Autocomplete Element Fields

Chen-Yuan Hsu, Li-Yi Wei, Lihua You Jian, Jun Zhang

Autocomplete Element Fields

Chen-Yuan Hsu
Bournemouth University

Li-Yi Wei
Adobe Research

Lihua You
Jian Jun Zhang
Bournemouth University

(a) graphic design (b) artistic collage (c) aggregate modeling

Figure 1: *Autocomplete element synthesis following partial user specifications.* Our autocomplete system can be applied for different output domains such as 2D planes (a), 3D surfaces (b), and 3D volumes (c), distinct types of aggregate elements, and various applications such as design (a), collage (b), and modeling (c). The proposed system can automatically optimize the entire element distributions, orientations and scales based on the partial user strokes (a) (inset), enable users to interactively arrange the elements over the domain (b), and directly compute the volumetric output (c) from a given surface direction field (see Figure 15).

ABSTRACT

Aggregate elements are ubiquitous in natural and man-made objects. Interactively authoring these elements with varying anisotropy and deformability can require high artistic skills and manual labor. To reduce input workload and enhance output quality, we present an autocomplete system that can help users distribute and align such elements over different domains. Through a brushing interface, users can place and mix a few elements, and let our system automatically populate more elements for the remaining output. Furthermore, aggregate elements often require proper direction/scalar fields for proper arrangements, but fully specifying such fields across entire domains can be difficult or inconvenient for ordinary users. To address this usability challenge, we formulate *element fields* that can smoothly orient all the elements based on partial user specifications without requiring full input fields in any step. We validate our prototype system with a pilot user study and show applications in design, collage, and modeling.

Author Keywords

element, field, synthesis, anisotropy, interface, modeling

CCS Concepts

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

INTRODUCTION

Aggregate elements are widely used in various applications such as rendering [54, 55], modeling [51, 65, 63], simulation [24, 25], interaction [34, 33], and design [35, 17, 83, 38]. However, interactively authoring such elements with varying anisotropy and deformability for ordinary users still remains challenging, especially in 3D domains [7].

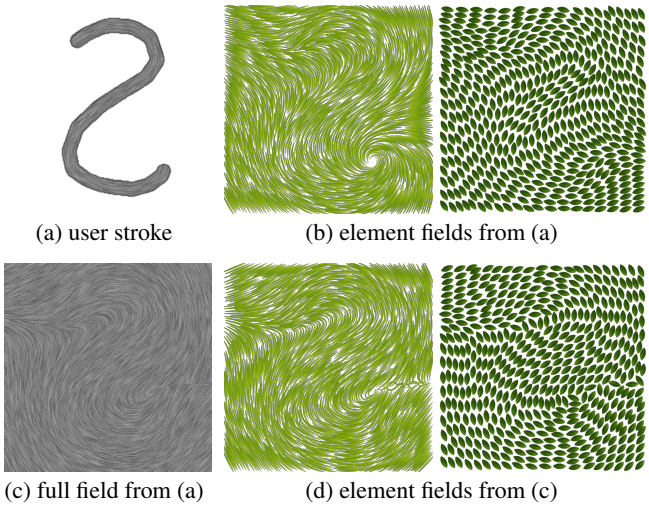
Manual creation of aggregate elements allows full control but can require high artistic skills and manual labor, while automatic batch synthesis can apply in 2D or 3D (e.g. [51, 65, 67]) but might not provide sufficient freedom and interactivity. Although there exist methods to automate interactive authoring of 2D elements (e.g. [34, 33]), users still need to place all broad strokes or position element collections step by step. Efficiently authoring elements with general shapes, distributions, and alignments in various output domains is significantly harder and lacks good solution so far. The ideal system should be user-friendly without requiring significant expertise or efforts and yet efficient and general enough to interactively design and explore diverse outcomes.

We propose an autocomplete system for interactive authoring of aggregate elements within different output domains including 2D planes, 3D surfaces, and 3D volumes (Figure 1). Analogous to conventional painting workflows, through a brushing

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
http://dx.doi.org/10.1145/3313831.3376248

interface, users can select and even mix distinct types of elements from an element palette, brush the selected elements over the output domain (e.g. 2D planar canvas or 3D object surface) and see the corresponding outcomes interactively.

To better orient aggregate elements, existing methods often need to incorporate full direction/scalar fields, but to obtain such fields across domains either manually or automatically can be challenging for ordinary users. The input fields, even when automatically computed, may also force undesirable artifacts, such as singularities which cannot be entirely avoided [56]. To reduce input workload and enhance output quality, we formulate *element fields* that can smoothly orient the whole elements following partial inputs (e.g. user strokes) based on inter- and intra-element relationships as illustrated in Figure 2.



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

(c) full field from (a) (d) element fields from (c)

Figure 2: *Element fields.* Our method can directly compute smoother element fields (b) from a partial input (a) and more adaptively fit different elements with (a) than (d) preprocessing a full input (c) from (a) via Laplacian interpolation [56].

Since to optimally compute an accurate layout for arbitrary mixtures can be time-consuming as described in [38, 67] and the computational complexity can further increase by element anisotropy and deformability as well as domain dimension, our algorithm is designed to balance between quality and interactivity. We thus devise an element representation that can better characterize elements with varying anisotropy and deformability to strengthen connection and reduce penetration. Our procedural approach can effectively synthesize aggregate elements based on user specifications and directly create novel mixtures from individual elements without the need to prepare input exemplars used in the data-driven methods (e.g. [51]).

The developed system, centered on the idea of *element fields*, can handle elements with general shapes, distributions, and alignments, offer more usability and interactivity than existing practices and enable users to create compelling artwork like Figure 1b without requiring significant expertise or efforts. We evaluate our prototype system with a pilot user study and show applications for both interactive design and collage as well as batch modeling, with partial or full input fields.

RELATED WORKS

Element packing

A variety of algorithms have been proposed to pack elements for various applications such as mosaics [21, 35, 10, 26], collages [17, 27, 29, 38, 67], glyphs [79, 83] and artistic layouts [69, 61, 68] in different domains. These algorithms optimize packing layouts without considering interactivity or user-specified element distributions, orientations and scales.

Element modeling

Modeling aggregate elements considering individual shapes and distributions can be achieved via data-driven methods [30, 51, 1, 39, 63, 20, 11] or procedural approaches [58, 42, 43, 65, 66]. By allowing merging or overlapping the elements, structures with desired appearance can be formed for practical manufacturing [82, 80, 6, 13]. However, these approaches mainly focus on automatic computation without providing sufficient user interaction (e.g. element mixing) and require fully specified input fields for appropriate element arrangements.

Unlike existing packing and modeling algorithms, our method can automatically optimize element distributions, orientations and scales with user specifications, and through a brushing interface, ordinary users can interactively arrange element collections over various output domains in an intuitive manner.

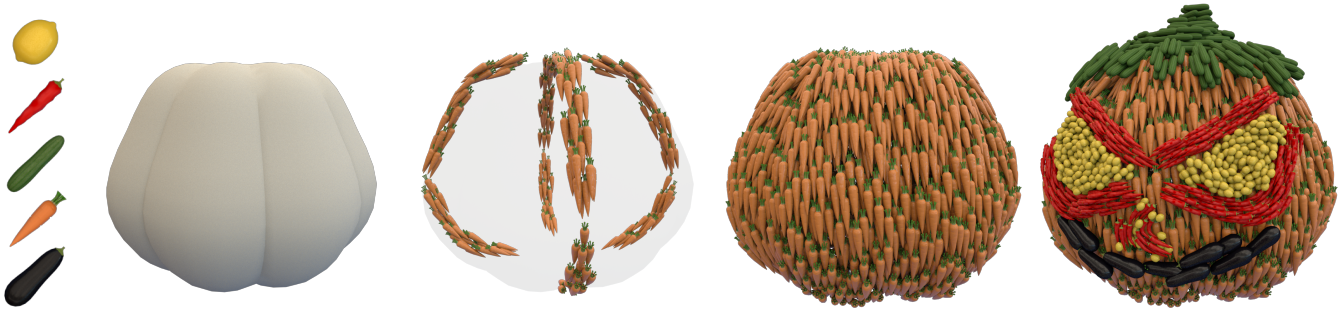
Anisotropic element placement

Placing anisotropic elements has various applications such as hatching [22, 57, 32], ornaments [68], calligrams [53], solids [74, 81], and visualization [56]. The anisotropic elements often need to follow certain direction fields, whose topology and resolution can directly influence the quality of the resulting outputs. Existing methods predominantly preprocess a full direction field and then force elements to follow. For better synthesis quality and user efficiency, our approach can directly compute smooth element fields (e.g. Figure 2b) according to user intention (e.g. partial user strokes) without requiring fully specified input fields from users in any algorithmic step.

Interactive design

Interactive design systems have been presented for patterns [62, 47, 45, 49, 46, 48], elements [34, 33], streamlines [70, 5], distributions [14, 11], geometric textures [4, 20, 16], and dynamic effects [78], but these proposed interfaces are tailored for specific applications in 2D planes and 3D surfaces. Interactive texturing with a general interface for applications across different output domains is a more challenging problem and has received less attention so far, especially in 3D volumes. Although a 3ds Max plugin [18] aims for interactive placement and simulation of rigid objects in 3D domains, similar to element packing, elements following directions are not considered aside from basic gravity and collision.

Analogous to a canvas-based palette tool [71], our brushing interface offers general and intuitive controls that enable users to interactively produce desired outcomes following their intention for various applications within different domains. Like common color mixing in [72], user-interactive element mixing can also be achieved via the interface. Our proposed system can not only more naturally fit element synthesis with interactive authoring but also provide higher functionality for users.



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

Figure 3: *Workflow.* Users can select single or multiple input elements (a), brush the chosen inputs over the domain (b) and see the brushing results (c) interactively. According to (c), our system can automatically create the full output (d) with intended element orientations and scales. The users can further arrange the elements for the final outcome (e) via corresponding brush operations.

SYSTEM DESIGN RATIONALE

We recap our system design rationale for element synthesis as follows. Manual placement offers full authoring freedom but can require heavy efforts and expertise, while prior practices need full input fields for arrangements but such fields can be hard for users to specify. Even if the fields can be supplied, the final outputs (e.g. Figure 2d) might not fit user intention. We thus aim for an interactive system that retains full user control and yet can smoothly synthesize distinct types of elements with partial user specifications. Through our system, users can iteratively design diverse compositions with reduced workload and enhanced quality under the same brushing workflow without needing to learn any field design systems like [56].

USER INTERFACE

As illustrated in Figure 4, our brushing interface is directly implemented into Autodesk Maya to leverage its overall system manipulation such as camera control. Users can choose and mix different elements from the element palette, tune relative parameters from the control panel, and perform corresponding brush operations to create desired works through the brushing canvas. The basic workflow is exemplified in Figure 3.

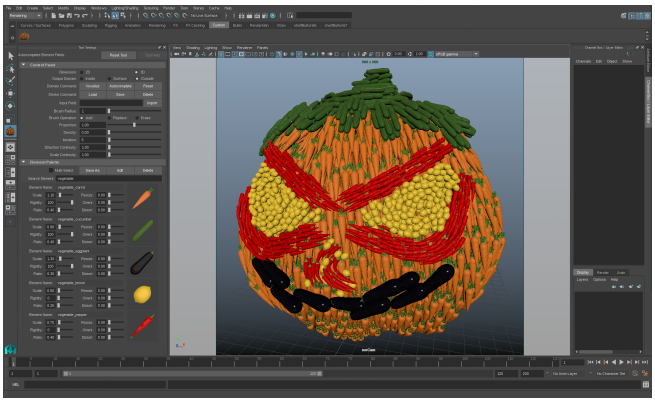


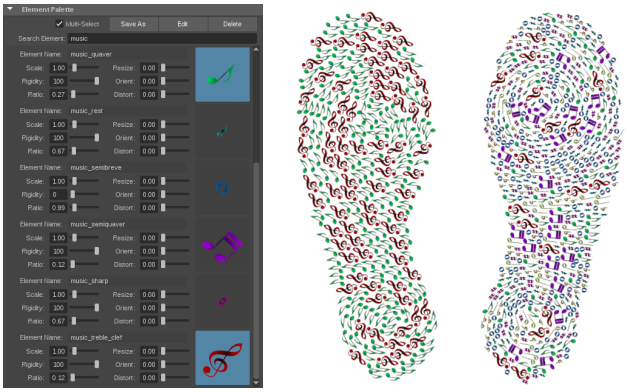
Figure 4: *User interface.* The element palette and the control panel are on the left. The brushing canvas is in the middle.

Brushing canvas

Analogous to traditional painting interface, we provide a brushing canvas for users to interactively arrange elements. Users can brush a few strokes of elements, and click a button to let our system autocomplete the rest. The output domain can be a 2D plane, 3D surface, or 3D volume for various applications.

Element palette

Users can directly select single or mix multiple input elements and set various properties such as scale and rigidity respectively via the element palette (Figure 5). The combination of the selected elements can be saved as a new palette entry for further use and remix analogous to color palettes [72].



(a) element palette (b) two types (c) eight types

Figure 5: *Element palette.* Through the element palette (a), several kinds of music symbols can be chosen by users to form a shoeprint consisting of only two types (b) or eight types (c).

Brush operations

In our system, we offer *add*, *erase*, and *replace* brush operations for interactive authoring. The *add* operation enables users to directly place elements across the domain while the *erase* operation can remove the already placed elements. The *replace* operation is a combination of *erase* and *add* operations. For automatic completion, the user-specified strokes (e.g. Figure 3c) can indicate the intended overall element orientations and scales, and instead of keeping these strokes unchanged, our autocomplete system optimizes the entire inter- and intra-element relationships (e.g. Figure 3d) to improve total synthesis quality and produce smoother results while adequately observing and reflecting the original user intention.

FORMULATION

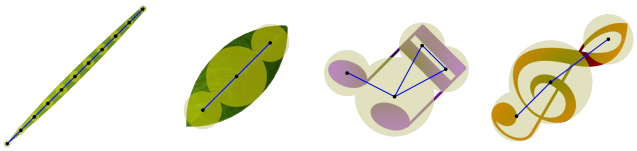
Given input element exemplars $I = \{e_1, \ldots, e_l\}$, an output domain D, and a partially specified direction or scalar field O over D, our goal is to automatically compute an output X composed of I such that all elements $e \in I$ are well distributed within D and have coherent orientations or scales with O.

Element Representation

We represent each element e by a set of samples $\{s_1, \ldots, s_m\}$ as [51, 50]. However, only using samples is not enough to properly depict elements with varying anisotropy and deformability. We thus extend the representation with weights and graphs.

Sampling

Since elements are represented by 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 6, instead of using unweighted samples in [51, 50], we adopt fewer weighted samples to better characterize distinct types of elements. Moreover, a graph structure $\mathbf{G}$ inspired by Sumner et al. [73] is further employed to indicate the connectivity among samples and their relationship with element shapes.



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

Figure 6: *Element representation.* The black dots indicate the sample positions. The radii of the yellow circles represent the sample weights $\mathbf{w}_s$. The blue lines denote the element graphs. The sampling can be sparse (a), dense (b), overlapping (c) or hybrid (d). These samples can be either manually placed by designers or assisted by algorithms [2, 77, 75, 41].

Sample attributes

In addition to the sample weight $\mathbf{w}_s$, each sample s contains individual attributes such as spatial position $\mathbf{p}$, orientation matrix $\mathbf{o}$ and scale $\mathbf{c}$. In our formulation, the attributes $\mathbf{p}$, $\mathbf{o}$ and $\mathbf{c}$ are variables to optimize for element synthesis.

Element Distribution

In order to well distribute aggregate elements, we devise a distribution objective E_e to effectively handle both inter- and intra-element relationships. E_e consists of a sample distribution E_d term, a conflict check E_k term, and a graph similarity E_g term. E_d and E_k measure the quality of inter-element distributions, and E_g preserves the intra-element connections for elements composed of multiple samples.

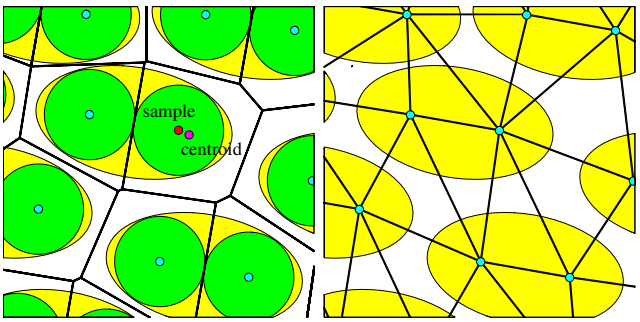
Sample distribution

To properly position elements within the given domain, we can calculate a balanced distance between each sample to evenly maintain their inter-element distributions. We utilize a power diagram [3] that partitions the domain based on the weighted samples of elements to obtain a potential position for each sample. As demonstrated in Figure 7, analogous to general Lloyd-like methods, we can move each sample to the centroid of its power cell to acquire evenly-distributed samples. Hence, the sample distribution E_d term can be formulated as:

$$E_d(X) = \sum_{s \in X} |\mathbf{p}(s) - \mathbf{centroid}(s)|^2, \tag{1}$$

where $\mathbf{centroid}(s)$ represents the centroid of s's power cell.

Here we quantify sample distributions procedurally instead of from input exemplars (like in [23]) to facilitate faster computation and more flexible manipulations (e.g. element mixing).



(a) centroid (b) triangulation

Figure 7: *Power diagram.* In (a), a potential position (purple) for a sample (red) can be derived from the centroid of the sample's power cell. A weighted Delaunay triangulation (b), which is the dual graph of the power diagram, can be further used to construct neighborhood information for each sample.

Conflict check

Mixtures consisting of elements with distinct sizes or shapes (e.g. Figure 1a) might not have the relevant distribution space partitioned by the power diagram. In order to increase synthesis quality for arbitrary mixtures, E_k is exploited to avoid the conflicts between samples by checking the distances between the weighted samples. Since the distance between two samples should not be less than the sum of their weights, we can formulate the conflict check E_k term as:

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

where $\mathbf{N}_s$ denotes a set of all neighboring samples of s derived from a triangulation (Figure 7b), $\hat{\mathbf{p}}$ is the displacement between s' and s we want to solve, and $\check{\mathbf{p}}$ means the current displacement between s' and s treated as a constant during optimization. Note that since elements might be represented by overlapping samples, E_k ignores samples in the same element.

Graph similarity

Elements consisting of multiple samples should keep similar graph structures. To appropriately preserve the intra-element connections for such elements, we can measure the distance

between the current graph $\mathbf{G}(s)$ and the original graph $\mathbf{G}'(s)$ for each sample s to consistently maintain the graph structure of each element and define the graph similarity E_g term as:

$$\begin{aligned} E_g(X) &= \sum_{s\in X} d\left(\mathbf{G}(s),\mathbf{G}'(s)\right), \\ d\left(\mathbf{G}(s),\mathbf{G}'(s)\right) &= \sum_{s'\in\mathbf{G}_s} \left|\hat{\mathbf{p}}\left(s',s\right) - \mathbf{c}(s)\mathbf{o}(s)\hat{\mathbf{p}}'\left(s',s\right)\right|^2, \end{aligned} \tag{3}$$

where $\mathbf{G}_s$ denotes a set of all connected samples of s, $\hat{\mathbf{p}}$ is the same as in Equation (2), and $\hat{\mathbf{p}}'$ represents the displacement between s' and s in the original graph treated as a constant.

Putting everything together, we have:

$$E_e = (E_d + E_k) \oplus \mathbf{w}_g E_g, \tag{4}$$

where $+$ means that E_d and E_k are minimized together, $\oplus$ indicates that E_g is minimized separately during optimization, and $\mathbf{w}_g$ is a relative weight set to 100 from our experiment to firmly restructure the graphs from unexpected sample distributions.

Element Fields

In addition to forming well-distributed elements, these synthesized elements should also match the user-specified field O while behaving reasonably well in the unspecified portions of the output domain D. We design a field objective E_f for both scalar and direction fields to include: a field alignment E_a term to effectively align the elements with O in the specified areas, a field continuity E_c term to smoothly orient all the elements over D, and an element rigidity E_r term to determine the element deformability. Each term is also formulated in samples for consistency. Here we first define the distance between two scales $\mathbf{c}$ and $\mathbf{c}'$ for scalar fields and the distance between two orientation matrices $\mathbf{M}$ and $\mathbf{M}'$ for direction fields as:

$$d\left(\mathbf{c},\mathbf{c}'\right) = \left|\mathbf{c}-\mathbf{c}'\right|^2; \tag{5}$$

$$d\left(\mathbf{M},\mathbf{M}'\right) = \sum_{i=1}^{n} \left|\mathbf{M}_i - \mathbf{M}'_i\right|^2, \tag{6}$$

where n denotes the domain dimension, and $\mathbf{M}_i$ and $\mathbf{M}'_i$ represent the i-th column vectors of $\mathbf{M}$ and $\mathbf{M}'$ respectively. Below we mainly detail the direction field objective E_f^d and then briefly describe the scalar field objective E_f^s.

Field alignment

Since each sample $s \in e$ is associated with a local orientation $\mathbf{o}$, the sample orientation $\mathbf{o}(s)$ in the specified domain should be properly aligned with the given field orientation $O(p)$, where $O(p)$ means the orientation matrix of O at p which is the closest specified domain point to the sample s. Hence, the distance between $\mathbf{o}(s)$ and $O(p)$ for each sample s in the specified areas should be minimized for the field alignment E_a term as:

$$\begin{aligned} E_a(\mathbf{o},O) &= \sum_{s\in X} \mathbf{w}_a(s)\, d\left(\mathbf{o}(s),O(p)\right), \\ \mathbf{w}_a(s) &= \begin{cases} \frac{\pi(\mathbf{c}(s)\mathbf{w}_s(s))^2}{A} & \text{in 2D} \\ \frac{\frac{4}{3}\pi(\mathbf{c}(s)\mathbf{w}_s(s))^3}{V} & \text{in 3D} \end{cases}, \end{aligned} \tag{7}$$

where $\mathbf{w}_a$ is a confident weight which means larger samples dominating larger domain spaces should contribute more influences than smaller samples, and A and V are the 2D area and 3D volume of a domain unit (e.g. pixel/voxel) respectively.

Field continuity

To fully complete the element alignments (including regions with or without user-specified O), we orient nearby samples as similarly as possible. Additionally, the effect of field continuity for each pair of samples with a closer distance or similar orientations should be increased accordingly in order to smoothly fit all the samples with their nearby samples. As a result, the field continuity E_c term can be formulated as:

$$\begin{aligned} E_c(\mathbf{o}) &= \sum_{s\in X}\sum_{s'\in\mathbf{N}_s} \mathbf{w}_c\left(s,s'\right) \sum_{i=1}^{n} \mathbf{w}_o\left(s,s',i\right) \left|\mathbf{o}_i(s) - \mathbf{o}_i(s')\right|^2, \\ \mathbf{w}_c\left(s,s'\right) &= \frac{1}{1+\left(\mathbf{w}_{\swarrow}(s,s')^{-1}\right)^2}, \mathbf{w}_o\left(s,s',i\right) = \left(1+\mathbf{o}'_i(s)\cdot\mathbf{o}'_i(s')\right), \end{aligned} \tag{8}$$

where $\mathbf{w}_c$ is an inverse quadratic radial basis function used to adaptively adjust the effect according to $\mathbf{w}_{\swarrow}(s,s')^{-1}$ in Equation (2) (i.e. the inter-distance between s and s' altered based on both $\mathbf{w}_s$ and $\mathbf{c}$), $\mathbf{w}_o$ aims to better orient samples with their nearby samples which have similar orientations, $\mathbf{o}_i$ represents the i-th column vector of $\mathbf{o}$ as in Equation (6), and $\mathbf{o}'_i$ treated as a constant here denotes the i-th column vector of the current orientation matrix $\mathbf{o}'$ of a sample. For dimension reduction, we optimize the closest three/four neighboring samples s' chosen from $\mathbf{N}_s$ for each sample s in 2D/3D according to $\mathbf{w}_{\swarrow}(s,s')^{-1}$.

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 term is consequently formulated as:

$$E_r(\mathbf{o}) = \sum_{s\in X}\sum_{s'\in\mathbf{G}_s} \mathbf{w}_r(e)\, d\left(\mathbf{o}(s),\mathbf{o}(s')\right), \tag{9}$$

where $\mathbf{w}_r(e)$ is the weight of element rigidity which can be defined by users via the element palette (e.g. 0 for deformable elements and 100 for rigid elements in our implementation).

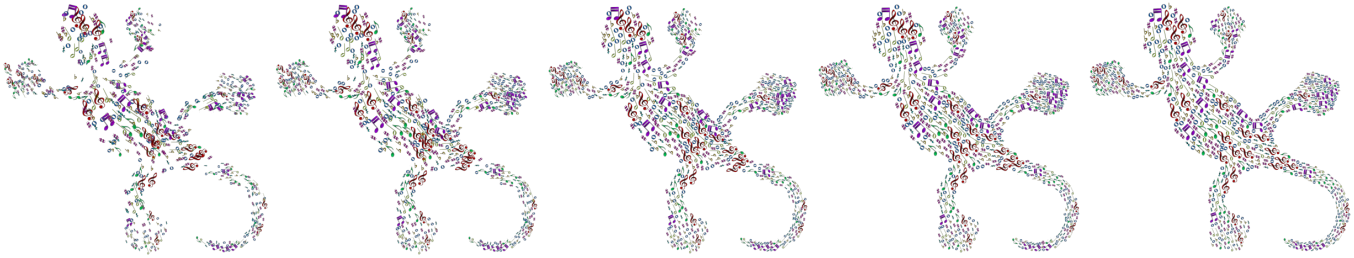
Since we define the local orientation $\mathbf{o}$ as a rotation matrix, in order to penalize the deviation of $\mathbf{o}$ to well keep a pure rotation matrix during the optimization as in [73], we apply an extra constraint E_o term for the direction field objective E_f^d as:

$$E_o(\mathbf{o}) = \sum_{s\in X} \mathbf{R}(\mathbf{o}(s)), \tag{10}$$

where $\mathbf{R}(\mathbf{o}) = (\mathbf{o}_1\cdot\mathbf{o}_1-1)^2 + (\mathbf{o}_2\cdot\mathbf{o}_2-1)^2 + (\mathbf{o}_3\cdot\mathbf{o}_3-1)^2 + (\mathbf{o}_1\cdot\mathbf{o}_2)^2 + (\mathbf{o}_2\cdot\mathbf{o}_3)^2 + (\mathbf{o}_3\cdot\mathbf{o}_1)^2$, and $\mathbf{o}_1$, $\mathbf{o}_2$ and $\mathbf{o}_3$ are the corresponding column vectors of $\mathbf{o}$. As a result, the direction field objective E_f^d is a summation of E_a, E_c, E_r and E_o as:

$$E_f^d = E_a + \alpha^2 E_c + E_r + E_o. \tag{11}$$

Replacing Equation (6) by Equation (5) in Equations (7) to (9) and removing $\mathbf{w}_o$ from Equation (8), we represent the scalar



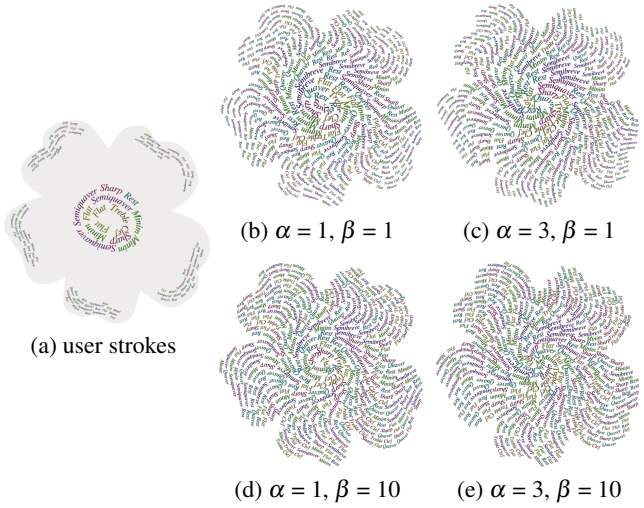
(a) initialization (b) iteration 2 (c) iteration 8 (d) iteration 20 (e) iteration 100

Figure 8: *Optimization process.* Our solver iteratively optimizes the scale, orientation and position of each element sample. The number of iterations for automatic completion can range from 5 to 100 depending on the number of elements, the completeness of fields, the complexity of mixtures, and the domain size and shape. Users can use fewer iterations (e.g. (d)) for fast previews during the iterative design procedure and utilize more iterations (e.g. (e)) for final designed outcomes or more complicated scenarios.

field objective E_f^s as follows:

$$E_f^s = E_a + \beta^2 E_c + E_r. \tag{12}$$

Note that α and β (both default values $= 1$) are parameters that users can tune for specific scenarios/applications (Figure 9).



(a) user strokes (b) $\alpha = 1, \beta = 1$ (c) $\alpha = 3, \beta = 1$ (d) $\alpha = 1, \beta = 10$ (e) $\alpha = 3, \beta = 10$

Figure 9: *Field continuity.* By increasing α and β via our interface, it can encourage smoother element orientations and scales. Our system can observe the user intention to optimize overall synthesis quality with the tuned values accordingly.

SOLVER

We optimize the set of samples for the objectives E_e and E_f respectively through a combination of the Lloyd-like method and gradient-based optimization in an iterative way. The overall optimization process is exemplified in Figure 8.

Initialization

We randomly place individual elements into the output domain, and the orientations and scales of samples of each element are initially set to the stroke direction and brush size captured in the closest user-specified domain point to the element.

Iterations

In each iteration, we take a distribution step, an orientation step, and a scale step for different objectives, and each step optimizes only one variable (e.g. $\mathbf{p}$, $\mathbf{o}$, or $\mathbf{c}$) at a time. At the beginning, each sample's neighborhood is built via the power diagram (Figure 7). Based on this, we first carry out the scale step for E_f^s and then run the orientation step for E_f^d. The distribution step is fulfilled for E_e at the end of each iteration.

In the scale step, $\mathbf{c}$ is the variable, and we can directly solve E_f^s via linear least squares. In the orientation step, $\mathbf{o}$ is the variable, and due to the E_o term, we optimize E_f^d via the Levenberg-Marquardt method [52] to solve this nonlinear least squares problem. Finally, in the distribution step, $\mathbf{p}$ is the variable, and we first solve E_d and E_k together to optimally rearrange the inter-element distributions via linear least squares. While the elements are composed of multiple samples, to precisely structure the intra-element connections, E_g multiplied by $\mathbf{w}_g$ can be solved according to the optimized positions, orientations and scales of samples via linear least squares as well.

IMPLEMENTATION

To obtain a brushing canvas, we voxelize the selected object and employ a sparse octree to indicate all the voxels similar to [36]. Each voxel also represents a domain point p and captures relevant information such as stroke direction and brush size. We also calculate the volume of each input element by voxels in preprocessing, so the total number of elements placed into the domain can be proportionally estimated according to the number of domain voxels. During the initialization, a k-d tree is utilized for the nearest domain point search. Furthermore, we extend Voro++ [64] to compute the power diagram in parallel and construct the sample neighborhoods. By adding extra fixed samples to the boundary voxels of the selected object, we can apply the conflict check term to the boundary samples and their nearby element samples, and thus boundary condition for various domain shapes can be efficiently handled. For the final output, the element samples can be used as control points to regularize the mesh vertices of elements as in [73].

EVALUATION

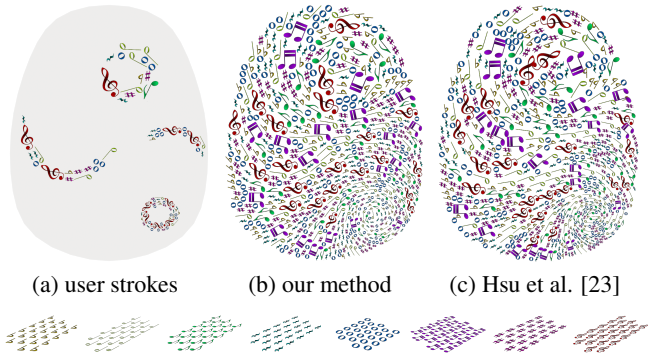
Element manipulation

Since previous algorithms either only work for rigid elements (e.g. [61, 38]) or need extra schemes such as physics solvers

(a) user strokes (b) density field (c) dense (d) sparse (e) spatially-varying

Figure 10: *Varying distributions.* Users can directly specify strokes (a) with density values to obtain a partial density field (b) via the same brush operations. Without adopting (b), dense (c) or sparse (d) element distributions can be uniformly formed by adjusting the number of elements placed into the domain. By incorporating (b), the elements can be distributed non-uniformly (e).

for deformable elements (e.g. [51, 67]), the capability of freely mixing user-specified elements is confined. Although an earlier version of our method [23] allows users to mix different elements from user-prepared input exemplars (Figure 11), its data-driven process can significantly slow down computation and thus compromise both usability and interactivity, and the prepared exemplars can also heavily limit the synthesis quality as well as the freedom of element distributions. In contrast, our procedural approach can enable users to individually mix distinct types of elements from the element palette without the necessity of preparing the input exemplars in advance and directly bring novel mixtures into existence as users desire as possible in high quality and efficiency without requiring extra schemes. Moreover, by assigning the samples of each element arbitrary local orientations and scales via the interface, users can further randomly orient, distort and resize the synthesized elements to enrich visual diversity of artworks (Figure 12).



(a) user strokes (b) our method (c) Hsu et al. [23]

(d) input exemplars used in (c)

Figure 11: *Solver comparison.* Our procedural approach (b) can more uniformly distribute elements in better performance than the data-driven method (c). Both contain the same 689 elements and 1598 samples, and the synthesis times of (b) and (c) are 9 and 44 seconds respectively in 100 iterations.

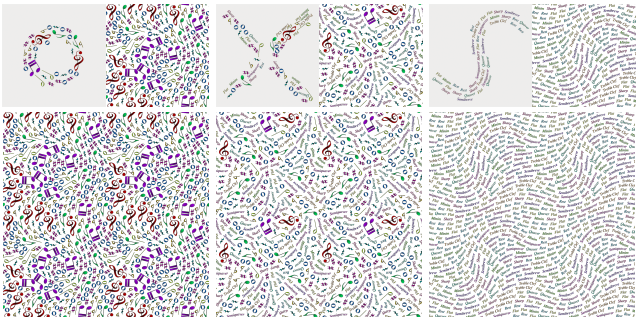
Distribution manipulation

Due to algorithmic limitations (e.g. computational cost), prior approaches cannot let users interactively manipulate element distributions, while our system enables users to freely form diverse compositions, such as spatially-varying element distributions, under the same brushing workflow. More detailedly, we can incorporate an extra density field to spatially vary the density of element distributions as illustrated in Figure 10. To

(a) aggregate vegetables (b) tangled metals

Figure 12: *Element mixing.* Users can create compelling mixtures composed of user-specified elements with chaotic alignments (a), distorted shapes (b) or varying sizes (Figure 1c).

fulfill it, each stroke can be equipped with a density value ρ between 0 and 1, and we directly modify the weights of samples by the given densities $\rho(s)$ but do not change the sizes of elements (e.g. $\mathbf{w}'_s(s) = (1+\rho(s))\mathbf{w}_s(s)$). This optional density field can be optimized individually via the same formulation and step with E^s_f during the iterations. With toroidal boundary conditions, our method can further generate 2D/3D element tiles to seamlessly tile a large output as shown in Figure 13.

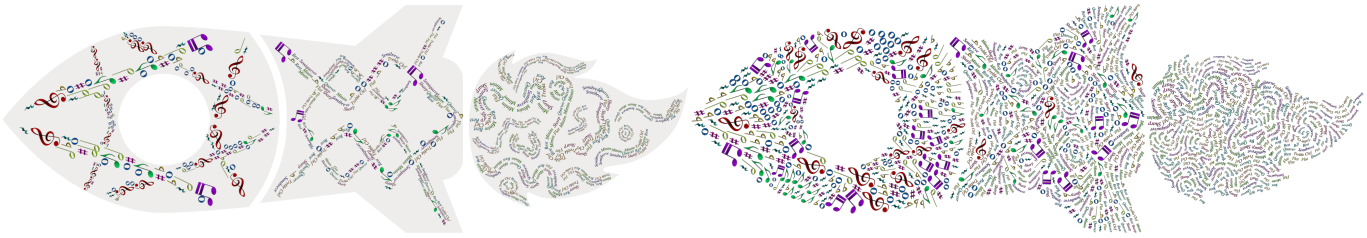


(a) rigid (b) mixed (c) deformable

Figure 13: *Tile-based distributions.* Our system can produce various element tiles. Note that in (b), we randomly align the music symbols, while the words follow the stroke directions.

Field manipulation

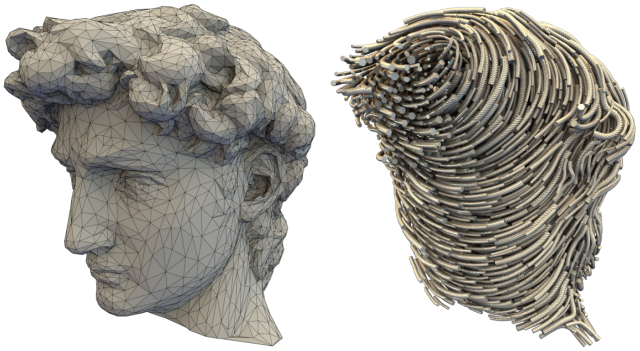
As exemplified in Figure 9, by tuning α and β from the interface to manipulate the effect of field continuity, different



(a) overlapping (b) intertwined (c) distorted (d) incoherent element fields from (a), (b) and (c) respectively

Figure 14: *Incoherent user strokes.* Our method can well handle overlapping (a), intertwined (b) and distorted (c) user strokes with incoherent directions or scales for desirable results (d), while to deal with such cases can be hard for users via existing methods.

outputs with desired appearances can be designed according to users' personal preferences. Moreover, Figure 14 shows that incoherent user strokes can be well tackled by our method without incorporating any extra processes, and Figure 15 demonstrates that our system can also automatically generate volumetric element distributions from only surface direction fields (a very small subset relative to the whole volume) without predefining the entire field resolution. More examples about field manipulation can be seen in the supplementary document.



(a) input mesh (b) cross section of Figure 1c

Figure 15: *Sparse input fields.* We compute an extrinsically smooth field [28] over a sparse set of mesh vertices (a) but apply our method to the entire volume not just the surface. The cross section (b) shows that the full outcome (Figure 1c) can be smoothly constructed from the sparse direction field.

Performance

The synthesis times of our representative works are reported for reference in Table 1. In our current prototype, there exists room to improve the performance. We leave it for future work.

USER STUDY

Our autocomplete system aims for reducing input workload (by partial user specifications) and enhancing output quality (by smooth field topology). To evaluate our system, we have conducted a pilot user study, and the goal of our study procedure was designed to measure how much workload users can reduce via our system while achieving the designated targets. The study participants included 2 professional artists and 3 novice users without experience in element authoring.

2D case	# elements	# samples	time	# iterations
Figure 1a	725	1656	9s	100
Figure 9b	303	1276	15s	200
Figure 10c	381	1560	8s	100
Figure 10d	215	905	5s	100
Figure 10e	274	1139	7s	100
Figure 13a	187	402	4s	200
Figure 13b	194	650	6s	200
Figure 13c	206	838	8s	200
Figure 14a	367	849	5s	100
Figure 14b	516	1675	18s	200
Figure 14c	355	1477	28s	400
3D case	# elements	# samples	time	# iterations
Figure 1b	3994	19970	28s	5
Figure 1c	6280	43960	170s	20
Figure 12a	5272	15931	60s	50
Figure 12b	5344	26344	145s	50

Table 1: *Synthesis timing.* The CPU we use is Intel® Xeon® E5-1650 3.20GHz. Note that Figures 9b to 9e have similar synthesis times as well as the same numbers of elements and samples, and Figures 14a to 14c here represent the corresponding outputs in Figure 14d respectively.

Procedure

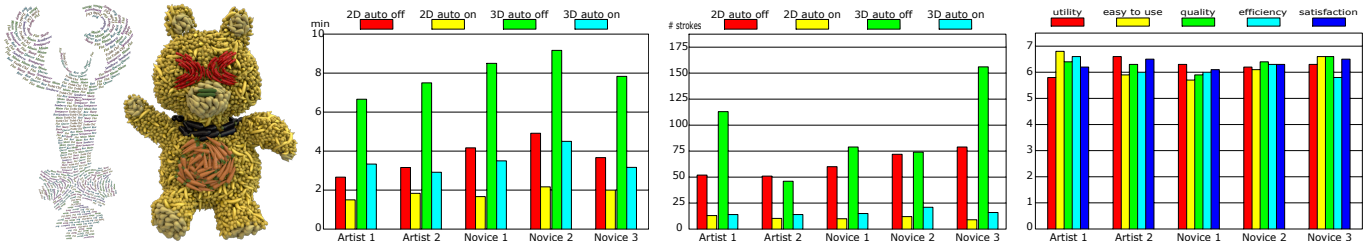
The study includes four sessions: warm-up, target brush, open brush, and final interview. All tasks were conducted on a desktop computer with a mouse and a Wacom tablet, and the whole study took around 2 hours per participant on average.

Warm-up session

This session aims to help the novice users have basic understanding of Autodesk Maya such as camera control and familiarize all the participants with our system. The process consists of interactive authoring and automatic completion for given 2D and 3D objects. One of the authors helped the participants to realize the brushing workflow through the session.

Target brush session

In this session, we aim to measure the usability of our brushing interface for the participants with different levels of expertise. During the session, each participant was asked to create similar outcomes to the designated targets (Figure 16). Although the 2D target (Figure 16a) can be achieved by Adobe Photoshop or Illustrator (e.g. [40, 76]) and the 3D target (Figure 16b) can be fulfilled by PhysX Painter (e.g. [19]) plus manual placements, it can require significant artistic skills and manual labor from users in a time-consuming procedure. Thus, instead of asking the participants to achieve the designated outputs by



(a) 2D target (b) 3D target (c) production time (d) number of user strokes (e) user feedback

Figure 16: *User study targets, outcome statistics, and user feedback.* Similar outputs to (a) and (b) were produced by the participants during the study procedure. The production time (c) and the total number of user strokes (including both placed and erased strokes) (d) for each target were measured individually. In (e), the participants marked our autocomplete system in terms of *utility*, *easy to use*, *quality*, *efficiency*, and *satisfaction* , and all quantities are expressed in a 7-point Likert scale.

existing tools, we let our participants directly create results through our system under two conditions: autocomplete on, and autocomplete off (i.e. synthesizing elements only under the user strokes). We then recorded the production time and the total number of user strokes for each task respectively.

Open brush session

To identify possible usability, we encouraged the participants to freely create other works and observed their behavior for potential improvement in the session. One of the authors also accompanied the participants, provided related supports, and discussed conceivable usage with them throughout the process.

Outcome

Figures 16c and 16d offer quantitative measures of production time and stroke counts for each target task. The outcome statistics demonstrate that our system can significantly reduce both the production time and the number of user strokes for either artists or novices. Please refer to Figures 27 and 28 for the sample user study outputs. Below we highlight the participants' authoring behaviors from our observation.

For the 2D target, without the autocomplete mode, the participants had to frequently adjust the brush size and place strokes following the target's element directions one by one to fill the entire domain. Even if they could quickly and intuitively brush all over the domain through our interface, like common sketching and painting, to perfectly place the strokes side by side was not easy to them as the brushing paths could be a little tilted. Hence, the outputs about manual brushing can contain some obvious gaps and the orientations and scales of elements are not smooth enough either. While with the autocomplete mode, the participants placed a few strokes around the singularity and some other evenly spaced strokes within the remaining domain for automatic completion. The results created by our autocomplete method can have smoother element distributions, orientations and scales as well as similar appearances to the 2D target even if the partially specified strokes are not the same. In general, some of the user strokes might follow the target's outline but the outline did not actually affect their authoring strategies, while the target's element directions could affect where to place the strokes for automatic completion.

Similar outputs to the 3D target could be made via both without and with the autocomplete mode as the participants did not need to consider element alignments for the target's shape (i.e. lemons and bananas), but it can be seen that with the mode, the outcomes can have overall tighter element distributions. An interesting situation is that since the target's nose (i.e. cucumbers) has precise element alignments (i.e. horizontal directions), to avoid obvious element misalignments appearing at the target's nose, multiple participants demanded to brush more than once for better oriented cucumbers, while the rest (except the target's shape) could be specified with a single brush stroke in most cases. In summary, through our autocomplete system, the participants could more flexibly produce the designated results with reduced user workload and enhanced synthesis quality without compromising their control.

Feedback

Figure 16e summarizes each participant's feedback about our system. Overall, the participants were content with the system and commented that they can easily learn our system and directly produce desirable results without requiring significant practice and expertise as the brushing interface fits the natural artist workflow. They also said that with our system, they only need to design a few specific strokes for automatic completion instead of full element arrangements, and this concept is creative and novel to them. When asked about comparing manual placement with autocomplete, the participants stated that the system can indeed help them reduce workload and obtain smooth layouts in an efficient way, and it can encourage them to explore more interesting experiments. They were happy to see that our system can improve existing production pipelines for artists and benefit more novices in artwork creation.

The participants also made comments related to possible enhancement and usage for our system. The artists recommended that it is desirable to have more advanced brush controls such as a single stroke with varying brush radii for more varied effects, and being able to save the designed strokes as presets for further reuse (i.e. relocating a set of predefined strokes at a desired position) may be handier. Some participants expressed that after automatic completion, to let users slightly adjust a few elements with specified orientations or scales as partial inputs for local optimization can be an alternative for local field manipulation as it can still work under our formulation, and this may benefit some scenarios which need more precise control of flow directions. Since our current prototype is a

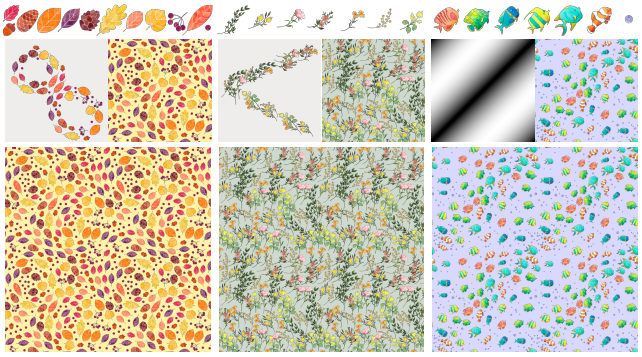
general interface for interactive texturing across different domains, we believe that their advice can be integrated into our framework for tailored applications according to demand.

OTHER APPLICATIONS

In addition to graphic design, artistic collage, and aggregate modeling, below we show more applications from our system.

Pattern design

Element arrangements are important for pattern design [44], but interactively designing tile-based patterns still remains challenging since existing tools require tedious manual process (see [9, 59]). As shown in Figure 17, our autocomplete system can form tileable patterns with a variety of element distributions without requiring a great deal of labor from users.



(a) colorful pattern (b) floral pattern (c) fish pattern

Figure 17: *Pattern design.* Tileable patterns with dense (a), evenly overlapping (b) or spatially-varying (c) element distributions can be formed by relevant user specifications via our system. Note that in (c), we utilize the user stroke in Figure 13c and a tileable density map to generate the pattern. The input elements used here are designed by Freepik [15].

Solid texturing

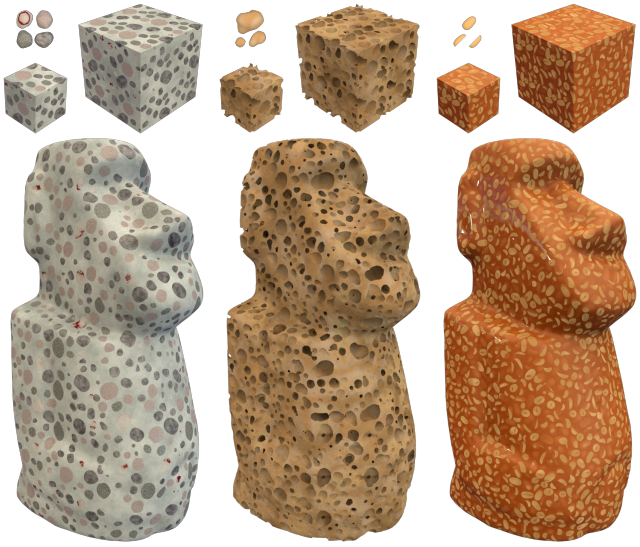
Since most proposed algorithms [31, 60, 37, 12] create solid textures from 2D input exemplars such as photos, the exemplars with specific structures might not be well synthesized. There is also a lack of algorithms which can generate tile-based solid textures from user-specified elements. As illustrated in Figure 18, our method can not only create such solids from specific elements but also maintain the element configuration.

Field visualization

As demonstrated in Figure 19, by properly aligning anisotropic elements with underlying fields, our algorithm can be considered as a geometry-based field visualization technique.

LIMITATIONS AND FUTURE WORK

Since we randomly place elements into the output domain for fast initialization, the distribution optimization might be trapped in a local minimum when the elements are highly anisotropic and the domain is irregular. As in Figure 1a, the distribution densities in the four feet are slightly different, but this can be relieved via a teleportation scheme like [8] or a progressive initialization like [11]. In addition, our approach cannot entirely avoid interpenetrations or floating elements,



(a) pebble cement (b) beehive rock (c) peanut caramel

Figure 18: *Solid texturing.* By randomly resizing and orienting a few elements, tile-based solid textures can be created for concrete (a), eroded (b) and colloidal (c) structures. Note that in (b), the elements are used as cutting objects to trim models.

but the issue is not visually obvious in our results and it is also feasible to integrate physics solvers into our framework if necessary. While autocompleting large outputs (e.g. Figure 12), our system may lack interactive speed. To incorporate GPU computing for further speedup can be an alternative solution.

For future work, since our current prototype does not consider continuous elements [63] due to the difficulty of mixing such elements, a potential direction is to extend our algorithm with [63] for continuous element synthesis. Additionally, VR brushing has recently received significant attention as it provides an immersive environment for 3D painting and modeling. We plan to implement our method under VR and offer a user interface for interactive authoring in immersive environments.

REFERENCES

[1] Zainab AlMeraj, Craig S. Kaplan, and Paul Asente. 2013. Patch-based Geometric Texture Synthesis. In *CAE '13*. 15–19.

[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. 2001. The Power Crust. In *SMA '01*. 249–266.

[3] F Aurenhammer. 1987. Power Diagrams: Properties, Algorithms and Applications. *SIAM J. Comput.* 16, 1 (1987), 78–96.

[4] Cyprien Buron, Jean-Eudes Marvie, Gaël Guennebaud, and Xavier Granier. 2015. Dynamic On-mesh Procedural Generation. In *GI '15*. 17–24.

[5] Guoning Chen, Vivek Kwatra, Li-Yi Wei, Charles D. Hansen, and Eugene Zhang. 2012. Design of 2D Time-Varying Vector Fields. *IEEE Transactions on*

(a) 2D plane

(b) 3D surface (c) 3D volume

Figure 19: *Field visualization.* To more clearly depict flow directions, elements can be assigned ramp textures to increase visual effects as in (a). By mapping elongated elements onto mesh surfaces (b), cross fields over surfaces can be visualized. Like [56], to visualize 3D tensor fields, anisotropic elements can be used to indicate the orientations of field axes (c).

Visualization and Computer Graphics 18, 10 (2012), 1717–1730.

[6] Weikai Chen, Yuexin Ma, Sylvain Lefebvre, Shiqing Xin, Jonàs Martínez, and wenping wang. 2017. Fabricable Tile Decors. *ACM Trans. Graph.* 36, 6, Article 175 (2017), 15 pages.

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

[8] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational Shape Approximation. *ACM Trans. Graph.* 23, 3 (2004), 905–914.

[9] Teela Cunningham. 2015. How to Create Seamless Patterns in Illustrator. (2015). `https://www.youtube.com/watch?v=ITRZ75OKrG0`.

[10] Ketan Dalal, Allison W. Klein, Yunjun Liu, and Kaleigh Smith. 2006. A Spectral Approach to NPR Packing. In *NPAR '06*. 71–78.

[11] Timothy Davison, Faramarz Samavati, and Christian Jacob. 2019. Interactive example-palettes for discrete element texture synthesis. *Computers & Graphics* 78 (2019), 23 – 36.

[12] Song-Pei Du, Shi-Min Hu, and Ralph R. Martin. 2013. Semiregular Solid Texturing from 2D Image Exemplars. *IEEE Transactions on Visualization and Computer Graphics* 19, 3 (2013), 460–469.

[13] Jérémie Dumas, Jonàs Martínez, Sylvain Lefebvre, and Li-Yi Wei. 2018. Printable Aggregate Elements. *CoRR* abs/1811.02626 (2018).

[14] Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. 2015. WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *ACM Trans. Graph.* 34, 4, Article 106 (2015), 11 pages.

[15] Freepik. 2019. Graphic resources for everyone. (2019). `https://www.freepik.com/`.

[16] Leonhard Frehse. 2018. AutoModeller Pro. (2018). `http://www.automodeller.com/`.

[17] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 2007. 3D Collage: Expressive Non-realistic Modeling. In *NPAR '07*. 7–14.

[18] Clovis Gay. 2016a. PhysX Painter. (2016). `http://www.scriptspot.com/3ds-max/scripts/physx-painter`.

[19] Clovis Gay. 2016b. PhysX Painter Teaser. (2016). `https://vimeo.com/162046605`.

[20] Eric Guérin, Eric Galin, François Grosbellet, Adrien Peytavie, and Jean-David Génevaux. 2016. Efficient modeling of entangled details for natural scenes. *Computer Graphics Forum* 35, 7 (2016), 257–267.

[21] Alejo Hausner. 2001. Simulating Decorative Mosaics. In *SIGGRAPH '01*. 573–580.

[22] Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *SIGGRAPH '00*. 517–526.

[23] Chen-Yuan Hsu, Li-Yi Wei, Lihua You, and Jian Jun Zhang. 2018. Brushing Element Fields. In *SIGGRAPH Asia 2018 Technical Briefs (SA '18)*. Article 6, 4 pages.

[24] Shu-Wei Hsu and John Keyser. 2010. Piles of Objects. *ACM Trans. Graph.* 29, 6, Article 155 (2010), 6 pages.

[25] Shu-Wei Hsu and John Keyser. 2012. Automated Constraint Placement to Maintain Pile Shape. *ACM Trans. Graph.* 31, 6, Article 150 (2012), 6 pages.

[26] Wenchao Hu, Zhonggui Chen, Hao Pan, Yizhou Yu, Eitan Grinspun, and Wenping Wang. 2016. Surface Mosaic Synthesis with Irregular Tiles. *IEEE Transactions on Visualization and Computer Graphics* 22, 3 (2016), 1302–1313.

[27] Hua Huang, Lei Zhang, and Hong-Chao Zhang. 2011. Arcimboldo-like Collage Using Internet Images. *ACM Trans. Graph.* 30, 6, Article 155 (2011), 8 pages.

[28] Zhiyang Huang and Tao Ju. 2016. Extrinsically smooth direction fields. *Computers & Graphics* 58 (2016), 109–117.

[29] Zhe Huang, Jiang Wang, Hongbo Fu, and Rynson W. H. Lau. 2014. Structured Mechanical Collage. *IEEE Transactions on Visualization and Computer Graphics* 20, 7 (2014), 1076–1082.

[30] Takashi Ijiri, Radomír Mêch, Takeo Igarashi, and Gavin Miller. 2008. An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum* 27, 2 (2008), 429–436.

[31] Robert Jagnow, Julie Dorsey, and Holly Rushmeier. 2004. Stereological Techniques for Solid Textures. *ACM Trans. Graph.* 23, 3 (2004), 329–335.

[32] Evangelos Kalogerakis, Derek Nowrouzezahrai, Simon Breslav, and Aaron Hertzmann. 2012. Learning Hatching for Pen-and-ink Illustration of Surfaces. *ACM Trans. Graph.* 31, 1, Article 1 (2012), 17 pages.

[33] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *CHI '14*. 351–360.

[34] 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.

[35] Junhwan Kim and Fabio Pellacini. 2002. Jigsaw Image Mosaics. *ACM Trans. Graph.* 21, 3 (2002), 657–664.

[36] Yeojin Kim, Byungmoon Kim, and Young J. Kim. 2018. Dynamic Deep Octree for High-resolution Volumetric Painting in Virtual Reality. *Computer Graphics Forum* 37, 7 (2018), 179–190.

[37] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. 2007. Solid Texture Synthesis from 2D Exemplars. *ACM Trans. Graph.* 26, 3, Article 2 (2007).

[38] Kin Chung Kwan, Lok Tsun Sinn, Chu Han, Tien-Tsin Wong, and Chi-Wing Fu. 2016. Pyramid of Arclength Descriptor for Generating Collage of Shapes. *ACM Trans. Graph.* 35, 6, Article 229 (2016), 12 pages.

[39] Pierre-Edouard Landes, Bruno Galerne, and Thomas Hurtut. 2013. A Shape-aware Model for Discrete Texture Synthesis. In *EGSR '13*. Aire-la-Ville, Switzerland, Switzerland, 67–76.

[40] Moshe Levis. 2014. Tutorial on How to do a typography design of a woman's face in Photoshop CC. (2014). `https://www.youtube.com/watch?v=LcZFp1s1AQI`.

[41] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2015. Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization. *ACM Trans. Graph.* 35, 1, Article 8 (2015), 16 pages.

[42] Yuanyuan Li, Fan Bao, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. 2011. Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars. *IEEE Transactions on Visualization and Computer Graphics* 17, 2 (2011), 231–243.

[43] Hugo Loi, Thomas Hurtut, Romain Vergne, and Joëlle Thollot. 2013. Discrete Texture Design Using a Programmable Approach. In *SIGGRAPH '13 Talks*. Article 43, 1 pages.

[44] Hugo Loi, Thomas Hurtut, Romain Vergne, and Joelle Thollot. 2017. Programmable 2D Arrangements for Element Texture Design. *ACM Trans. Graph.* 36, 3, Article 27 (2017), 17 pages.

[45] Jingwan Lu, Connelly Barnes, Stephen DiVerdi, and Adam Finkelstein. 2013. RealBrush: Painting with Examples of Physical Media. *ACM Trans. Graph.* 32, 4, Article 117 (2013), 12 pages.

[46] 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.

[47] Jingwan Lu, Fisher Yu, Adam Finkelstein, and Stephen DiVerdi. 2012. HelpingHand: Example-based Stroke Stylization. *ACM Trans. Graph.* 31, 4, Article 46 (2012), 10 pages.

[48] M. Lukáč, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sýkora. 2015. Brushables: Example-based Edge-aware Directional Texture Painting. *Comput. Graph. Forum* 34, 7 (2015), 257–267.

[49] Michal Lukáč, Jakub Fišer, Jean-Charles Bazin, Ondřej Jamriška, Alexander Sorkine-Hornung, and Daniel Sýkora. 2013. Painting by Feature: Texture Boundaries for Example-based Image Creation. *ACM Trans. Graph.* 32, 4, Article 116 (2013), 8 pages.

[50] Chongyang Ma, Li-Yi Wei, Sylvain Lefebvre, and Xin Tong. 2013. Dynamic Element Textures. *ACM Trans. Graph.* 32, 4, Article 90 (2013), 10 pages.

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

[52] K. Madsen, H. B. Nielsen, and O. Tingleff. 2004. Methods for Non-Linear Least Squares Problems (2nd ed.). (2004). `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3215`.

[53] Ron Maharik, Mikhail Bessmeltsev, Alla Sheffer, Ariel Shamir, and Nathan Carr. 2011. Digital Micrography. *ACM Trans. Graph.* 30, 4, Article 100 (2011), 12 pages.

[54] Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-scale Modeling and Rendering of Granular Materials. *ACM Trans. Graph.* 34, 4, Article 49 (2015), 13 pages.

[55] Thomas Muller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novak. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Trans. Graph.* 35, 6 (2016).

[56] Jonathan Palacios, Lawrence Roy, Prashant Kumar, Chen-Yuan Hsu, Weikai Chen, Chongyang Ma, Li-Yi Wei, and Eugene Zhang. 2017. Tensor Field Design in Volumes. *ACM Trans. Graph.* 36, 6, Article 188 (2017), 15 pages.

[57] Jonathan Palacios and Eugene Zhang. 2007. Rotational Symmetry Field Design on Surfaces. *ACM Trans. Graph.* 26, 3, Article 55 (2007).

[58] Adrien Peytavie, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou. 2009. Procedural generation of rock piles using aperiodic tiling. *Computer Graphics Forum* 28, 7 (2009), 1801–1809.

[59] Charles Purdy. 2019. Make It, Sell It: Repeating Patterns in Adobe Illustrator. (2019). `https://create.adobe.com/2019/4/2/make_it_sell_it_repe.html`.

[60] Xuejie Qin and Yee-Hong Yang. 2007. Aura 3D Textures. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (2007), 379–389.

[61] Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. 2013. Interactive By-example Design of Artistic Packing Layouts. *ACM Trans. Graph.* 32, 6, Article 218 (2013), 7 pages.

[62] Lincoln Ritter, Wilmot Li, Brian Curless, Maneesh Agrawala, and David Salesin. 2006. Painting with Texture. In *EGSR '06*. 371–376.

[63] 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.

[64] Chris Rycroft. 2009. Voro++: A three-dimensional Voronoi cell library in C++. (2009). `http://math.lbl.gov/voro++/`.

[65] K. Sakurai and K. Miyata. 2014. Modelling of Non-Periodic Aggregates Having a Pile Structure. *Comput. Graph. Forum* 33, 1 (2014), 190–198.

[66] Christian Santoni and Fabio Pellacini. 2016. gTangle: A Grammar for the Procedural Generation of Tangle Patterns. *ACM Trans. Graph.* 35, 6, Article 182 (2016), 11 pages.

[67] Reza Saputra, Craig Kaplan, and Paul Asente. 2018. RepulsionPak: Deformation-Driven Element Packing with Repulsion Forces. In *GI 2018*. 10 – 17.

[68] Reza Adhitya Saputra, Craig S. Kaplan, Paul Asente, and Radomír Měch. 2017. FLOWPAK: Flow-based Ornamental Element Packing. In *GI '17*. 8–15.

[69] Alexander Schiftner, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2009. Packing Circles and Spheres on Surfaces. *ACM Trans. Graph.* 28, 5, Article 139 (2009), 8 pages.

[70] D. Schroeder, D. Coffey, and D. Keefe. 2010. Drawing with the Flow: A Sketch-based Interface for Illustrative Visualization of 2D Vector Fields. In *SBIM '10*. 49–56.

[71] Martin Schwarz, Tobias Isenberg, Katherine Mason, and Sheelagh Carpendale. 2007. Modeling with Rendering Primitives: An Interactive Non-photorealistic Canvas. In *NPAR '07*. 15–22.

[72] Maria Shugrina, Jingwan Lu, and Stephen Diverdi. 2017. Playful Palette: An Interactive Parametric Color Mixer for Artists. *ACM Trans. Graph.* 36, 4, Article 61 (2017), 10 pages.

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

[74] Kenshi Takayama, Makoto Okabe, Takashi Ijiri, and Takeo Igarashi. 2008. Lapped Solid Textures: Filling a Model with Anisotropic Textures. *ACM Trans. Graph.* 27, 3, Article 53 (2008), 9 pages.

[75] Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: Shape Approximation Using Spherical Quadric Error Metrics. *ACM Trans. Graph.* 32, 6, Article 178 (2013), 12 pages.

[76] VideoLot. 2016. Adobe Illustrator Cs6 | Typography Portrait | Bruno Mars. (2016). `https://www.youtube.com/watch?v=_kdhB-8tNeM`.

[77] Rui Wang, Kun Zhou, John Snyder, Xinguo Liu, Hujun Bao, Qunsheng Peng, and Baining Guo. 2006. Variational Sphere Set Approximation for Solid Objects. *Vis. Comput.* 22, 9 (2006), 612–621.

[78] Jun Xing, Rubaiat Habib Kazi, Tovi Grossman, Li-Yi Wei, Jos Stam, and George Fitzmaurice. 2016. Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics. In *UIST '16*. 755–766.

[79] Jie Xu and Craig S. Kaplan. 2007. Calligraphic Packing. In *GI '07*. 43–50.

[80] Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing Structurally-sound Ornamental Curve Networks. *ACM Trans. Graph.* 35, 4, Article 99 (2016), 10 pages.

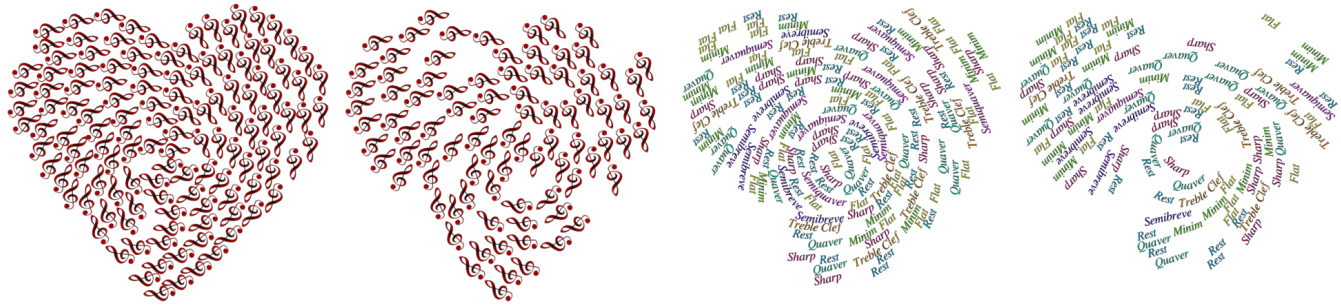
[81] Guo-Xin Zhang, Song-Pei Du, Yu-Kun Lai, Tianyun Ni, and Shi-Min Hu. 2011. Sketch guided solid texturing. *Graphical Models* 73, 3 (2011), 59–73.

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

[83] Changqing Zou, Junjie Cao, Warunika Ranaweera, Ibraheem Alhashim, Ping Tan, Alla Sheffer, and Hao Zhang. 2016. Legible Compact Calligrams. *ACM Trans. Graph.* 35, 4, Article 122 (2016), 12 pages.

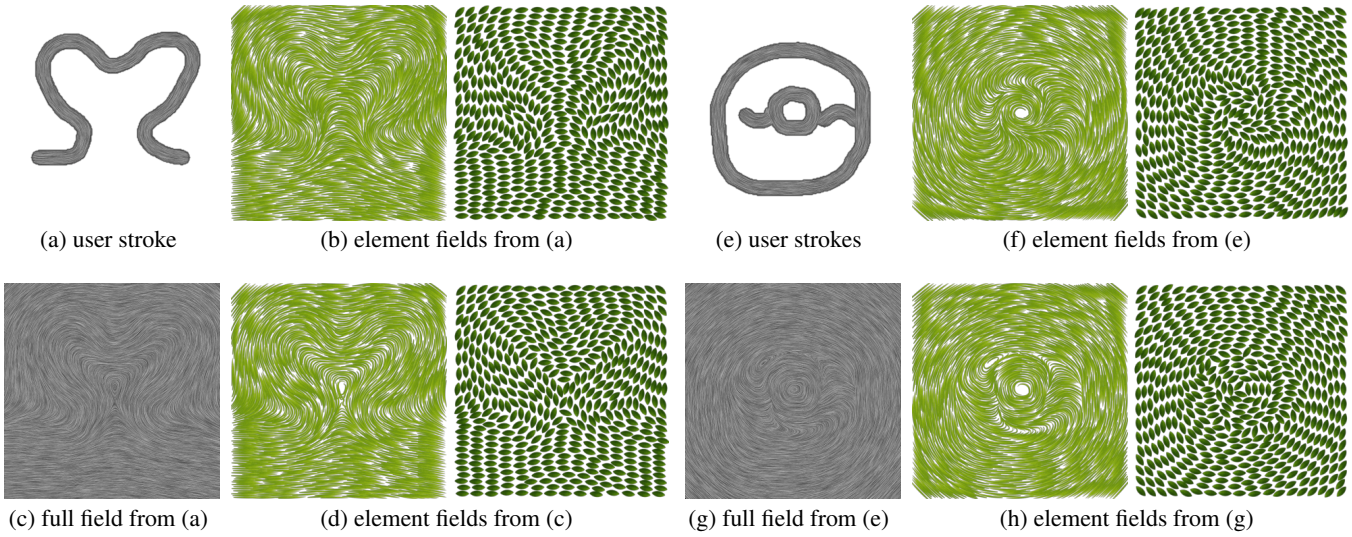
(a) with both E_d and E_k (d) with both E_d and E_k (g) with both E_d and E_k (j) with both E_d and E_k

(b) with E_d, without E_k (e) with E_d, without E_k (h) with E_d, without E_k (k) with E_d, without E_k



(c) without E_d, with E_k (f) without E_d, with E_k (i) without E_d, with E_k (l) without E_d, with E_k

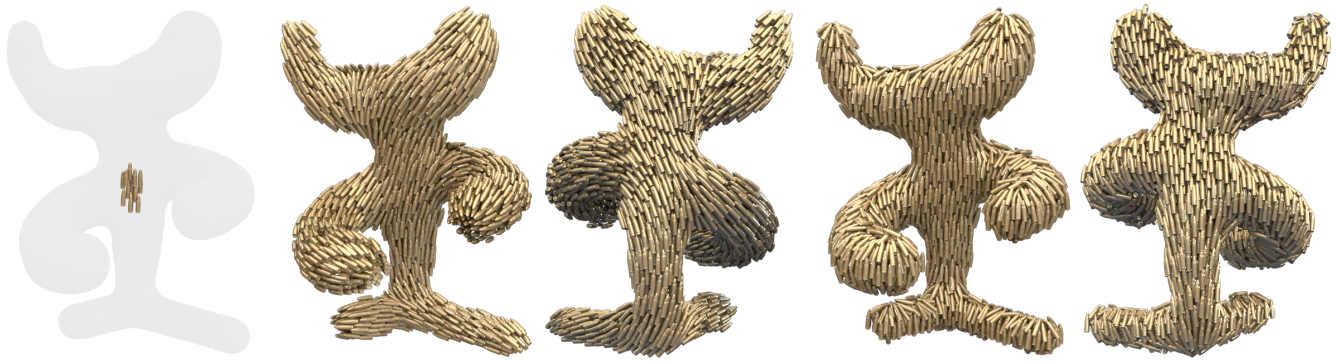
Figure 20: *Ablation study.* To demonstrate the effect of our objective functions, we densely and sparsely distribute rigid and deformable elements respectively and study the sample distribution E_d term and the conflict check E_k term in different cases. Without E_k, anisotropic elements might overlap when the elements are distributed densely as in (b) and (h). Without E_d, the elements do not overlap but can be distributed unevenly as in (c), (f), (i) and (l).



(a) user stroke (b) element fields from (a) (e) user strokes (f) element fields from (e)

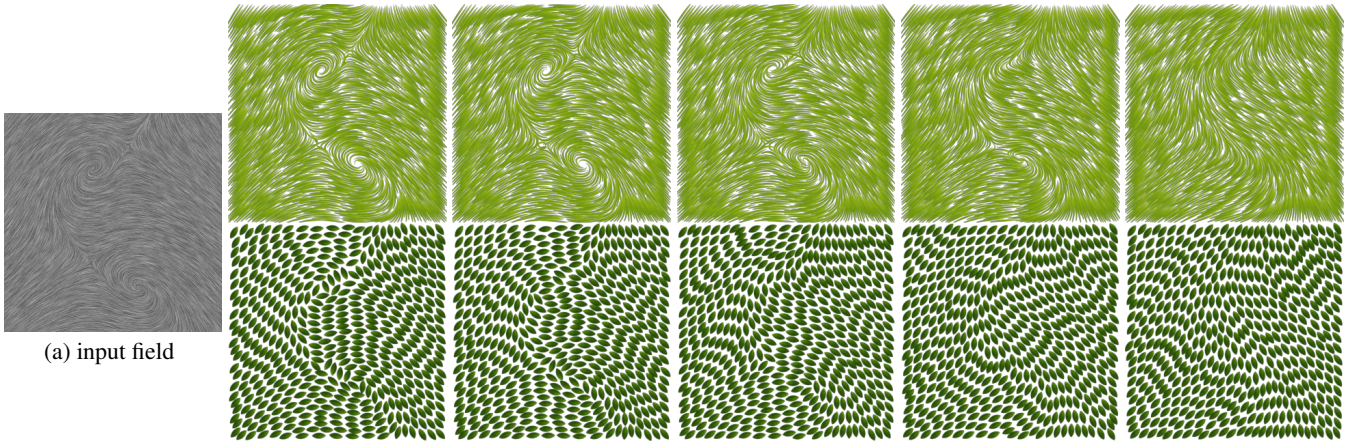
(c) full field from (a) (d) element fields from (c) (g) full field from (e) (h) element fields from (g)

Figure 21: *Field comparison.* Our method can directly complete *element fields* (b) and (f) from partially user-specified strokes (a) and (e) in a one-step automatic optimization process. As compared with (d) and (h) computed via a two-step process, which needs to first produce full input fields (c) and (g) from (a) and (e) respectively via Laplacian interpolation and forces elements to follow, our one-step process can more smoothly orient the elements all over the domain.



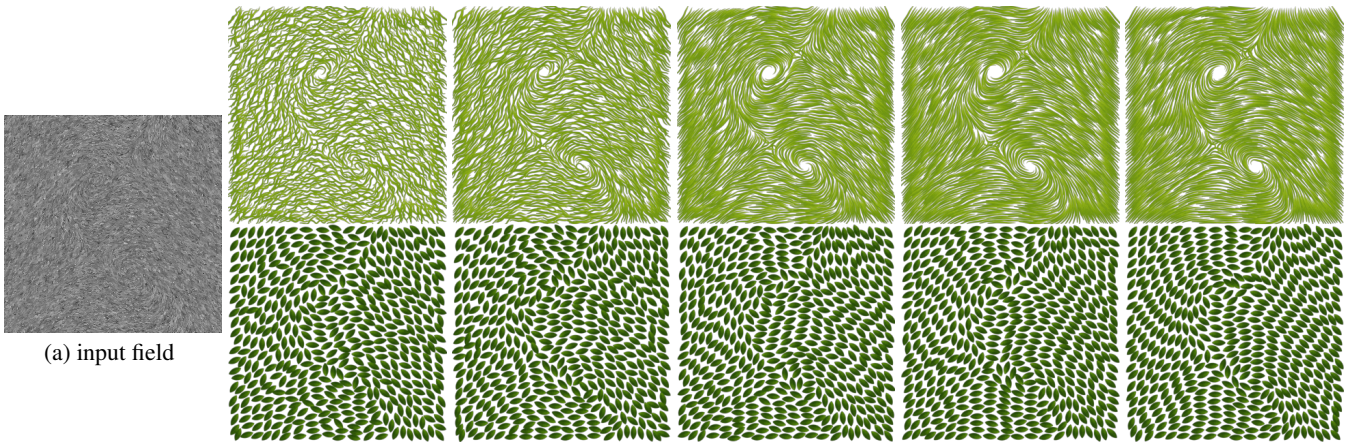
(a) user stroke (b) one-step process in different views (c) two-step process in different views

Figure 22: *Domain boundary normals.* Given a small user stroke (a), our one-step process can also combine the domain boundary normals as another partial input field and generate smoother results (b) than the two-step process (c) which can cause discontinuous element alignments. Please refer to the accompanying video for more views.



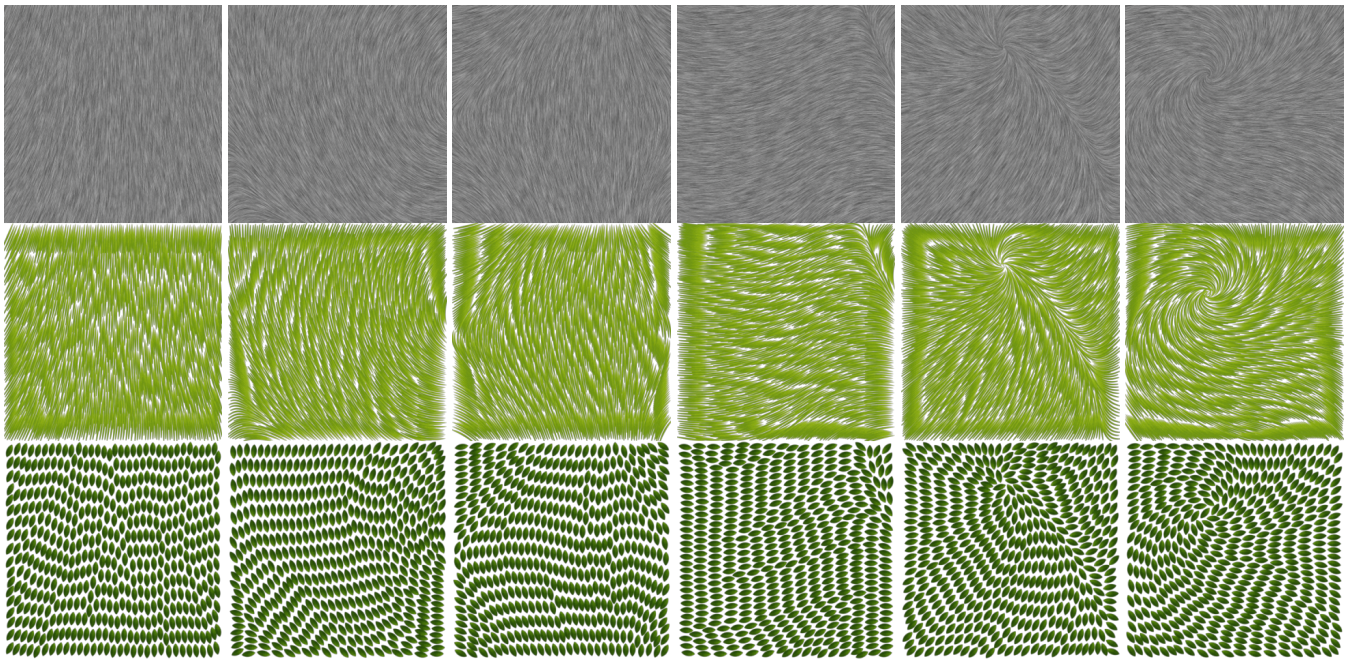
(a) input field (b) $\alpha = 0$ (c) $\alpha = 5$ (d) $\alpha = 10$ (e) $\alpha = 15$ (f) $\alpha = 20$

Figure 23: *Singularity handling.* Given an input field with singularities (a), by tuning α to increase the effect of field continuity, our method can directly hide the undesired singularities without changing the underlying field. Note that the orientations of elements near the boundary can still remain almost consistent.



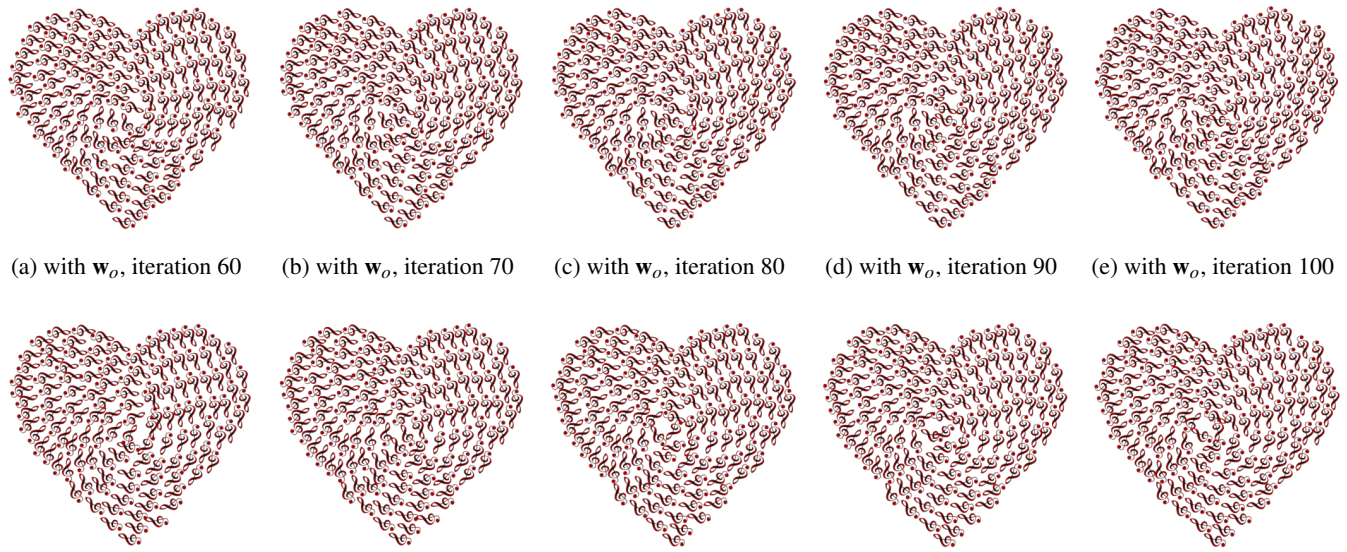
(a) input field (b) $\alpha = 0$ (c) $\alpha = 1$ (d) $\alpha = 2$ (e) $\alpha = 3$ (f) $\alpha = 4$

Figure 24: *Chaotic input fields.* The input field (a) is generated by adding random noise to the field in Figure 23a. By increasing α to enhance the effect of field continuity as in Figure 23, the noise effect can be reduced accordingly as well.



(a) frame 0 (b) frame 24 (c) frame 48 (d) frame 72 (e) frame 96 (f) frame 120

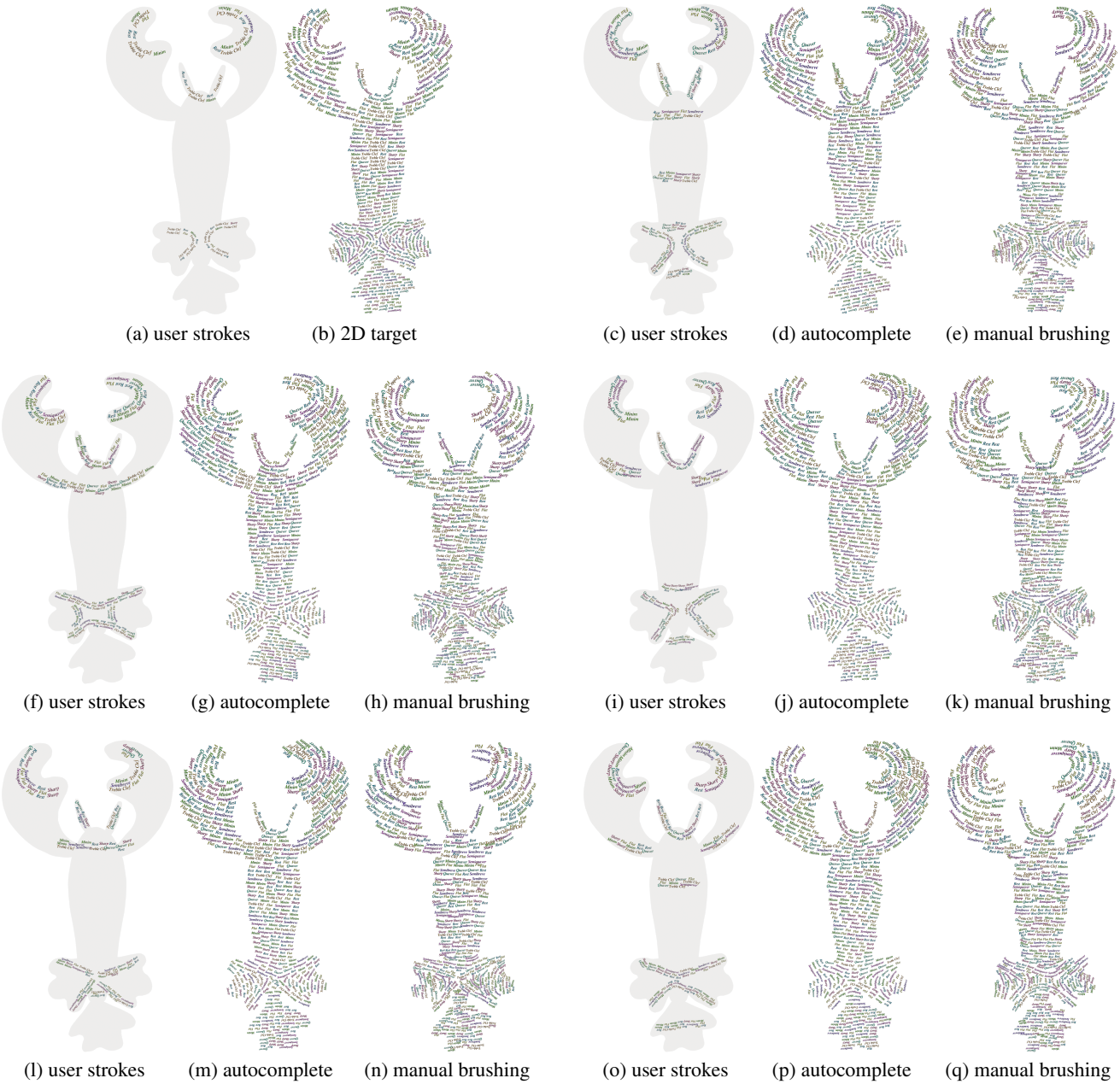
Figure 25: *Dynamic input fields.* Our formulation can also tackle dynamic input fields. The process is that we generate the output for the first frame and take this output as the initialization for the next frame. Please refer to the supplementary video for animations. In the video, the outputs are produced by 100 iterations for the first frame and 20 iterations for the rest of frames respectively. We plan to more fully explore dynamic fields and believe this feature can be further extended for motion graphics and element animations as future work.



(a) with $\mathbf{w}_o$, iteration 60 (b) with $\mathbf{w}_o$, iteration 70 (c) with $\mathbf{w}_o$, iteration 80 (d) with $\mathbf{w}_o$, iteration 90 (e) with $\mathbf{w}_o$, iteration 100

(f) without $\mathbf{w}_o$, iteration 60 (g) without $\mathbf{w}_o$, iteration 70 (h) without $\mathbf{w}_o$, iteration 80 (i) without $\mathbf{w}_o$, iteration 90 (j) without $\mathbf{w}_o$, iteration 100

Figure 26: *The $\mathbf{w}_o$ effect for the field continuity E_c term.* It can be seen that with $\mathbf{w}_o$, the elements near the singularity (near the heart's center) can be better stabilized after 70 iterations, whereas without $\mathbf{w}_o$, the element distributions around the singularity can still remain unsteady throughout. The supplementary video is also available for illustration.

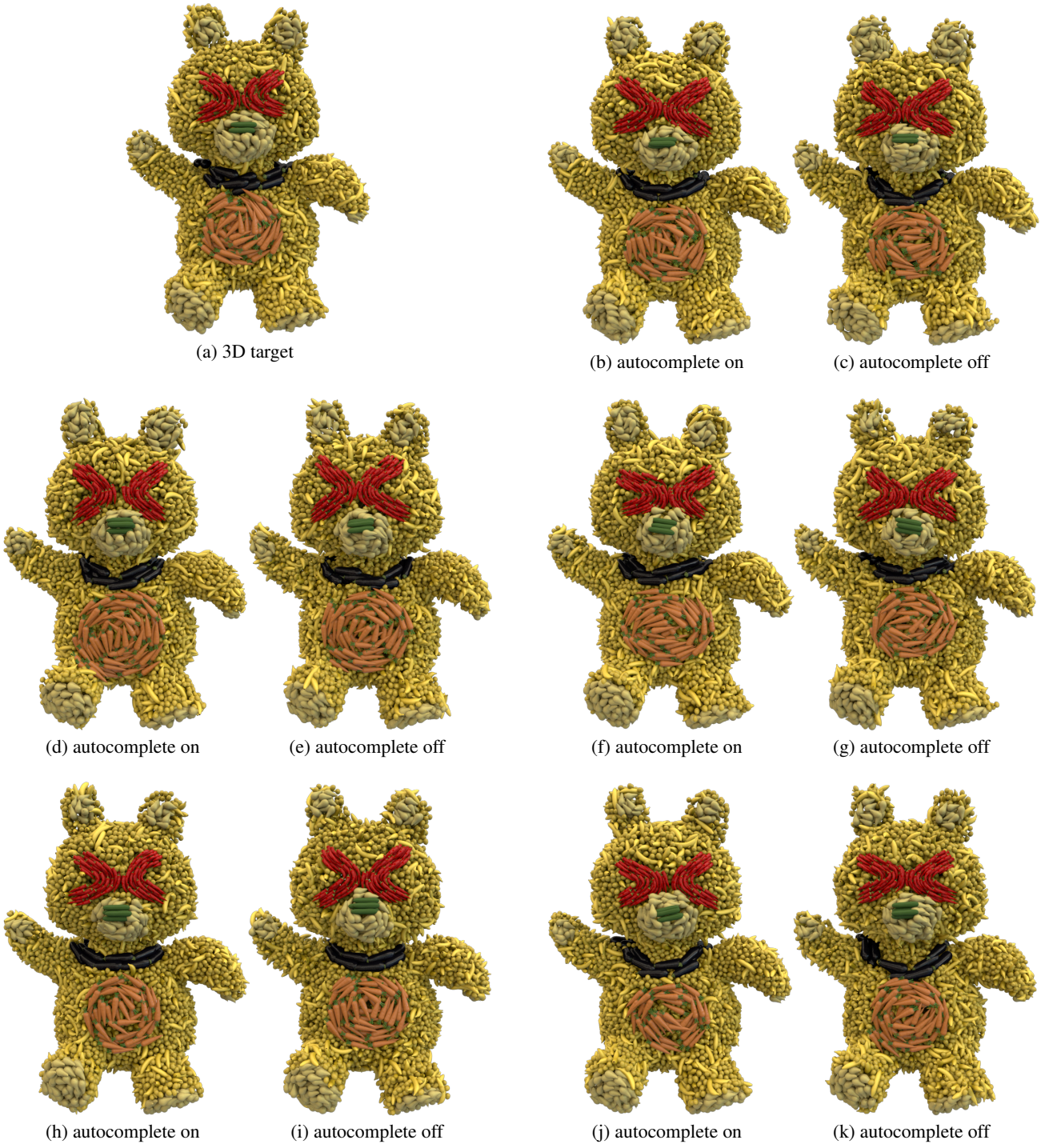


(a) user strokes (b) 2D target (c) user strokes (d) autocomplete (e) manual brushing

(f) user strokes (g) autocomplete (h) manual brushing (i) user strokes (j) autocomplete (k) manual brushing

(l) user strokes (m) autocomplete (n) manual brushing (o) user strokes (p) autocomplete (q) manual brushing

Figure 27: *Sample user study outputs for the 2D target.* Each group contains the partially specified user strokes for autocomplete, the autocomplete result computed from the user strokes, and the output created by fully manual brushing.



(a) 3D target (b) autocomplete on (c) autocomplete off

(d) autocomplete on (e) autocomplete off (f) autocomplete on (g) autocomplete off

(h) autocomplete on (i) autocomplete off (j) autocomplete on (k) autocomplete off

Figure 28: *Sample user study outputs for the 3D target.* Each group contains the autocomplete result with partially manual brushing and the output created by fully manual brushing. With the autocomplete mode as in (b), (d), (f), (h) and (j), the lemons and bananas were automatically synthesized by our system within the target. Without the autocomplete mode as in (c), (e), (g), (i) and (k), the lemons and bananas were interactively brushed by the participants over the target.