring of distinct types of aggregate elements across different output



(a) inputs (b) output domain (c) user-specified strokes (d) autocomplete (e) final arrangement

Figure 2: *The workflow in our brushing system.* Users select one or several input exemplars (a), brush the selected elements over the output domain (b) and interactively see the brushing results (c). To reduce the users' workload, our system can automatically generate the remaining outputs (d) according to the partially user-specified strokes (c). The users can further arrange the elements for the final outcome (e) via corresponding brush operations.

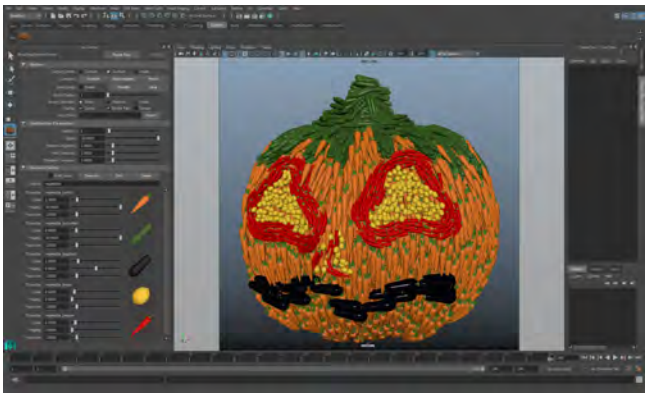


Figure 3: *User interface.* Our system is directly integrated into Autodesk Maya. The control panel is on the left, and the main canvas is in the middle. Users can choose input exemplars from the element palette, perform corresponding operations, and tune relative parameters from the control panel to create desired works in different output domains.

domains (Figure 1). The basic workflow of using the brushing system can be seen in Figure 2. Through the interface (Figure 3), users can select and even mix elements from an element palette of input exemplars. Analogous to common painting interfaces, the element palette enables the users to save multiple selected exemplars as a new input exemplar and then reuse it in an efficient way.

Moreover, anisotropic elements often require proper direction fields for proper alignments, and it can require significant expertise or labor for users to fully specify these fields. To enhance output quality and reduce input workload, our main idea is *element fields* that can automatically complete all the element alignments in better quality based on partially user-specified inputs like manual strokes.

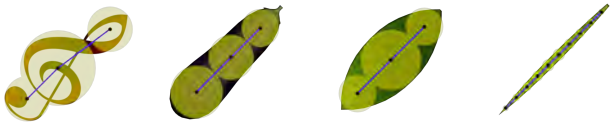
Our system can handle general element shapes, distributions and alignments, has a friendly user interface, and can produce outputs with high quality that match the user specifications and domain shapes. Unlike prior example-based approaches, which the outputs are limited by the inputs, the developed system, centered on the idea of element fields, can effectively solve a complex authoring issue and enable users to create compelling artwork like Figure 1b without requiring significant artistic expertise or manual labor.

2 ELEMENT REPRESENTATION

We extend the sample-based representation in [Ma et al. 2011], where each element e is represented as a set of samples $\{s_1, \ldots, s_n\}$. However, only using samples is not sufficient to properly depict distinct elements such as deformable or anisotropic elements. Thus, we devise a framework that different types of elements with various properties such as scale and rigidity can be better characterized.

2.1 Sampling

Since we represent elements via samples, the number and locations of these samples are important. Ideally, we would like to use as few samples as possible while representing the elements as accurately as possible. As illustrated in Figure 4, instead of using unweighted samples in [Ma et al. 2011], we adopt weighted samples to better characterize diverse elements with fewer samples which can be either overlapping or nonoverlapping depending on the requirement of synthesis quality and performance.



(a) treble clef (b) eggplant (c) leaf (d) grass

Figure 4: *Element representation.* The black dots indicate the sample positions. The yellow circles represent the relative sample weights. The blue lines denote the element graphs.

2.2 Element graph

In Figure 4, in order to well tackle different types of elements, we further employ a graph structure G, inspired by Sumner et al. [2007], to indicate the connectivity among samples and coherently deal with the relationship between the element shape and samples. While Ma et al. [2011] incorporate an extra physics solver with their algorithm to improve synthesis quality, by applying this graph structure in our algorithm, we can efficiently handle a variety of elements without any physics solvers.

In our current implementation, we manually sample the elements and connect the samples into graphs.

3 ELEMENT SYNTHESIS

Given an input exemplar I consisting of a collection of elements $\{e_1, \ldots, e_l\}$ along with their overall distributions, an output domain D with desired size and shape, and an optional direction field O over D, our goal is to compute an output X composed of $\{e_1, \ldots, e_l\}$ that have similar distribution to I and orientation to O.

3.1 Distribution similarity

As in [Ma et al. 2011], to determine whether an output X has similar element distribution with the input exemplar I, we can measure their distribution similarity via sample neighborhoods. Each neighborhood $\mathbf{N}$ centered at s with differences of all neighboring samples $\mathbf{N}_s = \{s'_1, \ldots, s'_n\}$ is defined as $\mathbf{N}(s) = \{\hat{\mathbf{p}}(s'_1, s), \ldots, \hat{\mathbf{p}}(s'_n, s)\}$, where $\hat{\mathbf{p}}$ indicates the displacement between s' and s. For an output sample s_o and an input sample s_i, their neighborhood distance $d(\mathbf{N}(s_o), \mathbf{N}(s_i))$ can be computed as:

$$\begin{aligned} d(\mathbf{N}(s_o), \mathbf{N}(s_i)) &= \sum_{s'_o \in \mathbf{N}_{s_o}} \left| \hat{\mathbf{p}}(s'_o, s_o) - \mathbf{w}_{scale} \mathbf{o}(s_o) \hat{\mathbf{p}}(s'_i, s_i) \right|^2, \\ \mathbf{w}_{scale}(s_i, s'_i, s_o, s'_o) &= \frac{\mathbf{c}(s'_o)\mathbf{w}_s(s'_o) + \mathbf{c}(s_o)\mathbf{w}_s(s_o)}{\mathbf{w}_s(s'_i) + \mathbf{w}_s(s_i)}, \end{aligned} \tag{1}$$

where $\mathbf{o}$, $\mathbf{c}$ and $\mathbf{w}_s$ denote the sample's local orientation, scale and weight respectively. Furthermore, for the output X composed of elements from multiple input exemplars $\{I_1, \ldots, I_m\}$, the distance between $\mathbf{N}(s_o)$ and $\mathbf{N}(s_i)$ for each pair of s_o and s_i should be properly computed according to the corresponding input exemplar $I_{\kappa(s_o)}$, where $\kappa(s_o) \in \{1, \ldots, m\}$, and $I_{\kappa(s_o)}$ means the exemplar where s_o originates from. We devise the overall distribution similarity E_d as:

$$E_d(X, \{I_1, \ldots, I_m\}) = \sum_{s_o \in X} \min_{s_i \in I_{\kappa(s_o)}} d(\mathbf{N}(s_o), \mathbf{N}(s_i)). \tag{2}$$

3.2 Conflict check

Mixtures consisting of elements with extremely distinct sizes and shapes (such as Figure 1a) might not have the relevant distribution information captured in the input exemplars, so we apply a conflict check term to avoid conflicts between samples by checking the distances between these weighted samples. Since the distances between two samples should not be less than the sum of their weights (which act like bounding spheres as illustrated in Figure 4), we can formulate the conflict check E_k as:

$$\begin{aligned} E_k(X) &= \sum_{s_o \in X} \sum_{s'_o \in \mathbf{N}_{s_o}} \mathbf{w}_k(s_o, s'_o) \left| \hat{\mathbf{p}}(s'_o, s_o) - \mathbf{w}_{check} \check{\mathbf{p}}(s'_o, s_o) \right|^2, \\ \mathbf{w}_k(s_o, s'_o) &= \begin{cases} 1 & \text{if } \mathbf{w}_{check} > 1 \text{ and } s_o \in e, s'_o \in e', e \neq e' \\ 0 & \text{if } \mathbf{w}_{check} \leq 1 \text{ or } s_o, s'_o \in e \end{cases}, \\ \mathbf{w}_{check}(s_o, s'_o) &= \frac{\mathbf{c}(s'_o)\mathbf{w}_s(s'_o) + \mathbf{c}(s_o)\mathbf{w}_s(s_o)}{|\check{\mathbf{p}}(s'_o, s_o)|}, \end{aligned} \tag{3}$$

where $\hat{\mathbf{p}}$ is the same as in Equation (1), and $\check{\mathbf{p}}$ represents the current displacement between s'_o and s_o treated as a constant quantity for optimization.

3.3 Graph similarity

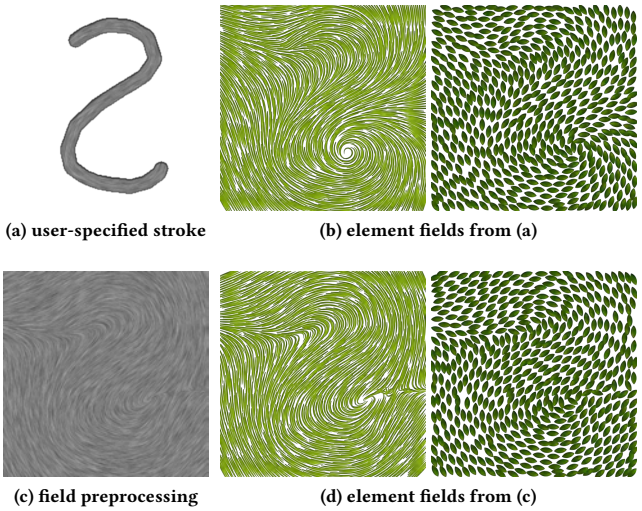
Besides the global distribution similarity, the local graph similarity is considered for those elements consisting of multiple samples to well maintain the element shapes based on their element graphs. Analogous to the distribution similarity, we can minimize the distance between the current graph $\mathbf{G}(s_o)$ and the original graph $\mathbf{G}'(s_o)$ for each output sample s_o to optimize the graph similarity E_g as:

$$\begin{aligned} E_g(X) &= \sum_{s_o \in X} d(\mathbf{G}(s_o), \mathbf{G}'(s_o)), \\ d(\mathbf{G}(s_o), \mathbf{G}'(s_o)) &= \sum_{s'_o \in \mathbf{G}_{s_o}} \left| \hat{\mathbf{p}}(s'_o, s_o) - \mathbf{c}(s_o)\mathbf{o}(s_o)\hat{\mathbf{p}}'(s'_o, s_o) \right|^2, \end{aligned} \tag{4}$$

where $\mathbf{G}_{s_o}$ denotes a set of all connected samples of s_o, and $\hat{\mathbf{p}}'$ represents the displacement between s'_o and s_o in the original graph.

4 ELEMENT FIELDS

Prior methods usually place anisotropic elements via a two-phase process: produce a full direction field, upon which the elements are placed to follow. The first step might be unnecessary and the second might be suboptimal. For better synthesis quality and user efficiency, our one-step process can directly construct element fields following the partially or fully specified input as shown in Figure 5.



(a) user-specified stroke (b) element fields from (a)

(c) field preprocessing (d) element fields from (c)

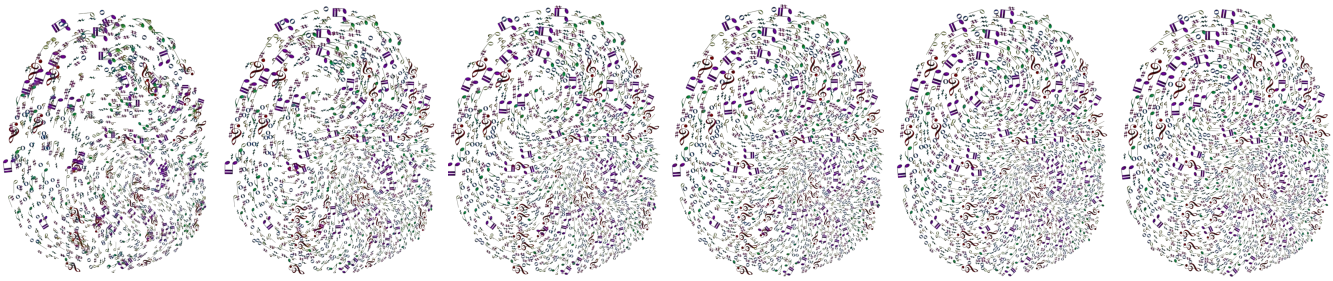
Figure 5: *Element field comparison.* Our method can directly compute element fields (b) from a partially specified input (a) and adaptively fit different elements to the field, with better quality than (d) preprocessing a full input (c) from (a) first via Laplacian interpolation.

4.1 Field alignment

Since each sample $s \in e$ is associated with a local orientation $\mathbf{o}$, the orientations of samples in the specified domain should be properly aligned with the given domain field O. Thus, the distance between $\mathbf{o}(s)$ and $O(\mathbf{p})$ for each sample s in the specified areas should be minimized for the field alignment E_a as:

$$E_a(\mathbf{o}, O) = \sum_{s \in X} \exp^{-\frac{d^2(s,\mathbf{p})}{\sigma^2}} d(\mathbf{o}(s), O(\mathbf{p})), \tag{5}$$

where $O(\mathbf{p})$ indicates the field direction at the point $\mathbf{p}$ which is the closest domain point to the sample s.



(a) initialization (b) iteration 2 (c) iteration 4 (d) iteration 8 (e) iteration 24 (f) iteration 48

Figure 6: *Iteration process.* At the beginning, individual elements from different input exemplars are randomly placed into the output domain and initially aligned along the partially user-specified strokes (Figure 1a). Then our solver iteratively optimizes the position and orientation of each element as a one-step automatic optimization of element distributions and alignments.

4.2 Field continuity

If the field O is partially specified, a field continuity term is conducted to not only automatically construct the orientations of samples within the unspecified regions but also smoothly fit all the samples with their nearby samples. We can minimize the distances between the orientations of samples and their nearby samples to optimize this field continuity E_c as:

$$E_c(\mathbf{o}) = \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \exp^{-\frac{d^2(s,s')}{\sigma^2}} d\left(\mathbf{o}(s), \mathbf{o}(s')\right). \quad (6)$$

4.3 Element rigidity

If the element e is rigid, all samples $s \in e$ should have identical local orientations $\mathbf{o}(s) = \mathbf{o}(s')$, $\forall s, s' \in e$. If e is deformable, all samples $s \in e$ might have different local orientations. Hence, to distinguish between rigid and deformable elements, the element rigidity E_r can be formulated as:

$$E_r(\mathbf{o}) = \sum_{s \in X} \sum_{s' \in \mathbf{G}_s} \mathbf{w}_r(e) d\left(\mathbf{o}(s), \mathbf{o}(s')\right), \quad (7)$$

where $\mathbf{w}_r(e)$ represents the rigidity of elements.

5 SOLVER

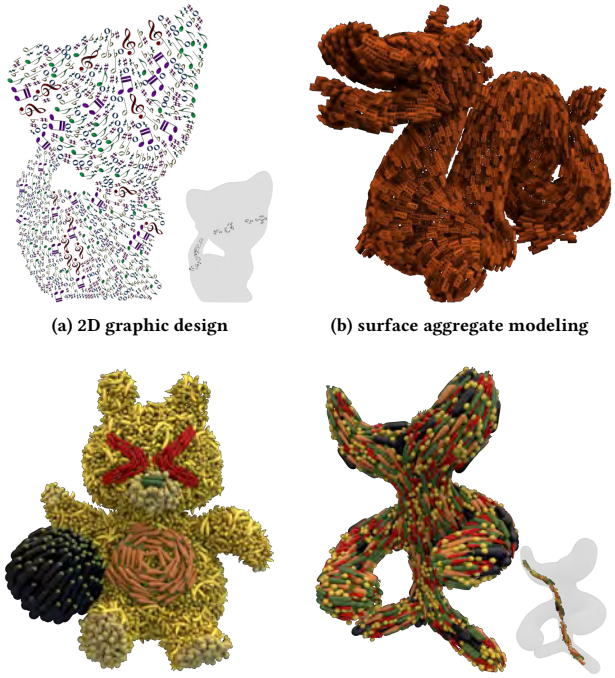
We iteratively optimize the set of samples for all objectives via a combination of the element-based method in [Ma et al. 2011] and gradient-based optimization (e.g. LBFGS) as exemplified in Figure 6.

6 DISCUSSIONS

The interactive brushing is a key feature to help users produce a variety of outputs, and the optimization of element fields can help reduce users' workload without compromising their control by automatically arranging the elements. In Figure 7, we show more works generated via our brushing system.

Our current prototype cannot synthesize continuous elements [Roveri et al. 2015] due to the complexity of brush controls in 3D and the difficulty of arbitrarily mixing these kinds of elements. A potential research direction is to extend our one-step process with [Lu et al. 2014; Zhou et al. 2014] for synthesizing continuous artistic patterns following direction fields.

Creating input exemplars can be tedious for ordinary users. To promote our developed system in public communities, we are investigating a more advanced framework to reduce the workload of preparing the input exemplars.



(a) 2D graphic design (b) surface aggregate modeling

(c) volumetric artistic collage (d) volumetric aggregate modeling

Figure 7: *Applications.* The works in (a) and (d) are automatically optimized from partially user-specified inputs.

REFERENCES

Jun Han Cho, Athena Xenakis, Stefan Gronsky, and Apurva Shah. 2007. Anyone Can Cook – Inside Ratatouille's Kitchen. In *SIGGRAPH 2007 Courses*.

Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. 2012. Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-ink Illustration. In *CHI '12*. 1727–1736.

Jingwan Lu, Connelly Barnes, Connie Wan, Paul Asente, Radomir Mech, and Adam Finkelstein. 2014. DecoBrush: Drawing Structured Decorative Patterns by Example. *ACM Trans. Graph.* 33, 4, Article 90 (2014), 9 pages.

Chongyang Ma, Li-Yi Wei, and Xin Tong. 2011. Discrete Element Textures. *ACM Trans. Graph.* 30, 4, Article 62 (2011), 10 pages.

Riccardo Roveri, A. Cengiz Öztireli, Sebastian Martin, Barbara Solenthaler, and Markus Gross. 2015. Example Based Repetitive Structure Synthesis. *Comput. Graph. Forum* 34, 5 (2015), 39–52.

Robert W. Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded Deformation for Shape Manipulation. *ACM Trans. Graph.* 26, 3, Article 80 (2007).

Shizhe Zhou, Changyun Jiang, and Sylvain Lefebvre. 2014. Topology-constrained Synthesis of Vector Patterns. *ACM Trans. Graph.* 33, 6, Article 215 (2014), 11 pages.