Autocomplete Element Fields and Interactive Synthesis System Development for Aggregate Applications

CHEN-YUAN HSU

A thesis submitted in partial fulfillment of the requirements of Bournemouth University for the degree of

Doctor of Philosophy

National Center for Computer Animation
Bournemouth University

July, 2020

Copyright Statement

Abstract

Aggregate elements are ubiquitous in natural and man-made objects and have played an important role in the application of graphics, design and visualization. However, to efficiently arrange these aggregate elements with varying anisotropy and deformability still remains challenging, in particular in 3D environments. To overcome such a thorny issue, we thus introduce autocomplete element fields, including an element distribution formulation that can effectively cope with diverse output compositions with controllable element distributions in high production standard and efficiency as well as an element field formulation that can smoothly orient all the synthesized elements following given inputs, such as scalar or direction fields. The proposed formulations can not only properly synthesize distinct types of aggregate elements across various domain spaces without incorporating any extra process but also directly compute complete element fields from partial specifications without requiring fully specified inputs in any algorithmic step. Furthermore, in order to reduce input workload and enhance output quality for better usability and interactivity, we further develop an interactive synthesis system, centered on the idea of our autocomplete element fields, to facilitate the creation of element aggregations within different output domains. Analogous to conventional painting workflows, through a palette-based brushing interface, users can interactively mix and place a few aggregate elements over a brushing canvas and let our system automatically populate more aggregate elements with intended orientations and scales for the rest of outcome. The developed system can empower the users to iteratively design a variety of novel mixtures with reduced workload and enhanced quality under an intuitive and user-friendly brushing workflow without the necessity of a great deal of manual labor or technical expertise. We validate our prototype system with a pilot user study and exhibit its application in 2D graphic design, 3D surface collage, and 3D aggregate modeling.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I deeply appreciate my family, especially my parents, for their support and encouragement to me. During my study for a doctoral degree, they always gave me full confidence and faith and made sure that I do not have anything to worry about and can concentrate on doing what I want.

I am very thankful to Dr. Li-Yi Wei since my research project was mainly guided by him. Dr. Li-Yi Wei gave me many helpful suggestions about the direction of my research and assisted me in developing algorithms from the beginning. With his valuable advice, I was able to step by step cultivate my research capability and successfully publish high-quality papers in top peer-reviewed conferences.

I would also like to give my deepest appreciation to my supervisors Professor Lihua You and Professor Jian Jun Zhang for their academic help. With their great assistance, I could smoothly achieve my research project in a very flexible process.

I am also grateful to Bournemouth University for offering me a comfortable research environment. The university was able to sufficiently supply me with not only essential study resources but also technical services, so I could make steady and productive progress on my research project without disruption.

Finally, I would like to thank everyone I met in Bournemouth. You brought me a memorable experience in my life. Wish you all have a good luck. Thank you very much.

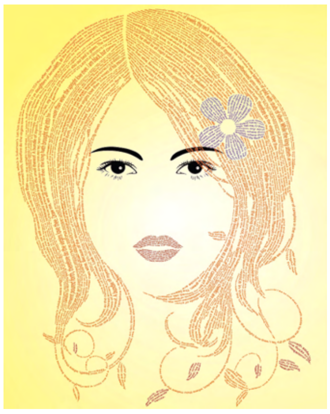
Declaration

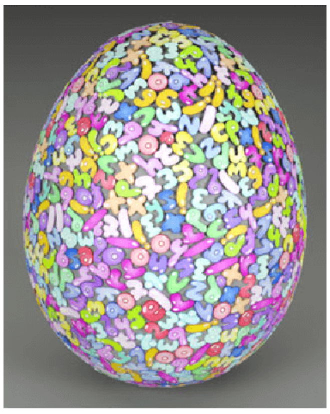
This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

1 Introduction

1.1 Overview

There can be no doubt that innovative interaction designs and techniques are important and worth investigating, as novel and creative interaction modalities are able to facilitate possible interface exploration and inspire future device development. Over the past decades, a large number of interactive technologies and systems have already been widely proposed in various research areas, such as computer graphics and human–computer interaction. It can also be seen that a certain portion of these interactive technologies and systems have been successfully integrated into industrial production or commercial products. Obviously, to develop an applicable algorithm along with an intuitive and user-friendly interface for practical applications can be a popular and meaningful research topic for scientific experts and software engineers, and there also exist several top peer-reviewed conferences, such as SIGGRAPH, SIGGRAPH Asia, CHI and UIST, for researchers to publish their works related to this topic. Since a well-developed framework for pragmatic usage can not only effectively assist both novice users and professional technicians in the simplification of production workflows and the creation of sophisticated designs but also significantly improve productive efficiency as well as resulting quality, it can be crucial and essential to explore potential avenues and study usable schemes for unaddressed usability issues or challenging interactivity tasks. Therefore, in terms of the research presented in this thesis, the goal is to construct such a well-developed framework, which includes relevant formulations and system development, for interactive synthesis of discrete element textures for aggregate applications across 2D and 3D output domains, and the aggregate applications which we focus on here are the applications related to packing or distributing a certain number of aggregate elements following design intention into a target region with a specific shape for desired artworks, such as graphic design, artistic collage, and aggregate modeling, as exemplified in Figure 1.1.


(a) *Maharik* et al. *(2011)*


(b) *Hu* et al. *(2016)*

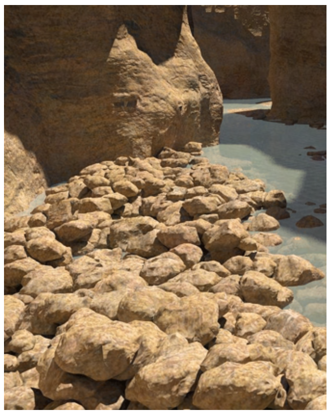

(c) *Peytavie* et al. *(2009)*

Figure 1.1: The examples about aggregate applications. *Distinct types of aggregate elements, such as letters in (a), numbers in (b), and rocks in (c), can be respectively placed over 2D image planes for graphic design (a), 3D object surfaces for artistic collage (b), and 3D environmental scenes for aggregate modeling (c).*

Like these individual letters, numbers and rocks shown in Figure 1.1, aggregate elements are omnipresent in man-made artistic works and can be observed around natural environments as well. Since well-organized element aggregations can be broadly exploited by artists and designers in a variety of applications, both practicable solutions and easy-to-use interfaces for making such element aggregations are popular and still remain in high demand. There are a great number of relative publications and academic literatures presented for different research purposes, such as rendering (Meng *et al.* 2015; Muller *et al.* 2016), modeling (Ma *et al.* 2011; Sakurai and Miyata 2014; Roveri *et al.* 2015), simulation (Hsu and Keyser 2010, 2012), interaction (Kazi *et al.* 2012, 2014), and design (Kim and Pellacini 2002; Gal *et al.* 2007; Zou *et al.* 2016; Kwan *et al.* 2016). Moreover, in relevant industry sectors, several notable software packages, such as Adobe Photoshop, Adobe Illustrator, Autodesk Maya, and Autodesk AutoCAD, have also been developed and released to offer designers and animators customized user interfaces to properly deal with specific aggregate elements for illustration designs and animation productions through either manual placement or physically-based simulation. However, to reasonably transform conceptual ideas into concrete design outcomes, ordinary users still can not only require adequate knowledge of crafting procedures and artistic skills in advance but also spend plenty of time and energy on production. It can be noted that there is a lack of efficient algorithms and general user interfaces for interactive authoring of aggregate elements with varying anisotropy and deformability, especially within 3D output domains (Cho *et al.* 2007).

It is clear that manual placement of aggregate elements can allow users to access full authoring freedom, but the users can demand to acquire essential understanding of technical expertise and make heavy efforts to repetitively adjust the distributions, orientations and scales of individual aggregate elements for final design outcomes in a tedious and energy-consuming procedure. While automatic batch computation in preceding approaches such as (Ma *et al.* 2011; Sakurai and Miyata 2014; Saputra *et al.* 2018) can apply in 2D or 3D output domains, it cannot provide sufficient authoring freedom and desirable interactivity for users due to algorithmic limitations or expensive computation cost. Even though there exist a handful of methods that can automate interactive authoring of 2D sketched elements for pictorial animations (e.g. (Kazi *et al.* 2012, 2014)), creators still need to individually place each brush stroke or position element collections step by step all over the output domain. To efficiently distribute aggregate elements with anisotropic shapes and various sizes across various domain spaces can be significantly complicated, and the degree of difficulty of distributing such aggregate elements can often notably increase along with the amount of element anisotropy and the variety of element deformability. Consequently, it can be observed that there have been a lack of good solutions that can supply desirable interactivity without compromising user controls so far.

Furthermore, as exemplified in Figure 1.2, in order to better orient distinct types of aggregate elements for desired element arrangements, existing element synthesis practices, such as (Maharik *et al.* 2011; Li *et al.* 2011; Ma *et al.* 2011; Saputra *et al.* 2017), often have to incorporate full scalar or direction fields into their algorithms as part of optimization, but it can be difficult or inconvenient for ordinary users to acquire such full scalar or direction fields either manually or automatically over arbitrary output domains, in particular in 3D environments. The full input fields, even when automatically computed by existing field design algorithms, may also force undesirable artifacts, such as misaligned aggregate elements near singular points which can be reduced via singularity manipulation but cannot be entirely avoided (Zhang *et al.* 2006; Palacios *et al.* 2017), and while separately employing alternative auxiliary production platforms or application softwares for the generation of fully specified input fields, this extra process can significantly break the natural artist workflow as well. Hence, being able to let users handily create compelling element aggregations that can accordingly match certain user specifications (e.g. scalar or direction fields) in high output standard via a general design procedure without sacrificing performance is more challenging and still remains an unaddressed problem.

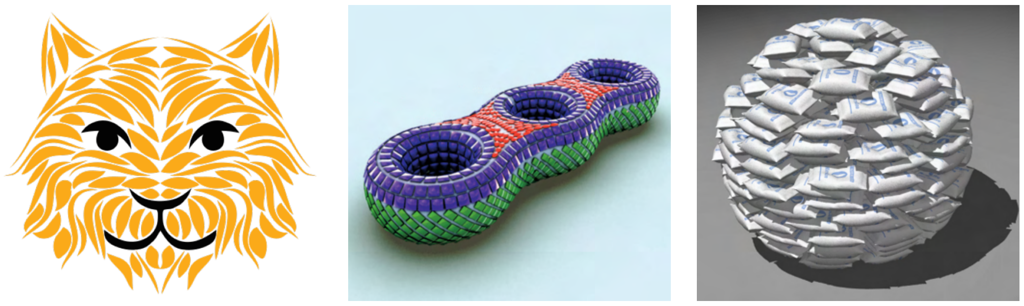

(a) *Saputra* et al. *(2017)* **(b)** *Li* et al. *(2011)* **(c)** *Ma* et al. *(2011)*

Figure 1.2: The examples about aggregate applications incorporating input fields. *Aggregate elements with anisotropic shapes and various sizes are oriented to follow certain directions for desired element arrangements.*

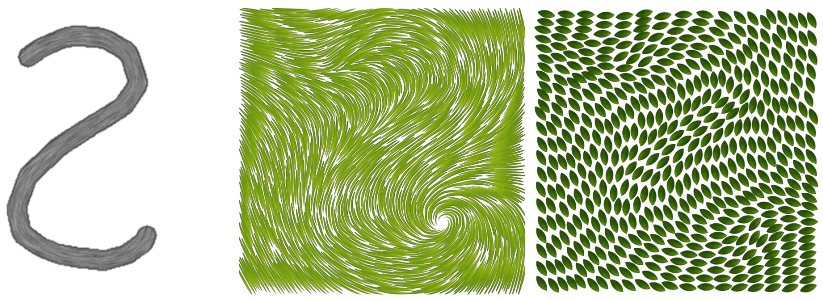
In addition to the user specifications, another challenging task that users can face for the creation of element aggregations is the controllability of freely mixing specific aggregate elements for diverse output compositions. Since preceding approaches lack the flexibility of directly manipulating the distribution of aggregate elements and do not take user-interactive element mixing into consideration either, it can be particularly tricky for creators to optionally determine the entire element arrangements within arbitrary mixtures. As a result, in order to effectively sort out the aforementioned issues and simultaneously address the requirement of both usability and interactivity, the ideal solution for element synthesis has to be efficient and versatile enough to flexibly deal with aggregate elements with general shapes, distributions, and alignments over different output domains, and the interactive synthesis system should also be user-friendly and intuitive enough for users to iteratively design and explore a variety of output formations with desirable appearances for practical applications without requiring a great deal of manual labor or technical expertise from the users.

1.2 Objectives

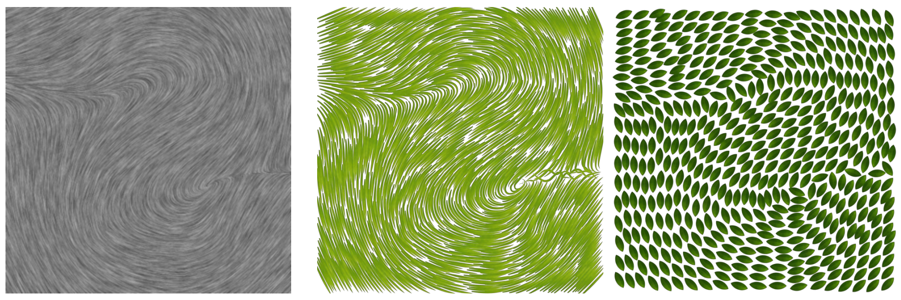
As mentioned in (Reinert *et al.* 2013), to optimally pack an arbitrary set of aggregate elements, especially those aggregate elements with varying anisotropy and deformability, into a target output domain can be an NP-hard problem (bin packing). In most instances, it can be time-consuming to directly compute an accurate packing layout for arbitrary mixtures consisting of distinct types of aggregate elements as described in (Kwan *et al.* 2016; Saputra *et al.* 2018), and the computational complexity of optimizing such a packing layout can further increase by the amount of element anisotropy, the variety of element deformability, the output domain's size, shape and dimension, as well as given user specifications. To overcome this challenging problem regarding element arrangements for better productivity, our solution is designed to balance a tradeoff between synthesis quality and performance so that the functionality of user interaction can be adequately carried out. Therefore, we devise an element representation that can properly characterize aggregate elements with varying anisotropy and deformability by a set of element samples and graphs to reasonably reduce inter-element penetrations and strengthen intra-element connections and specifically propose autocomplete element fields, which include an element distribution formulation that can effectively cope with diverse output compositions with controllable element distributions across various domain spaces as well as an element field formulation that can smoothly orient all the synthesized elements following given specifications such as scalar or direction fields.

Our element distribution formulation, which contains a data-driven method (Hsu *et al.* 2018) and a procedural approach (Hsu *et al.* 2020), concentrates on effectively distributing aggregate elements with anisotropic shapes and various sizes in high synthesis quality and performance within the target output domain. Analogous to previous example-based algorithms (Ijiri *et al.* 2008; Ma *et al.* 2011; AlMeraj *et al.* 2013; Landes *et al.* 2013; Roveri *et al.* 2015; Guérin *et al.* 2016; Davison *et al.* 2019), our data-driven method can appropriately synthesize the aggregate elements on the basis of given element distribution references captured in the input element exemplars, but unlike these preceding example-based algorithms, which cannot allow to freely mix desired aggregate elements from multiple input element exemplars, our data-driven method can directly mix distinct types of aggregate elements from different input element exemplars to arbitrarily form a variety of novel mixtures. Nevertheless, since the data-driven method can demand to individually prepare the input element exemplars in advance as a pre-process, the preparation of such input element exemplars can significantly increase user workload, and a higher level of technical expertise can also be required to make relevant input element exemplars for corresponding element distribution references. In order to further streamline work procedures, we thus present the procedural approach that can not only fulfill faster computation performance for interactive speed but also sufficiently provide more functionality, such as the manipulation of user-specified element distributions, without heavy implementation workload. Similar to previous procedural techniques (e.g. (Kwan *et al.* 2016; Saputra *et al.* 2018)), our procedural approach can also directly produce an assortment of mixtures composed of specific aggregate elements without the need to prepare any input element exemplars in advance, but unlike these prior procedural techniques, which are not able to flexibly manipulate the aggregate elements' distribution, our procedural approach can versatilely deal with diverse output compositions with dense, sparse or even spatially varying element distributions under the same element synthesis process. On the basis of our element distribution formulation, aggregate elements with anisotropic shapes and various sizes can be well synthesized to create compelling element aggregations in satisfactory output standard and efficiency within 2D planes, 3D surfaces and 3D volumes without needing to incorporate any additional solvers (e.g. physically-based simulation).

Furthermore, while taking user specifications (i.e. scalar or direction fields) into consideration, to reasonably match individual aggregate elements with the given user specifications, existing element synthesis practices predominantly take a two-step process, which first requires to preprocess a full input field and then forces all the synthesized elements to follow correspondingly. However, such a two-step process can not only break the natural artist workflow but also may result in undesirable artifacts (i.e. misaligned aggregate elements around singular points). In order to better orient these aggregate elements with varying anisotropy and deformability for desired element arrangements, our element field formulation focuses on adequately matching all the aggregate elements with given inputs and adaptively smoothing the overall element arrangements depending upon their inter- and intra-element relationships at the same time through a one-step automatic optimization process. As demonstrated in Figure 1.3, through directly optimizing each aggregate element with the user-specified input together, the synthesized elements can be more smoothly oriented with the intended element orientations and scales via the one-step automatic optimization process, whereas the final outcomes created via the two-step process might not either properly reach the expected output standard or well reflect the original user intention. Our element field formulation can automatically compute complete element fields from partial user specifications (e.g. manual brush strokes or presimulated orientation fields) without the necessity of fully specified input fields in any algorithmic step and even support tunable controllability, such as the manipulation of the topology of element fields, without requiring any additional field preprocessing (e.g. field smoothing and interpolation). In consequence, by means of our one-step automatic optimization process, arbitrary mixtures consisting of specific aggregate elements with certain orientations and scales can be entirely constructed in a productive and handy manner without breaking the natural artist workflow or compromising the output level of production.



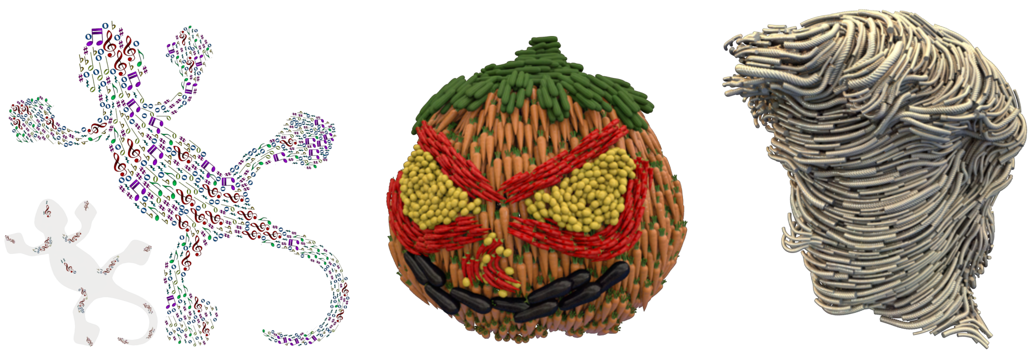
(a) *partially specified input* **(b)** *element fields computed from (a)*



(c) *full input field from (a)* **(d)** *element fields computed from (c)*

Figure 1.3: The element fields. *Given a partially specified input (a), our one-step automatic optimization process (b) can directly compute smoother element fields and more adaptively arrange distinct types of aggregate elements (i.e. deformable grasses and rigid leaves) than the two-step process (d), which needs to process a full input field (c) from the partial input (a) in advance via Laplacian smoothing (Zhang* et al. *2006; Palacios* et al. *2017) and then compels all the synthesized elements to follow accordingly.*

In addition to the formulations described above, to reduce input workload and enhance output quality for better usability and interactivity, we further develop an interactive synthesis system with our proposed formulations to facilitate the creation of element aggregations for practical applications, such as graphic design, artistic collage and aggregate modeling, across 2D or 3D output domains. As exemplified in Figure 1.4, the developed system can enable users to interactively arrange distinct types of aggregate elements over given output domains, automatically optimize the distributions, orientations and scales of all the synthesized elements according to given user specifications (e.g. partially user-specified brush strokes), and directly compute complete and smooth outcomes that can appropriately reflect user intention. With the interactive synthesis system, assorted output formations with desirable appearances like Figure 1.4a can be correspondingly generated in accordance with personal preferences, whereas it can be fairly difficult if not impossible for users to accomplish such output formations through preceding approaches.



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

Figure 1.4: Autocomplete element synthesis following partial user specifications. *Our interactive synthesis 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 graphic design (a), artistic collage (b), and aggregate modeling (c). The developed system can automatically optimize the entire element distributions, orientations and scales based on the partially user-specified brush strokes (a) (inset), enable users to interactively arrange the element collections over the output domain (b), and directly compute the volumetric output (c) from a given surface direction field (see Figure 3.6).*

Analogous to traditional painting workflows, through a palette-based brushing interface, users can freely select single or even mix multiple aggregate elements from an element palette, directly brush the selected elements over a given canvas (e.g. 2D planar region or 3D object surface) and see the corresponding results interactively. Like common color palettes (e.g. in (Shugrina *et al.* 2017)), the combination of user-picked element collections can also be subsequently saved as a new input entry into the element palette for further reuse, and therefore the users are empowered to optionally customize their own element palettes and remix the saved input entry with other aggregate elements in a convenient and flexible procedure. Moreover, while with several controllable parameters, ordinary users are able to iteratively design diverse output compositions with reduced user workload and enhanced synthesis quality within different output domains under an intuitive and user-friendly brushing workflow without sacrificing their authoring freedom. Our developed system, centered on the idea of the autocomplete element fields, can effectively cope with aggregate elements with general shapes, distributions, and alignments, sufficiently provide more usability and interactivity than existing element synthesis practices, and let creators more quickly and properly transform their conceptual ideas into concrete element aggregations like Figure 1.4b without the requirement of a great amount of manual labor or technical expertise. To evaluate the prototype system, we perform a pilot user study and demonstrate its applications for both interactive design and collage as well as batch modeling, with partial or full user specifications.

In the end, the overall objectives we aim to achieve in this research study are briefly summed up as follows:

- To carry out a literature survey, which includes existing element synthesis practices and interactive design interfaces related to aggregate applications, to elaborate on challenging issues related to interactive authoring of aggregate elements or unaddressed points (e.g. the manipulation of user-specified element distributions).

- To devise a general framework, that can simplify the representation of aggregate elements with varying anisotropy and deformability (i.e. by a set of element samples and graphs), to effectively deal with a variety of phenomena and output formations.

- To formulate a practicable solution, that can handle certain specifications (i.e. scalar or direction fields), operate across different output domains (i.e. 2D planes, 3D surfaces, and 3D volumes), support desirable controls (e.g. the manipulation of the topology of element fields) and achieve interactive speed, to facilitate the creation of element aggregations while fulfilling satisfactory production standard.

- To develop a one-step automatic optimization process, that can not only directly compute full outputs from partial inputs but also properly avoid undesirable artifacts (i.e. element misalignments around singularities) and well reflect design intention, to simultaneously reduce input workload and enhance output quality.

- To implement an easy-to-use interface, that can provide sufficient user-friendliness (e.g. brushing workflow) and intuitive interactivity (e.g. element mixing), as an external plugin into a commercial application software to match the natural artist workflow and more flexibly fit element synthesis with interactive authoring.

- To evaluate both theoretical formulation and methodological design, detail information about the system development, conduct a pilot user study for measuring the usability of our prototype system and point out limitations as well as potential future work.

1.3 Contributions

To sum up, the key contributions of the research presented in this thesis are listed as follows:

1. **An element distribution formulation:** On the basis of our devised element representation, the element distribution formulation, which includes the data-driven method (Hsu *et al.* 2018) as well as the procedural approach (Hsu *et al.* 2020), can appropriately distribute aggregate elements with anisotropic shapes and various sizes, considerably reduce the undesired overlaps between all the aggregate elements to a certain degree, effectively maintain each synthesized element's appearance without needing to integrate any additional solvers or processes and yet significantly provide sufficient functionality to flexibly manipulate the distributions of the aggregate elements (e.g. dense, sparse or spatially-varying element distributions) without either expensive computation cost or heavy implementation workload.
2. **An element field formulation:** Our element field formulation, which is able to be applied for both scalar and direction fields, can smoothly orient distinct types of aggregate elements all over the output domain, properly deal with the element arrangements around the singularities, more adequately match the synthesized elements with the original user intention and further supply the controllability for the topology of element fields (i.e. smoother or less smooth element scales and orientations). Additionally, through compatibly combining the element field formulation with the element distribution formulation together into a one-step automatic optimization process, complete element fields can be automatically computed according to inter- and intra-element relationships from given par-

tial specifications (e.g. partially user-specified brush strokes) without the necessity of fully specified inputs in any algorithmic step.

3. **An interactive synthesis system:** Our interactive synthesis system, to this day, is the first for interactive authoring of aggregate elements with varying anisotropy and deformability for various applications within different output domains. Analogous to traditional painting workflows, through the palette-based brushing interface, creators are enabled to directly arrange user-picked element collections or automatically compute full outcomes under an intuitive and user-friendly brushing workflow. With adjustable parameters, ordinary users can iteratively design a variety of novel mixtures consisting of specific aggregate elements with intended orientations and scales without requiring a great deal of technical expertise or manual labor. Like common color mixing in a color palette, user-interactive element mixing can be similarly fulfilled via the element palette as well. The developed system can not only more naturally fit element synthesis with interactive authoring but also significantly offer users better usability and interactivity as compared with existing element synthesis practices.

1.4 Publications

The research papers for this thesis have been published in peer-reviewed journals and conferences as listed below:

1. **Hsu, C.-Y.**, Wei, L.-Y., You, L. and Zhang, J. J., 2020. Autocomplete Element Fields. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 1–13. Available from: `https://doi.org/10.1145/3313831.3376248` [Accessed 1 July 2020]

2. **Hsu, C.-Y.**, Wei, L.-Y., You, L. and Zhang, J. J., 2018. Brushing Element Fields. In *SIGGRAPH Asia 2018 Technical Briefs*, SA '18. Available from: `https://doi.org/10.1145/3283254.3283274` [Accessed 1 July 2020]

3. Palacios, J., Roy, L., Kumar, P., **Hsu, C.-Y.**, Chen, W., Ma, C., Wei, L.-Y. and Zhang, E., 2017. Tensor Field Design in Volumes. *ACM Trans. Graph.*, 36(6). Available from: `https://doi.org/10.1145/3130800.3130844` [Accessed 1 July 2020]

In brief, our data-driven method is first devised in (Palacios *et al.* 2017) to synthesize aggregate elements with varying anisotropy and deformability for tensor field visualization through a two-step process. Then, by combining the data-driven method with the basic idea of our element field formulation as proposed in (Hsu *et al.* 2018), the distributions, orientations and scales of all the synthesized elements can be directly optimized based on partial user specifications via a one-step automatic optimization process. Eventually, our procedural approach with more complete formulations, evaluations and applications is presented in (Hsu *et al.* 2020), and for better usability and interactivity, the interactive synthesis system is consequently carried out.

2 Related Works

Here we carry out a literature review on recent advancements and existing publications about element synthesis and interactive authoring. For better clarification, we categorize related works into four major classifications: element packing, element modeling, field-guided element placement, and interactive design. The element packing algorithms concentrate on optimally packing a certain number of specific aggregate elements (e.g. graphical primitives or image cutouts) into a target output domain. Therefore, in order to acquire an ideal output with compact element distributions, the individual aggregate elements often have to be deformed and resized accordingly to best match the domain shape and minimize the empty space in the output. While the element modeling methods focus on automatically computing output compositions with programmable or user-specified element distributions, both computer-generated geometric shapes (e.g. pentagons and ellipses) and predefined element exemplars can be utilized to design complicated mixtures with customized or special appearances (e.g. repetitive structures or entangled details).

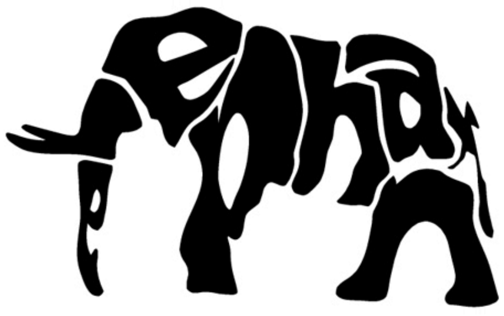
Furthermore, existing element synthesis practices often incorporate full input fields, which can be preprocessed via either manual specification or automatic computation, with their algorithms to better orient aggregate elements with anisotropic shapes and various sizes. In consequence, oriented element arrangements can be accomplished based on such input fields as described in relevant applications regarding the field-guided element placement. Since both interface development and interaction modality can play a vital role in the research field of human-computer interaction, in order to more enhance the usability and interactivity of our interactive synthesis system, it can be desirable and necessary to investigate practicable solutions with general user interfaces related to the interactive design. In the following sections, we thus elaborate the element packing in Section 2.1, the element modeling in Section 2.2, the field-guided element placement in Section 2.3, and the interactive design in Section 2.4 respectively.

2.1 Element Packing

A variety of element packing algorithms have been broadly investigated for various applications, such as mosaics (Hausner 2001; Kim and Pellacini 2002; Dalal *et al.* 2006; Hu *et al.* 2016; Doyle *et al.* 2019), collages (Gal *et al.* 2007; Huang *et al.* 2011; Kwan *et al.* 2016; Saputra *et al.* 2018), glyphs (Xu and Kaplan 2007; Xu *et al.* 2010; Zou *et al.* 2016), and artistic layouts (Schiftner *et al.* 2009; Reinert *et al.* 2013; Huang *et al.* 2014; Saputra *et al.* 2017), over different output domains, including 2D planes, 3D surfaces and 3D volumes. As illustrated in Figure 2.1, the mosaic approaches aim to evenly partition the output domain space into a large number of small pieces, followed by fully matching individual primitives with these partitioned regions, while the collage methods tend to tightly place various objects with specific shapes within the entire output domain. In addition, the glyph avenues utilize characters or symbols to shape or outline the contour of the output domain, and for the artistic layouts, additional specifications like constraints and rules are required to correspondingly arrange the packed aggregate elements. Below we briefly describe each element packing algorithm and highlight open issues in existing practices.

(a) *Doyle* et al. *(2019)*

(b) *Saputra* et al. *(2018)*


(c) *Xu and Kaplan (2007)*

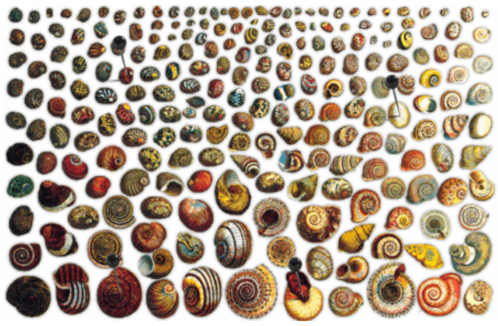

(d) *Reinert* et al. *(2013)*

Figure 2.1: The examples about element packing. *By optimally packing aggregate elements into a given container, compelling outcomes like mosaics (a), collages (b), glyphs (c) and artistic layouts (d) can be produced.*

2.1.1 Mosaics

Hausner (2001) presents an approach to pack similarly-shaped objects, such as square tiles, following a derived direction field into a rectangular image region. The approach utilizes a Voronoi diagram with a manhattan metric to randomly divide the image region into several subregions and then employs these subregions to contain individual objects. Subsequently, each object is moved to the centroid of its corresponding Voronoi cell and aligned with the direction field via an iterative process.

Kim and Pellacini (2002) propose a method to compactly fill an arbitrarily-shaped container image with a set of arbitrarily-shaped image tiles. Similar to (Hausner 2001), this method first constructs a centroidal Voronoi diagram to partition the whole container and then initially places the image tiles into the divided regions one by one on the basis of an energy-based framework which includes color, gap, and overlap energy terms with adjustable weights. In order to minimize the gaps and overlaps between the image tiles, each image tile can be adaptively deformed according to its nearby image tiles via a refinement phase.

By adopting an evenness metric and a tile adjustment step, Dalal *et al.* (2006) introduce an improved technique to accommodate image tiles of varying shapes, such as triangles and stars, within different image containers. Analogous to (Hausner 2001), this technique also constructs a Voronoi diagram to initialize the placement of image tiles. However, instead of utilizing the centroid of each Voronoi cell, the technique iteratively moves and rotates each image tile to the position and orientation that can minimize the sum of squared distances among the perimeter of this image tile and all pixels inside its Voronoi cell.

Hu *et al.* (2016) propose a synthetic method to tightly tile a set of objects with irregular shapes and different textures over curved surfaces. At the beginning, the method randomly distributes a specified number of objects over the base surface and properly shrinks the distributed objects to avoid any overlap between each object. On the basis of this initial object distribution, the surface can be divided into approximate Voronoi regions of the objects via the chordal axis transform (Prasad 1997). In order to fulfill a better coverage for the surface, each tiled object is repetitively inflated, shrunk, spun and translated to best cover its corresponding Voronoi region. Through further replacing and permutating certain objects with large uncovered regions by more compatible candidate objects to minimize vacant holes, a compact configuration can be optimized eventually.

Doyle *et al.* (2019) offer another synthetic method to create digital pebble mosaics with a historical style over an image plane. The method first utilizes simple linear iterative clustering with a modified distance metric to acquire a number of segmented tiles. Subsequently, by smoothing these tiles into rounded shapes, the input image content can be approximated with highly expressive pebbles. Additionally, an extra height field generated from the pebble contours via a Laplace equation can be further used to enhance the final output details.

2.1.2 Collages

Huang *et al.* (2011) give an avenue to produce Arcimboldo-like collages that can disguise a given source image with multiple image cutouts of arbitrary shapes from the Internet while the selected image cutouts can still be recognizable in the source image. To achieve it, the avenue directly segments the source image into several patches according to the mean-shift clustering method (Comaniciu and Meer 2002) at the beginning and then searches for the best matching image cutout for each patch by measuring the consistency of their color histograms and shapes.

Kwan *et al.* (2016) aim to compactly fill a given region with reasonably-sized and irregular shapes. To accomplish this, a pyramid of arclength descriptor is devised by Kwan *et al.* (2016) to facilitate efficient partial shape matching.

By continuously translating, rotating and scaling the selected shapes one by one to match the given region in a progressive packing process, compact collages can be carried out automatically. Moreover, for the improvement of visual quality, an optional deformation step can be further included as a postprocess to minimize the gaps and overlaps between all the shapes via standard shape morphing.

Saputra *et al.* (2018) develop a system, called RepulsionPak, to explicitly pack deformable instances of geometric elements picked from an element library into a container shape for the generation of artistic compositions. In the system, each geometric element is represented by a triangle mesh, and during the initial element placement, a user-specified number of elements with random orientations and minimized sizes are placed inside the target container via blue noise sampling (Bridson 2007). Through a growth process based on physically-based simulation, the meshes of all the elements are gradually translated, rotated, enlarged and deformed until the target container can be tightly filled with these geometric elements.

In 3D volumes, a computerized system devised by Gal *et al.* (2007) focuses on packing a collection of 3D elements retrieved from a database of models into a target shape to form expressive 3D compound shapes. The system employs a partial shape matching algorithm to determine multiple matched elements from a list of candidate elements, and by allowing users to manually select the desired one, a 3D compound layout with expressive and aesthetic aspects can be built accordingly.

2.1.3 Glyphs

Xu and Kaplan (2007) introduce a solution that concentrates on assembling calligraphic letters with a few extra rendering styles into a container region. The solution first adopts a level-set method to initially divide the container region into several subregions and subsequently maps the specific letters, such as English characters, into these subregions through a best-fit warping approach.

Xu *et al.* (2010) provide an approach that can well approximate the main line structure of a given image content by the shape of characters to automatically generate structure-based ASCII art. The approach integrates an alignment-insensitive shape similarity metric and a constrained deformation metric together to determine the best-matched characters and simultaneously minimize the shape dissimilarity and deformation between the characters and the reference image.

Zou *et al.* (2016) present an algorithm to directly create compact calligrams that can have a balance among conveying the shape of a given input image and the legibility of associated letters. This algorithm initially generates an approximate text layout path within the input image outline according to corresponding color-coded letter anchors, and then the individual letters can be collectively deformed based on the approximate path to best match the image outline. To properly preserve the legibility of all the letters, a deformation scheme with a legibility measure learned from crowdsourced data is used to enhance overall readability as well as aesthetics. In addition, a post-process for the final calligram refinement is also exploited so as to minimize the gaps and overlaps between the letters.

2.1.4 Artistic Layouts

Reinert *et al.* (2013) design a system that can automatically arrange a small number of graphical primitives on the basis of their visual properties, such as size, anisotropy and brightness, across a 2D spatial target container. The system is able to not only iteratively order the whole primitives but also infer design intention from user-defined features (e.g. shape and texture) to compute a balanced distance between each primitive while following artistic goals derived from the design intention. Reinert *et al.* (2013) also devise a projection from the space of visual properties into a lower-dimensional layout space to achieve feature mapping for the design intention's inference. Moreover, a generalization of centroidal Voronoi diagram is proposed to evenly distribute the primitives with spatial extent, and in order to enable interactive exploration of different artistic layouts, a GPU computing implementation is consequently utilized to accelerate the computation performance of the iterative relaxation.

Saputra *et al.* (2017) develop a technique, called FLOWPAK, for packing ornamental elements chosen from a library of templates into a container region. The technique first computes a full direction field from user-specified directional guides within the container and then traces evenly-spaced streamlines with desired lengths in the field as a preprocess. During the initial placement of ornamental elements, the target container is partitioned into subregion blobs based on the preprocessed streamlines via an approximate generalized Voronoi diagram, and by using a shape matching step to determine the best ornamental element for each blob, all the ornamental elements can be initially located within the corresponding blobs. Subsequently, to improve the overall layout, each ornamental element can be slightly rotated and deformed to gradually eliminate empty zones and make the spacing more uniform via an iterative refinement process.

Schiftner *et al.* (2009) introduce an algorithm that can compute a triangle mesh with the property of incircle packing for various circle pattern layouts over arbitrary freeform surfaces. The algorithm aims to simultaneously optimize the resulting quality of circle packing, the similarity between the triangle mesh and the target surface as well as the boundary proximity of both the triangle mesh and the target surface through a damped Gauss-Newton approximation.

Huang *et al.* (2014) present an avenue that can automatically construct 3D well-structured layouts composed of mechanical elements from a given database and properly reflect some of the design objectives without user intervention. During the automatic construction, an artist-designed proxy model is required to coarsely depict the desired domain shape consisting of multiple separate proxy components, and an iterative unification algorithm is exploited to repetitively adjust the assembled mechanical elements within each proxy component. Furthermore, the avenue refers to a few graphic design principles, such as unity, variety and contrast, to better emphasize on internal element configuration.

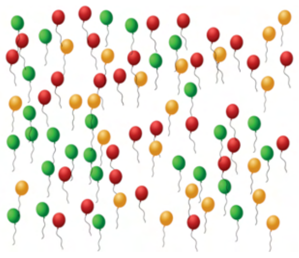
2.1.5 Summary

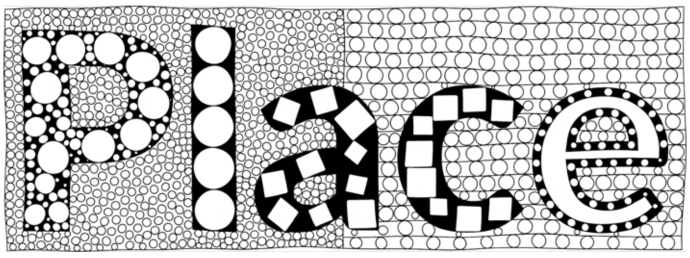
It can be seen that the aforementioned element packing algorithms directly optimize element packing layouts without taking interactivity or user-specified element distributions, orientations and scales into consideration. As the degree of difficulty of packing aggregate elements for arbitrary mixtures can also increase by the amount of element anisotropy, the variety of element deformability, the output domain's size, shape and dimension, as well as given user specifications, to interactively synthesize distinct types of aggregate elements and simultaneously consider their individual distributions, orientations and scales is much more challenging and still remains an open problem, especially in 3D output domains (Cho *et al.* 2007).

Additionally, as described in (Kwan *et al.* 2016; Saputra *et al.* 2018), since to explicitly calculate optimal element packing layouts can require significant computation time and thus compromise interactivity, we formulate the auto-complete element fields that can not only well balance between synthesis quality and performance but also provide flexible manipulations to effectively deal with a variety of phenomena and output formations. Unlike previous element packing algorithms, which mainly concentrate on packing certain types of aggregate elements (e.g. isotropic or rigid elements) within a single type of output domain (e.g. only 2D planar regions), our element distribution formulation can efficiently distribute aggregate elements with varying anisotropy and deformability within different output domains, including 2D planes, 3D surfaces and 3D volumes, as exemplified in Figure 1.4. With our element field formulation, both inter- and intra-element relationships can be taken into account to smoothly organize the overall element arrangements across the entire output domain and match all the synthesized elements with given user specifications at the same time. Through directly incorporating the proposed formulations into the interactive synthesis system to facilitate the creation of element aggregations, not only high functionality but also satisfactory user-friendliness can be accomplished and user-specified element distributions, orientations and scales can be adequately coped with via a general design procedure as well.

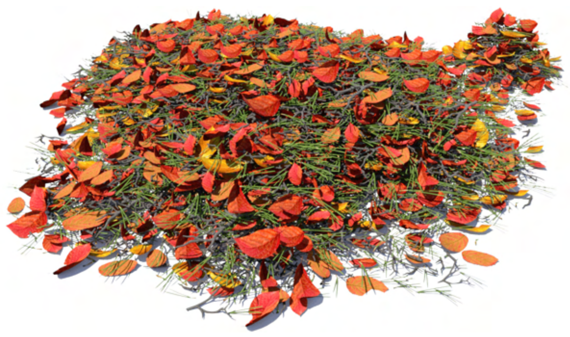
2.2 Element Modeling

As demonstrated in Figure 2.2, modeling aggregate elements considering their individual shapes and distributions can be applied for different applications, such as arrangements (Ijiri *et al.* 2008; AlMeraj *et al.* 2013; Landes *et al.* 2013; Davison *et al.* 2019), structures (Li *et al.* 2011; Roveri *et al.* 2015; Loi *et al.* 2013; Santoni and Pellacini 2016), and aggregations (Peytavie *et al.* 2009; Ma *et al.* 2011; Sakurai and Miyata 2014; Guérin *et al.* 2016), via either data-driven methods or procedural approaches, and by allowing merging or overlapping the synthesized elements, fabricated objects with desired appearances can be formed for practical manufactures (Zhou *et al.* 2014; Zehnder *et al.* 2016; Chen *et al.* 2017; Dumas *et al.* 2018).


(a) *Landes* et al. *(2013)*



(b) *Santoni and Pellacini (2016)*


(c) *Guérin* et al. *(2016)*

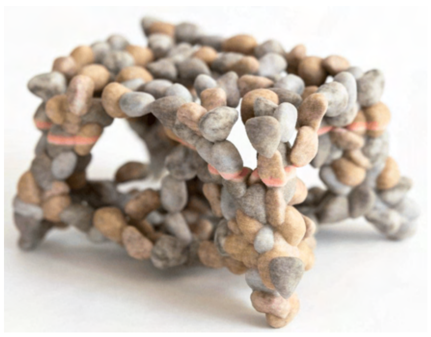

(d) *Dumas* et al. *(2018)*

Figure 2.2: The examples about element modeling. *By properly distributing aggregate elements within the output domain, arrangements (a), structures (b), aggregations (c) and manufactures (d) can be fulfilled respectively according to demand.*

Generally speaking, for the arrangements, predefined element exemplars are needed in advance so as to form user-specified element distributions through the data-driven methods, and about the structures, when with specific specifications (e.g. grammars, programs or examples), structured element distributions can be organized for corresponding configuration outputs. In terms of the aggregations, by reasonably distributing aggregate elements which come from surroundings, natural details can be generated to design and decorate environmental scenes. Moreover, the manufactures provide alternative 3D printing algorithms to fabricate output formations with desired appearances. Below we summarize the prior element modeling methods and discuss unaddressed problems.

2.2.1 Arrangements

Ijiri *et al.* (2008) present an element arrangement system for creative design. The system can produce large output patterns that have local relationships and topologies similar to given reference inputs (e.g. small desired patterns) via an example-based synthesis process. Ijiri *et al.* (2008) arrange the synthesized elements on the basis of a combination of a local neighborhood comparison that can search for the best matching element as well as a local growth process that can avoid undesired structures. Through directly employing a global relaxation to maintain the local characteristic of the resulting patterns, the output patterns with smooth element arrangements can be further acquired.

AlMeraj *et al.* (2013) offer a patch-based strategy that can well synthesize geometric elements from a range of input element exemplars, especially those whose element distributions are not too much regular, over a large container region. This patch-based strategy first tiles the container with a number of overlapping patch copies from the input element exemplar during the initialization and subsequently utilizes a heuristic method to remove the synthesized elements that have undesired overlaps with neighboring geometric elements. In order to better retain the visual property of the input element exemplar, an additional adjustment step is incorporated to repetitively modify local element densities and orientations for the final outputs via an iterative process.

Another data-driven algorithm developed by Landes *et al.* (2013) can properly fill specified output domains with distinct types of aggregate elements which originate from an input element exemplar. The data-driven algorithm relies on proxy geometries, which are rougher versions of the real geometries of individual aggregate elements, to represent the aggregate elements by low-resolution polylines or meshes. Through exploiting spatial relationship measurements between the synthesized elements for the enhancement of element interaction modeling, aggregate elements with anisotropic shapes and various sizes can be well distributed, and undesirable collisions among them can also be effectively avoided.

Davison *et al.* (2019) implement a sketch-based system that can allow users to design virtual objects consisting of a variety of discrete elements from given input element exemplars on 3D surfaces. Analogous to (Ma *et al.* 2011), each discrete element can be represented by single or multiple point samples. In the sketch-based system, a fast region-growing manner is adopted to progressively synthesize new elements around previously synthesized elements, and the sample-based neighborhood metric of Ma *et al.* (2011) is extended to measure the similarity between pairs of the discrete elements. As a consequence, the outcomes with similar element distributions to the input element exemplars can be appropriately produced over the target surface.

2.2.2 Structures

Loi *et al.* (2013) propose a programmable technique for designing a large variety of 2D structured textures. This technique relies on a set of combinable and extensible operators to factorize common element distribution algorithms, such as Lloyd's relaxation and dart throwing. The factorized formulation can enable users to write their own texture programs to automate the production of discrete textures with stylized structures by combining corresponding operators.

Santoni and Pellacini (2016) present a procedural method that can generate 2D tangle patterns composed of a collection of hierarchical and recursive elements such as dots and curves. The procedural method similarly utilizes a set of geometric, decorative and grouping operators to well express tangle drawings with related group grammars. Through explicitly designing the relevant grammars, the tangle patterns with desired structures can be produced correspondingly without the need to specifically represent individual elements in advance.

Li *et al.* (2011) introduce an approach that can procedurally construct geometric or organic patterns with a global structure over 3D manifold surfaces according to the concept of field-guided shape grammars. By properly building on the shape grammars, the approach is able to automatically model the structured patterns following a fully specified vector or tensor field in the Euclidean plane. While further adopting additional collision detection and shape merging steps, the local structures of the patterns can be appropriately optimized.

General structures with repetitive element geometries in 3D output domains can be accomplished by an example-based synthesis avenue (Roveri *et al.* 2015). The example-based synthesis avenue directly combines the sample-based element representation of Ma *et al.* (2011) and a meshless element representation together to indicate discrete and continuous element geometries respectively. Analogous to (Ma *et al.* 2011), through a neighborhood matching metric, Roveri *et al.* (2015) can also measure the structure similarity between the given example geometries and the generated geometries to progressively construct the structures composed of repetitive element geometries in the outputs.

2.2.3 Aggregations

Peytavie *et al.* (2009) propose a procedural tiling approach that can model piles of rocks by a collection of aperiodic cube tiles without incorporating any physically-based simulation. The approach modifies a corner cube generation algorithm (Lagae and Dutré 2006) to initially distribute a set of point samples within the cube tiles and then subdivides the volumes of the individual cube tiles into small rock volumes on the basis of these distributed point samples via the Voronoi diagram with an anisotropic distance function. After utilizing an eroding process that can sculpt and carve all the rock volumes and preserve adequate contact points between the rocks in the cube tiles, the mesh of each rock can be subsequently generated through polygonizing its rock volume.

Ma *et al.* (2011) introduce a data-driven method that can synthesize discrete aggregate elements based on a user-prepared input element exemplar within a given output domain to form a large output with similar appearance to the input element exemplar. At the beginning, the data-driven method adopts point samples to represent individual aggregate elements and subsequently analyzes relative element properties and distributions captured in the input element exemplar via these point samples. Then, according to this analysis, Ma *et al.* (2011) devise a sample-based neighborhood similarity metric to compute the distributions of the synthesized elements in an iterative energy optimization framework. In addition, an extra physics solver can be optionally integrated into the optimization process for the improvement of synthesis quality.

Sakurai and Miyata (2014) present a procedural avenue that can automatically model heaps of arbitrary components, such as fruits and pebbles, within a volumetric aggregate shape. To achieve it, the procedural avenue involves a placement step that first determines the initial positions and orientations of all the components through a dart throwing method as well as a refinement step that can gradually reduce the interpenetrations between the components by iteratively translating or even removing the overlapping components.

Guérin *et al.* (2016) develop a unified framework for modeling large scenes with a large number of natural details, such as grass tufts and leaves. In the framework, a Ghost Tile structure is created to record numerous overlapping candidate objects in a tile and capture a precomputed graph that can indicate the collisions between the candidates. Through traversing the scene with the structure to determine the optimal candidate objects, the natural details can be properly generated over given surfaces or volumes. By means of further incorporating additional density fields into the generation process, the distribution density of different candidate objects can be specified correspondingly.

2.2.4 Manufactures

Zhou *et al.* (2014) mainly concentrate on synthesizing decorative elements from an input pattern exemplar along a specified curve for fabrication. With topology descriptors, their synthesizer can automatically analyze and decompose the input pattern into a certain number of pieces for synthesis optimization. Through directly assembling these pieces along the curve and optimizing the topological properties and pattern geometries of the assembled pieces, the final fabricated target with decorative element components can be precisely constructed.

A computational tool developed by Zehnder *et al.* (2016) can be applied for designing structurally-sound curve networks consisting of ornamental elastic rods over complex surfaces. This tool can assist users in mapping planar rods to arbitrary surfaces, avoiding undesired intersections, tackling contacts, and detecting structural weakness. With a set of editing operations, the users can then enhance the stability of the designed output target for digital fabrication.

Chen *et al.* (2017) propose a method to automatically synthesize a set of basic filigree elements, such as flower and leaf patterns, over a target surface for digital fabrication. The method first employs the medial axis of each filigree element to indicate its skeletal structure and then optimizes filigree configurations on the basis of the skeletal structures of the synthesized elements via a pattern matching energy, which can measure the synthesis quality of the entire filigree pattern. During the optimization period, a nonrigid deformation algorithm is exploited so as to better improve the contacts and alignments between the filigree elements. By partially merging adjacent overlapping elements in an inconspicuous way, the fabricated structure can be connected firmly and sufficiently.

Dumas *et al.* (2018) present a technique to automatically create element aggregations into structurally-sound objects that can be directly manufactured by 3D printing machines. Similar to (Ma *et al.* 2011), the technique also adopts a collection of point samples to represent rigid elements, but in order to well deal with deformable elements, an additional articulated skeleton is necessarily embedded as extra deformation constraints for synthesis optimization. To form printable element aggregations, Dumas *et al.* (2018) depend on a density-based topology optimization process (Bendsoe and Sigmund 2004), which includes a continuation scheme as well as a connection step, to iteratively move and rotate individual aggregate elements and gradually decrease the gaps between the synthesized elements.

2.2.5 Summary

In general, on account of algorithmic limitations or expensive computation cost, a majority of these element modeling techniques mostly concentrate on automatic batch computation without providing sufficient user interaction (e.g. user-interactive element mixing or the manipulation of user-specified element distributions, orientations and scales), and for appropriate element arrangements, multiple proposed algorithms such as (Ma *et al.* 2011; Li *et al.* 2011; Roveri *et al.* 2015; Guérin *et al.* 2016) can demand to incorporate fully specified input fields with their algorithms as part of optimization. Even though interactive speed can be accomplished by Roveri *et al.*

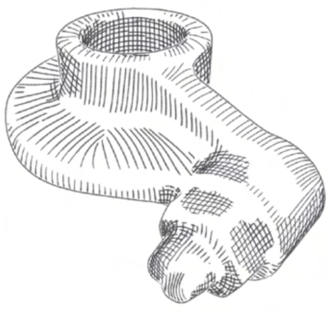
(2015) for certain aggregate elements (e.g. linear helix and chain) or by Davison *et al.* (2019) for specific element arrangements (e.g. repeating element distributions), a general user interface with functional manipulations for interactive authoring is not taken into consideration.

Furthermore, since the previous data-driven element modeling methods (Ijiri *et al.* 2008; Ma *et al.* 2011; Landes *et al.* 2013; AlMeraj *et al.* 2013; Zhou *et al.* 2014; Roveri *et al.* 2015; Guérin *et al.* 2016; Davison *et al.* 2019) rely on the given input element exemplars to synthesize aggregate elements, both synthesis quality and performance can be significantly confined by the prepared input element exemplars, and element mixing among different input element exemplars is not supplied either. Even if the prior procedural element modeling approaches (Peytavie *et al.* 2009; Li *et al.* 2011; Loi *et al.* 2013; Sakurai and Miyata 2014; Santoni and Pellacini 2016; Zehnder *et al.* 2016; Chen *et al.* 2017; Dumas *et al.* 2018) can directly distribute individual aggregate elements, the distributions of the synthesized elements still cannot be flexibly manipulated. As compared with these preceding element modeling techniques, our element distribution formulation can not only freely mix distinct types of aggregate elements but also efficiently generate diverse output compositions with dense, sparse or even spatially varying element distributions over given output domains, and when assisted with a palette-based brushing interface, aggregate elements following certain user specifications can be reasonably accomplished under an intuitive and user-friendly brushing workflow.

2.3 Field-Guided Element Placement

Obviously, aggregate elements following certain input fields (e.g. vector or tensor fields) can be widely seen in a variety of popular applications, such as graphic design (Hausner 2001; Maharik *et al.* 2011; Saputra *et al.* 2017), hatching illustration (Hertzmann and Zorin 2000; Palacios and Zhang 2007; Kalogerakis *et al.* 2012), and texture synthesis (Takayama *et al.* 2008; Zhang *et al.* 2011; Ma *et al.* 2011; Li *et al.* 2011), in 2D or 3D output domains as exemplified in Figure 2.3. In terms of these field-guided element placement applications, there can be no doubt that it is important and necessary to derive a complete and well-designed input field for adequate element alignments due to the fact that the topology and resolution of the input field can have a direct and significant influence on the overall quality of the resulting element alignments, especially for those aggregate elements with anisotropic shapes and various sizes.

(a) *Hausner (2001)*


(b) *Kalogerakis* et al. *(2012)*

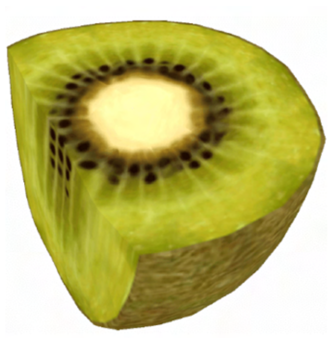

(c) *Takayama* et al. *(2008)*

Figure 2.3: The examples about field-guided element placement. *By appropriately orienting anisotropic elements to pursue fully specified direction fields, multiple applications like graphic design (a), hatching illustration (b) and texture synthesis (c) can be achieved accordingly.*

In the main, the full input fields generated in the graphic design applications have to appropriately match either the color segmentation of a given image or the contour of a container region in order to better express and preserve recognizable features (e.g. boundary edges). While the hatching illustration applications derive the smooth input fields from 3D object surfaces, depending on the point of view, elongated streamlines can be adaptively placed and rendered for fine cross-hatching to illustrate the surface shapes. For the texture synthesis applications, through tailored field design interfaces or algorithms, both image and geometry textures with anisotropic or structural properties can be better synthesized to follow the underlying input fields and maintain the texture features at the same time. In the following, we focus on describing the relationship between the placement of aggregate elements and the generation of input fields and point out potential issues about the field-guided element placement.

2.3.1 Graphic Design

As mentioned in Section 2.1.1, Hausner (2001) demands to incorporate a derived direction field to control the orientations of all the square tiles across the image domain region. Therefore, in order to obtain such a direction field, Hausner (2001) evaluates the image gradients on the basis of information about boundary edge features or optional user-specified curves within the given image. Subsequently, through directly smoothing the directions of the gradients via a Gaussian kernel, the resulting square tiles with a smooth-looking flow can be correspondingly arranged according to the smoothed image gradients.

A digital micrography design tool (Maharik *et al.* 2011) is developed for placing textual content (e.g. words or letters) following a smooth and singularityfree vector field with low curvatures within a given region. Maharik *et al.* (2011) first devise alternative user-imposed alignment constraints and then integrate the method of Palacios and Zhang (2007) to completely generate the desired vector field across the entire region. Furthermore, necessary boundary conditions are also incorporated by Maharik *et al.* (2011) to make sure that all the texts are able to be properly aligned with the region boundaries. Through individually computing the vector field within a collection of user-extracted areas, a well-spaced text layout over the given region can be formed accordingly.

As described in Section 2.1.4, a precomputed direction field is utilized by Saputra *et al.* (2017) to arrange ornamental elements within a container region. Saputra *et al.* (2017) adopt the N-RoSy field design algorithm of Palacios and Zhang (2007) to construct the full direction field according to the given directional guides (e.g. partially user-specified curves) and then compulsively match the placed ornamental elements with this precomputed direction field via a deformation scheme.

2.3.2 Hatching Illustration

In (Hertzmann and Zorin 2000), in order to better illustrate streamlines over surfaces, a full input field for the definition of hatching directions has to be supplied in advance. Thus, Hertzmann and Zorin (2000) directly compute a raw principle curvature direction field (Interrante 1997) from the given surface (e.g. a subdivision surface or an implicit surface) at the beginning and then proceed an optimization procedure to smooth all the principle curvature directions across the whole surface. After smoothing the direction field's topology, by evenly placing the streamlines along with the smoothed direction field on the surface as in (Jobard and Lefer 1997), a smooth cross-hatching illustration can be rendered accordingly.

Palacios and Zhang (2007) provide a system for designing a N-way rotational symmetry (N-RoSy) field on mesh surfaces and then apply the N-RoSy field to pen-and-ink sketching. Palacios and Zhang (2007) employ a relaxation technique (like in (Wei and Levoy 2001)) to initially generate a complete N-RoSy field with a few singularities at the beginning and then smooth the N-RoSy field with several topological editing operations. Analogous to (Zhang *et al.* 2006), the system allows users to remove unwanted singularities or move a singular point to a more desirable position to acquire the smoothed N-RoSy field for streamline placement. In consequence, smooth cross-hatching illustrations can be rendered eventually via the same strategy as in (Hertzmann and Zorin 2000).

Kalogerakis *et al.* (2012) analyze pen-and-ink strokes from a given 3D object's hatching illustration to extract the corresponding per-pixel orientation feature of the pen-and-ink strokes. Through directly mapping the extracted orientation feature to the target object, an entire orientation field over the target surface can be relatively computed via a machine learning framework. Subsequently, the streamline placement algorithm of Hertzmann and Zorin (2000) can be similarly adopted to render illustrations with fine cross-hatching as well.

2.3.3 Texture Synthesis

Takayama *et al.* (2008) present a system that can fill a model with anisotropic solid textures along a user-defined tensor field in 3D volumes. By means of a customized sketch-based user interface, users are able to draw a few brush strokes as inputs across the tetrahedralized mesh. Then, the system can completely interpolate a volumetric tensor field based on the directions of the brush strokes via Laplacian smoothing (Fu *et al.* 2007), and the axes of the solid texture space can be respectively aligned with the corresponding axes of the tensor field for the final output.

Zhang *et al.* (2011) propose an example-based avenue that can extract geometric structures from given 2D examples to synthesize solid textures that match design intention (e.g. guiding lines) within 3D output domains. This avenue also provides a sketch-based interface for users to incrementally draw desired feature curves as specifications, and therefore a smooth tensor field all over the output domain can be fully interpolated according to the user-sketched feature curves via the Laplacian operator. Furthermore, a correction scheme is further utilized in order to properly maintain the geometric structures presented in the given examples, so the resulting solid textures can be well synthesized to follow the tensor field while keeping the original texture features.

As mentioned in Section 2.2.2, a vector or tensor field is employed by Li *et al.* (2011) for the appropriate alignment of shape patterns. Li *et al.* (2011) respectively adapt the algorithms of Zhang *et al.* (2006) and Zhang *et al.* (2007) for interactively designing desirable vector and tensor fields across the output domains. While with a smoothing operation, a few unwanted singular points in the fields can be correspondingly reduced to improve the overall placement of the shape patterns.

Similarly, in (Ma *et al.* 2011), an extra orientation field also has to be integrated so as to well orient individual aggregate elements, and such an orientation field can be either fully specified by hand or automatically generated by existing field interpolation algorithms. By directly incorporating the orientation field as part of synthesis optimization, more interesting element aggregations can be created accordingly.

2.3.4 Summary

It can be observed that in these field-guided element placement applications, their algorithms predominantly adopt a two-step process, which demands to process a full scalar or direction field in advance and then compels the entire aggregate elements to pursue correspondingly. However, it can be difficult and inconvenient for ordinary users to fully specify such scalar or direction fields across arbitrary output domains, especially in 3D environments. Even though the full input fields can be automatically preprocessed by several field interpolation algorithms, desirable output standard or design intention might not be properly fulfilled as exemplified in Figure 1.3d, especially when there can exist undesired singular points in the input fields. While separately employing other standalone field design systems for the generation of the fully specified input fields, users can demand to take extra time and energy to learn essential understanding and knowledge regarding the field design systems in advance and the natural artist workflow as well as the production pipeline can be significantly compromised as well.

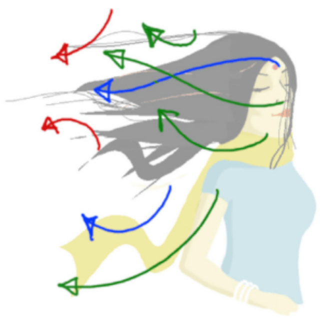
As a result, for better synthesis quality and production efficiency, we thus propose the element field formulation that can completely construct smoother element fields (e.g. Figure 1.3b) according to original user intention (e.g. partial user-specified brush strokes) via our one-step automatic optimization process without the requirement of fully user-specified input fields in any algorithmic step. In contrast to these algorithms presented in the field-guided element placement applications, our element field formulation is able to effectively arrange aggregate elements with varying anisotropy and deformability within different output domains and appropriately match all the synthesized elements with user-specified scalar or direction fields at the same time. In addition, through properly enhancing the field continuity condition in the one-step automatic optimization process, misaligned aggregate elements around singular points in the output can be directly smoothed out without the need to incorporate any additional field preprocessing (e.g. field smoothing). Depending upon personal preferences, smoother, less smooth or even incoherent element fields can also be flexibly dealt with without the requirement of a great deal of technical expertise and manual labor from users.

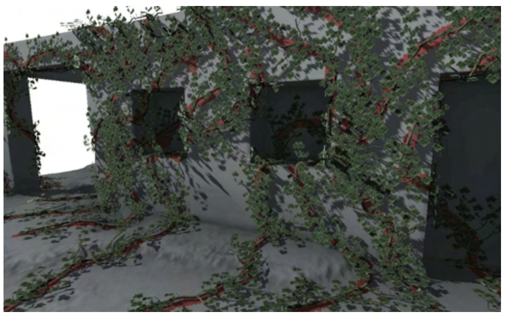
2.4 Interactive Design

Over the past few years, there have been a large number of interactive design systems which are specifically developed for various applications, such as drawings (Lu *et al.* 2012, 2014; Kazi *et al.* 2012, 2014), paintings (Ritter *et al.* 2006; Lu *et al.* 2013; Lukáč *et al.* 2013, 2015), visual effects (Schroeder *et al.* 2010; Chen *et al.* 2012; Xing *et al.* 2016), modelings (Buron *et al.* 2015; Gay 2016a; Frehse 2018), and distributions (Emilien *et al.* 2015; Guérin *et al.* 2016; Davison *et al.* 2019), as demonstrated in Figure 2.4.

(a) *Lu* et al. *(2014)*

(b) *Lukáč* et al. *(2013)*


(c) *Xing* et al. *(2016)*


(d) *Buron* et al. *(2015)*

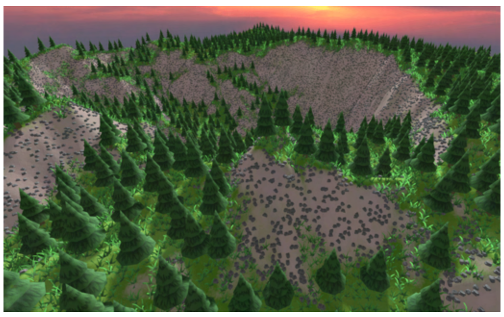

(e) *Emilien* et al. *(2015)*

Figure 2.4: The examples about interactive design. *A number of interactive design systems have been developed for applications in drawings (a), paintings (b), visual effects (c), modelings (d) and distributions (e). With customized interfaces, users are able to create outcomes more efficiently.*

Overall, the drawing systems can assist users in designing decorative or animated illustrations with stylized appearances, while the painting systems directly allow to interactively brush a 2D digital canvas with prepared image or stroke exemplars. When with dynamic input fields or moving particles, certain illustrative, painterly or motion effects for animations can be carried out via the visual effect systems. Furthermore, through the modeling systems, geometric elements with textural details can be effectively modeled onto surfaces of 3D objects, and in the distribution systems, by interactively positioning a desired collection of 3D natural primitives at specified locations, complicated landscapes can be iteratively designed in accordance with user intention. Below we mainly concentrate the discussion on interactive authoring interfaces as well as production workflows and emphasize the aspect about human-computer interaction.

2.4.1 Drawings

Lu *et al.* (2012) present a data-driven approach to empower novice users to produce expressive strokes with stylized trajectories via limited input devices (e.g. mice and multitouch screens). By sufficiently gathering a library of user strokes with 6 degrees of freedom (DOFs: 2D position, 1D rotation, pressure, and 2D tilt) data, the data-driven approach allows users to use less expensive hardware with lower DOF to draw 2-DOF strokes that can look like the 6-DOF strokes in the library. Through applying this stylization approach to line art and handwriting, a drawing system for interactive usage can be designed specifically.

Another data-driven drawing system introduced by Lu *et al.* (2014) can enable users to generate highly structured decorative patterns following user-sketched input paths. Similar to (Lu *et al.* 2012), the data-driven system also builds a library of input stroke exemplars (i.e. geometric primitives) for the synthesis of stylized strokes. With a drawing interface, users are empowered to interactively specify the overall output outline of a pattern, and then the decorative pattern with a similar style to the input stroke exemplars can be produced automatically.

Kazi *et al.* (2012) carry out a sketch-based system to facilitate interactive design and manipulation of 2D pen-and-ink illustrations. Users can first draw a small portion of sketched textures as well as an intended direction path and subsequently let the sketch-based system automatically fill the rest of empty space with the textures along the direction path while well preserving the users' personal design style.

Kazi *et al.* (2014) propose an animation tool that can allow users to draw 2D illustrations with kinetic elements, such as flying birds and falling leaves. The users can directly sketch individual elements and a motion path, set an element emitter which is automatically located at the starting point of the motion path, and then let the element emitter emit the sketched elements moving along the motion path. By further defining the scales and velocities of the emitted elements, a rich set of motion elements can be generated to depict continuous dynamic phenomena.

2.4.2 Paintings

Ritter *et al.* (2006) develop a painting system for generating 2D digital images with nature-looking effects through painting with a set of input texture exemplars. Users can interactively paint an output region with a specific texture selected in the input texture exemplars. With a layered painting framework, which allows users to merge, intersect and overlap their textural strokes, the final output image with desired boundary appearance and alpha information can be generated adequately.

Lu *et al.* (2013) present a data-driven painting system for interactively synthesizing both brush strokes and complicated stroke interactions from a collection of input stroke exemplars without incorporating physically-based simulation. The painting system first scans numerous images of real media such as isolated, overlapping or smudged brush strokes and then factorizes the physical properties and behaviors of the captured stroke data into a library. Users are able to digitally paint a 2D image canvas with the captured media and directly produce plausible artworks with a wide range of painterly appearances.

Lukáč *et al.* (2013) propose an example-based digital painting tool that can assist users in painting an output image with visual stylizations similar to given input image exemplars. The users can initially set a few line features in a reference source image and subsequently design corresponding lines within a target image. The painting tool can consequently transfer relevant textures derived from the source images to the target image via an automatic content-aware fill procedure.

Lukáč *et al.* (2015) provide an example-based painting interface for interactively drawing edge-aware directional textures. Through the painting interface, users are enabled to transfer textural details of a selected source image to a user-specified target shape. Moreover, in order to better control both global and local textural details in the output target, a direction detection and authoring scheme is devised so as to assist the users in the manipulation of textural directions. Therefore, the users can well generate a variety of natural textures with desirable edge effects and interior structures captured in the selected source image.

2.4.3 Visual Effects

Schroeder *et al.* (2010) carry out a sketch-based interface for creating illustrations of 2D vector fields, such as simulated fluid flows, without requiring a great amount of mathematical background or programming expertise from users. By means of translating hand-drawn marks related to an underlying vector field, the users can explore the placement of streamlines with new visual designs and illustrate different styles of outputs that can be consistent with the underlying field data.

Chen *et al.* (2012) propose a system for interactively designing 2D time-varying vector fields over both planar regions and manifold surfaces for controllable dynamic effects, including steerable crowd movement and painterly animation. With this field design system, users can directly specify a few flow descriptors, such as streamlines, pathlines, singularity paths and bifurcations, to generate the full time-varying vector fields. Through employing a certain number of topological editing operations to modify the instantaneous fields at specific times, the users can consequently obtain the desired time-varying vector fields to smoothly control the instantaneous transformation of the dynamic effects.

Xing *et al.* (2016) develop an interactive tool for designing dynamic effects, such as fire and smoke, for 2D animated illustrations. Users can interactively specify several motion paths along with a set of predefined moving elements (i.e. flow particles with different velocities) to dynamically deform underlying illustrations. In order to facilitate the creation of stylized animation effects, the users can further record existing motion paths and subsequently apply these recorded motion paths to other target illustrations through a motion scripting interface.

2.4.4 Modelings

Buron *et al.* (2015) introduce a procedural method for interactively synthesizing structured models, such as growth plants, on top of geometric surfaces. With a GPU-based implementation, users are empowered to dynamically produce highly detailed textural models via a context-sensitive shape grammars modeling system. Through an editing interface, the users can directly guide the generation process and interactively paint the geometric surfaces with the textural models in a production environment.

PhysX Painter (Gay 2016a), which is a 3ds Max plugin, is developed for users to quickly populate scenes with 3D digital assets, such as metal gears, through physically-based simulation. The users can freely select a few assets, interactively brush the selected assets across the output domain and progressively see the resulting simulation outcomes. The plugin can assist the users in effectively creating piles of objects in less time and benefit the production of natural scenes.

Another 3ds Max plugin, called AutoModeller Pro (Frehse 2018), adopts UV mapping to appropriately position geometric elements, such as bricks and wooden sticks, over surfaces. The plugin can not only automatically slice, resize or deform the geometric elements at the borders of object surfaces but also smoothly wrap these geometric elements around the curvature of the target objects. With an editing interface, users can directly model geometries from a user-specified group of meshes on any objects with predefined UV layouts.

2.4.5 Distributions

Emilien *et al.* (2015) present an example-based editing system for interactively designing virtual worlds such as islands with trees. The editing system can capture the distributions of given aggregate elements as well as structured graphs and then construct their interactions on the basis of the relative distributions. By analyzing the captured information, users are enabled to interactively arrange consistent content over specific regions via common painting operations.

Guérin *et al.* (2016) implement a simple painting tool for interactive editing of 3D primitive details for the design of natural scenes (e.g. terrains). Nevertheless, instead of the inside space of a target object, the painting tool merely allows users to paint the target object's surface with the primitive details, and the editing process is still restricted on account of a lack of user-friendly controls.

Davison *et al.* (2019) develop an interactive interface for arranging aggregate elements within 2D planes and 3D surfaces. Users can first choose a single input element exemplar from a palette and then interactively populate the output domain with the chosen input element exemplar via common brush operations.

2.4.6 Summary

It is obvious that these existing interactive design systems are mostly tailored for specific applications in 2D planes or 3D surfaces. Although the 3ds Max plugin (Gay 2016a) can deal with interactive placement of rigid objects in 3D output domains via physically-based simulation, similar to previous element packing algorithms, aggregate elements following user specifications (e.g. directions or scales) are not properly taken into consideration aside from basic gravity and collision. Interactive texturing with a general user interface for multiple applications across different output domains is a more challenging problem and has received less attention until now, in particular in 3D environments.

Analogous to a canvas-based palette tool (Schwarz *et al.* 2007), our interactive synthesis system with a palette-based brushing interface aims to supply users with general and functional authoring controls so that the users can interactively produce desired element arrangements according to their design intention for various applications within different output domains under an intuitive and user-friendly brushing workflow. To facilitate the creation of element aggregations with reduced input workload and enhanced output quality, the developed system is able to automatically compute complete and smooth outcomes on the basis of partial user specifications without the requirement of fully manual authoring. Like common color mixing in a color palette (Shugrina *et al.* 2017), the users can similarly mix desired aggregate elements with specific properties (e.g. element size and rigidity) through an element palette and optionally customize their own personal element palettes as well. As compared with these existing interactive design systems, our interactive synthesis system, focused on the concept of our autocomplete element fields, can not only more naturally fit element synthesis with interactive authoring but also significantly provide more usability and interactivity for users to widely deal with a variety of phenomena and output formations.

3 Autocomplete Element Fields

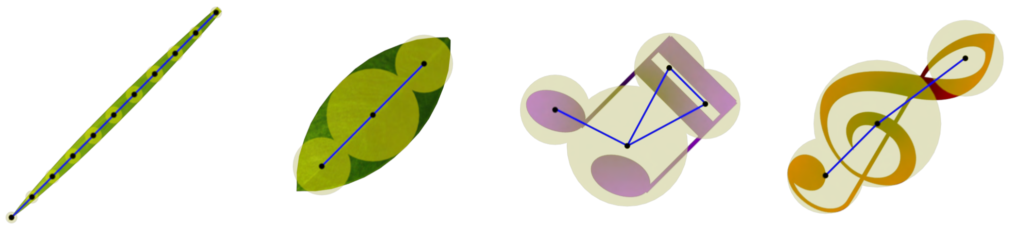
When given multiple input element exemplars $\{I_1, \ldots, I_m\}$ with specific aggregate elements and optional distribution information, an output domain D with desired shape and size, and a partially or fully specified scalar or direction field O over the output domain D, our goal is to automatically complete a full output X composed of aggregate elements from the input element exemplars such that all the aggregate elements within the output domain can be well distributed and have smooth scales or orientations that match the specified input field. In order to accomplish the goal, we therefore formulate a distribution objective E_e, which includes the data-driven method (Hsu *et al.* 2018) and the procedural approach (Hsu *et al.* 2020), in the element distribution formulation and a field objective E_f, which can be applied for both scalar and direction fields, in the element field formulation to simultaneously observe the input element exemplars and the specified input field for the final output. In the following sections, we introduce the devised element representation in Section 3.1, describe the input element exemplar I in Section 3.2, specifically detail the distribution objective E_e in Section 3.3 as well as the field objective E_f in Section 3.4, elaborate the optimization process about our synthesizer in Section 3.5, and offer a summing-up evaluation about our proposed formulations in Section 3.6.

3.1 Element Representation

As inspired by Ma *et al.* (2011, 2013), in our element representation, each aggregate element e can be individually represented by a set of element samples. Nevertheless, only employing a set of element samples is not sufficient enough to appropriately depict distinct types of aggregate elements, especially those aggregate elements with varying anisotropy and deformability. As a consequence, in order to better characterize such aggregate elements for element synthesis, we extend the prior sample-based element representation along with relevant sample weights as well as element graphs to not only effectively reduce inter-element penetrations but also firmly strengthen intra-element connections.

3.1.1 Element Samples

Since we characterize each aggregate element by a collection of element samples, the total number of the element samples and their relative locations in the aggregate element are important. Ideally, we would like to use as few element samples as possible while representing the aggregate element as accurately and reasonably as possible. As exemplified in Figure 3.1, instead of utilizing a certain number of unweighted element samples in (Ma *et al.* 2011, 2013), we exploit a few element samples with related weights to well characterize distinct types of aggregate elements, and by directly considering these sample weights as bounding spheres in our element synthesis process, the inter-element penetrations can be effectively reduced. Depending on the requirement of synthesis quality and performance, the number of these weighted element samples and their positions within an aggregate element can be correspondingly determined in the element representation. For example, in general, using more element samples to represent aggregate elements can more or less increase the synthesis quality, but this can accordingly decrease the overall performance as well. Based on our experiment, in most instances, in order to acquire a satisfactory balance between interactive speed and output standard for element synthesis, the total number of the element samples of a single aggregate element can range between 1 and 8.



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

Figure 3.1: The element representation. *The black dots individually indicate the positions of element samples of each aggregate element. The radius of each yellow circle stand for the sample weight* w_s*, and the blue lines denote the element graphs. In the element representation, the distributions of these weighted element samples in an aggregate element can be sparse (a), dense (b), overlapping (c) or even hybrid (d). Both the element samples and graphs can be either manually specified by designers or directly constructed by existing algorithms (Amenta* et al. *2001; Wang* et al. *2006; Thiery* et al. *2013; Li* et al. *2015). In our current prototype, we set the element samples and graphs via the manual specification.*

3.1.2 Element Graphs

As illustrated in Figure 3.1, in our element representation, we further employ a graph structure G, which is inspired by Sumner *et al.* (2007), to indicate the connectivity among element samples and the relationship between the shapes of aggregate elements and their element samples. As compared with (Ma *et al.* 2011, 2013), which demand to integrate an extra physics solver (Coumans 2013) with their algorithms to better cope with deformable elements (e.g. noodles) for the improvement of synthesis quality (e.g. element shape preservation), by applying this graph structure $\mathbf{G}$ in our element representation, our synthesizer can directly synthesize rigid and deformable elements in high production quality and efficiency without the need to incorporate any additional physics solvers or processes and reasonably dealt with the intra-element connections to firmly preserve the shapes of the synthesized elements in the final output.

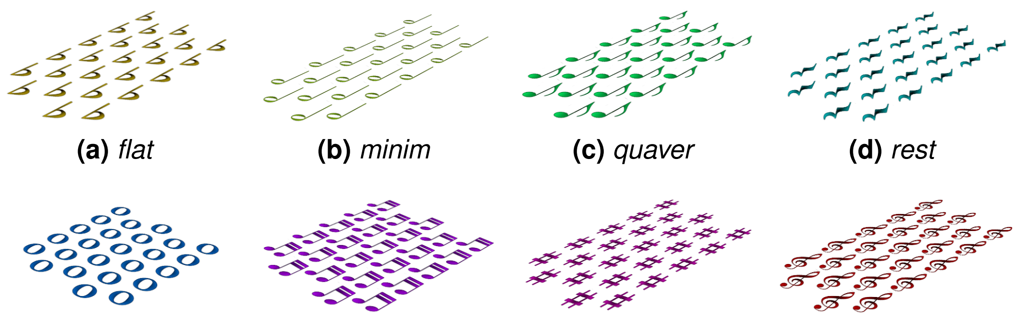
3.1.3 Sample Attributes

In addition to the sample weight $\mathbf{w}_s$, each element sample s also contains individual attributes such as a spatial position $\mathbf{p}$, an orientation matrix $\mathbf{o}$ and a scale $\mathbf{c}$ as well as extra information, such as a sample id, an element id and an exemplar id. Note that the sample id indicates the index of this element sample in the aggregate element, and the element id represents the index of the aggregate element in the output, while the exemplar id means the index of the input element exemplar where the aggregate element originates from. For the distribution objective E_e, the spatial position $\mathbf{p}$ is the variable to optimize, while for the field objective E_f, the optimized variables are the orientation matrix $\mathbf{o}$ and the scale $\mathbf{c}$.

3.2 Element Exemplar

Since to distribute individual aggregate elements can be achieved through either the data-driven method (Hsu *et al.* 2018) or the procedural approach (Hsu *et al.* 2020), depending on which process to carry out, the input element exemplars can possess different information for element synthesis. As the data-driven method aims to synthesize distinct types of aggregate elements on the basis of given element distribution references over output domains, the input element exemplars for the data-driven method have to supply not only a certain number of identical aggregate elements $\{e_1, \ldots, e_l\}$ but also relative distribution information about these identical aggregate elements. While the procedural approach can directly distribute aggregate elements with anisotropic shapes and various sizes without requiring additional distribution information, each input element exemplar for the procedural approach just contains only one specific aggregate element (like each aggregate element in Figure 3.1) as the input primitive (i.e. polygon geometry or vector graphics). Additionally, all the input element exemplars can also capture other relevant information, such as the graph structures of the corresponding aggregate elements, or even their individual image textures and shaders.

In terms of the relative distribution information, as demonstrated in Figure 3.2, it can be observed that each input element exemplar can include a certain number of the same kind of aggregate elements as the pure input primitives for user-interactive element mixing (like pure colors for color mixing), and in order to extract the element distribution references from the input element exemplars, analogous to prior data-driven algorithms (Ma *et al.* 2011, 2013), we can also measure the inter-distances between the element samples of each aggregate element in the input element exemplars. In other words, each input element exemplar for the data-driven method can capture the displacements between each element sample and its neighboring element samples as the relative distribution information, and to determine the neighboring element samples of each element sample, we can alternatively adopt either a specified range (like in (Ma *et al.* 2011, 2013)) or an auxiliary structure used in our procedural approach (see Section 3.3.1.2). Similarly, according to the requirement of synthesis quality and performance, the total number of aggregate elements and their relative distributions, orientations and scales in each input element exemplar can be correspondingly determined as well.



(a) *flat* **(b)** *minim* **(c)** *quaver* **(d)** *rest*

(e) *semibreve* **(f)** *semiquaver* **(g)** *sharp* **(h)** *treble clef*

Figure 3.2: The input element exemplars for the data-driven method. *Here we demonstrate some of the input element exemplars we use. In general, we regularly place distinct types of music symbols in the input element exemplars and capture their relative distribution information via their element samples. Note that the music symbols in the input element exemplars can also be irregularly placed to acquire chaotic element distribution references for element synthesis if there is need.*

3.3 Element Distribution Formulation

To well synthesize distinct types of aggregate elements across different output domains, both interpenetrations and intraconnections have to be properly taken into consideration for all the synthesized elements. In the element distribution formulation, we specifically devise the distribution objective E_e to appropriately distribute the aggregate elements while reasonably dealing with such inter-element penetrations and intra-element connections, and this distribution objective E_e consists of a sample distribution E_d term, a conflict check E_k term, and a graph similarity E_g term.

More detailedly, the sample distribution E_d term aims to adequately distribute aggregate elements with anisotropic shapes and various sizes on the basis of their element samples through either the data-driven method (Hsu *et al.* 2018) or the procedural approach (Hsu *et al.* 2020), and the conflict check E_k term focuses on effectively reducing the inter-element penetrations, while the graph similarity E_g term concentrates on precisely preserving the intra-element connections based on the element graphs of the aggregate elements. Both the sample distribution E_d term and the conflict check E_k term are employed to globally measure the overall quality of the distributions of the synthesized elements (i.e. inter-element distributions) all over the output domain, and the graph similarity E_g term is adopted to locally maintain the coherence of intra-element connections for each aggregate element represented by multiple element samples. Thus, putting the sample distribution E_d term, the conflict check E_k term and the graph similarity E_g term together, we can have the distribution objective E_e as:

$$E_e = (E_d + E_k) \oplus \mathbf{w}_g E_g, \tag{3.1}$$

where $+$ means that the sample distribution E_d term and the conflict check E_k term are minimized together, $\oplus$ indicates that the graph similarity E_g term is minimized separately during the optimization process, and w_g is a relative weight, which is set to 100 from our experiment, to firmly reconstruct the graph structures of the synthesized elements from the distributions of their element samples. More details about the optimization process can be seen in Section 3.5, and in the following sections, to elaborate our distribution objective E_e, we detail the sample distribution E_d term, which includes the data-driven process and the procedural process, in Section 3.3.1, the conflict check E_k term in Section 3.3.2 and the graph similarity E_g term in Section 3.3.3 respectively.

3.3.1 Sample Distribution

To properly place individual aggregate elements within a given output domain can be accomplished by directly distributing the element samples of these aggregate elements via either the data-driven method or the procedural approach. We thus define the sample distribution E_d term as an alternative form of a data-driven sample distribution E_d^d term for the data-driven process and a procedural sample distribution E_d^p term for the procedural process as follows:

$$E_d = \begin{cases} E_d^d & \text{for the data-driven process} \\ \\ E_d^p & \text{for the procedural process} \end{cases} . \qquad (3.2)$$

Based on whether the input element exemplars provide the relative distribution information or not, we can directly determine which process to carry out for element synthesis. Below we detailedly describe the data-driven process for the data-driven sample distribution E_d^d term as well as the procedural process for the procedural sample distribution E_d^p term.

3.3.1.1 Data-Driven Process

The data-driven process depends on the given element distribution references captured in the input element exemplars $\{I_1, \ldots, I_m\}$ to correspondingly compute an output X with similar element distributions to the input element exemplars. As inspired by Ma *et al.* (2011, 2013), to determine whether the output can have the similar element distributions to the input element exemplars, we can directly measure the distribution similarity among the output and the input element exemplars via the neighborhood distances between the element samples of aggregate elements. Nevertheless, since Ma *et al.* (2011, 2013) employ unweighted element samples to characterize distinct types of aggregate elements, the neighborhood distance between two element samples in (Ma *et al.* 2011, 2013) is directly calculated only based on the positions of the element samples without considering inter-sample relationships (i.e. the inter-sample distances associated with the scales and weights of the element samples). Therefore, in (Ma *et al.* 2011, 2013), the distribution similarity among the output and the input element exemplars may not be properly measured, especially for those aggregate elements with anisotropic shapes and various sizes. In order to well deal with such aggregate elements, Ma *et al.* (2011, 2013) consequently have to integrate an additional physics solver (Coumans 2013) with their algorithms for the improvement of synthesis quality.

In contrast to (Ma *et al.* 2011, 2013), in our data-driven process, we not only include the position $\mathbf{p}$ of each element sample but also take the weight $\mathbf{w}_s$, scale $\mathbf{c}$ and orientation $\mathbf{o}$ of the element sample into consideration to more reasonably calculate the neighborhood distances between all the element samples for the measurement of the distribution similarity. To compute the neighborhood distance between two element samples, here we first define the displacement between an element sample s and its neighboring element sample s' as:

$$\hat{\mathbf{p}}\left(s', s\right) = \mathbf{p}\left(s'\right) - \mathbf{p}\left(s\right). \tag{3.3}$$

Then, each neighborhood $\mathbf{N}$ centered at an element sample s with differences of all neighboring element samples $\mathbf{N}_s = \{s'_1, \ldots, s'_n\}$ can be subsequently defined as:

$$\mathbf{N}(s) = \{\hat{\mathbf{p}}\left(s'_1, s\right), \ldots, \hat{\mathbf{p}}\left(s'_n, s\right)\}. \tag{3.4}$$

Furthermore, while an element sample from an input element exemplar is denoted by s_i here, the element sample in the output is represented by s_o. In consequence, as demonstrated in Figure 3.3, when a displacement $\hat{\mathbf{p}}(s'_i, s_i)$ between two element samples s_i and s'_i (i.e. the relative distribution information described in Section 3.2) can be offered by the input element exemplar, we can derive a prediction displacement $\tilde{\mathbf{p}}(s'_i, s_i, s'_o, s_o)$ between two element sample s_o and s'_o in the output from the displacement $\hat{\mathbf{p}}(s'_i, s_i)$ as follows:

$$\begin{aligned}
\tilde{\mathbf{p}}\left(s'_i, s_i, s'_o, s_o\right) &= \mathbf{w}_\omega\left(s'_i, s_i, s'_o, s_o\right)\mathbf{o}(s_o)\hat{\mathbf{p}}\left(s'_i, s_i\right), \\
\mathbf{w}_\omega\left(s'_i, s_i, s'_o, s_o\right) &= \frac{\mathbf{c}(s'_o)\mathbf{w}_s(s'_o) + \mathbf{c}(s_o)\mathbf{w}_s(s_o)}{\mathbf{w}_s(s'_i) + \mathbf{w}_s(s_i)},
\end{aligned} \tag{3.5}$$

where $\mathbf{w}_\omega(s'_i, s_i, s'_o, s_o)$ is an associated weight used to adaptively adjust the prediction displacement $\tilde{\mathbf{p}}(s'_i, s_i, s'_o, s_o)$ based on the scales and weights of the element samples among the input element exemplar and the output. Note that in order to compatibly derive the prediction displacement

$\tilde{\mathbf{p}}(s'_i, s_i, s'_o, s_o)$, both the element samples s_o and s_i should originate from the same kind of aggregate elements in the same input element exemplar (i.e. the same exemplar id) and their sample ids should be the same as well, while the neighboring element samples s'_o and s'_i do not have this limitation as the neighboring aggregate elements can come from different input element exemplars in an arbitrary mixture.

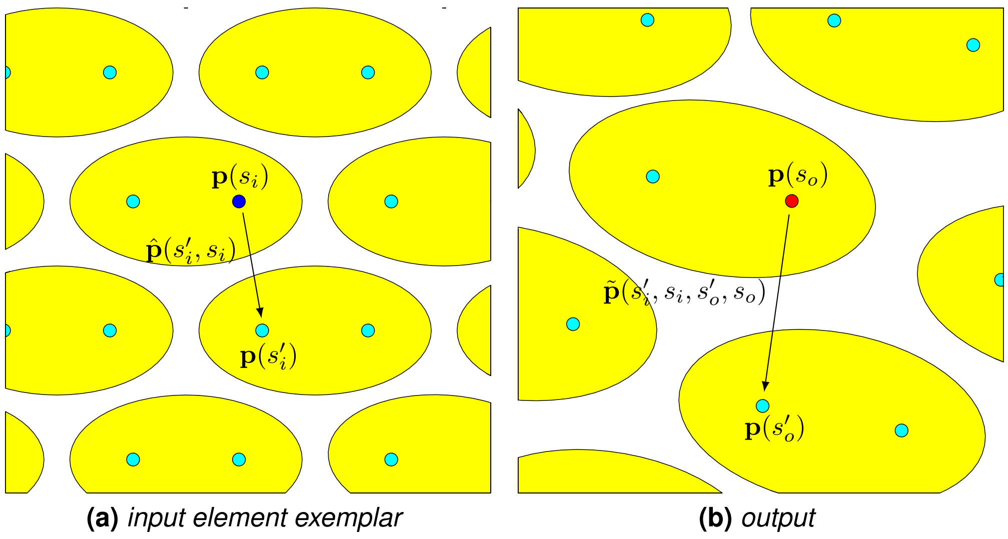

$\mathbf{p}(s_i)$
$\hat{\mathbf{p}}(s'_i, s_i)$
$\mathbf{p}(s'_i)$
(a) *input element exemplar*
$\mathbf{p}(s_o)$
$\tilde{\mathbf{p}}(s'_i, s_i, s'_o, s_o)$
$\mathbf{p}(s'_o)$
(b) *output*

Figure 3.3: The prediction displacement. *Here each yellow ellipse individually stands for an aggregate element, and each aggregate element is represented by two element samples. According to the displacement* $\hat{\mathbf{p}}(s'_i, s_i)$ *between a matching element sample* s_i *(blue) and its neighboring element sample* s'_i *in the input element exemplar (a), the prediction displacement* $\tilde{\mathbf{p}}(s'_i, s_i, s'_o, s_o)$ *between an element samples* s_o *(red) and its neighboring element sample* s'_o *in the output (b) can be derived correspondingly.*

As a result, the neighborhood distance $d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right)$ for a pair of element samples s_o and s_i can be formulated as:

$$d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right) = \sum_{s'_o \in \mathbf{N}_{s_o}} \left|\hat{\mathbf{p}}\left(s'_o, s_o\right) - \tilde{\mathbf{p}}\left(s'_i, s_i, s'_o, s_o\right)\right|^2, \tag{3.6}$$

where $\hat{\mathbf{p}}(s'_o, s_o)$ is the relative displacement between the element samples s'_o

and s_o that we aim to solve here, and $\tilde{\mathbf{p}}(s_i', s_i, s_o', s_o)$ is a constant quantity. As described in (Ma *et al.* 2011, 2013), by searching for the best matching element sample s_i from the input element exemplar for each element sample s_o in the output, we can minimize the neighborhood distance $d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right)$ for each pair of element samples s_o and s_i. Since to look for the best match between two sets (i.e. $\mathbf{N}(s_o)$ and $\mathbf{N}(s_i)$) can be considered as an assignment problem (Ramshaw and Tarjan 2012), to exactly determine these best matches for all pairs of the element samples s_o and s_i, we can employ the Hungarian algorithm (Kuhn 1955) to solve this assignment problem as described in (Ma *et al.* 2013). Hence, when given an input element exemplar I, to acquire the output with similar element distributions to the input element exemplar, the data-driven sample distribution E_d^d term can be subsequently formulated as:

$$E_d^d\left(X, I\right) = \sum_{s_o \in X} \min_{s_i \in I} d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right). \tag{3.7}$$

Moreover, while considering the output X consisting of aggregate elements from multiple input element exemplars $\{I_1, \ldots, I_m\}$, the neighborhood distance $d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right)$ for each pair of element samples s_o and s_i should be properly computed according to the corresponding input element exemplars. Therefore, in terms of the multiple input element exemplars, the data-driven sample distribution E_d^d term for arbitrary mixtures can be consequently extended as follows:

$$E_d^d\left(X, \{I_1, \ldots, I_m\}\right) = \sum_{s_o \in X} \min_{s_i \in I_{\kappa(s_o)}} d\left(\mathbf{N}(s_o), \mathbf{N}(s_i)\right), \tag{3.8}$$

where $\kappa(s_o) \in \{1, \ldots, m\}$, and $I_{\kappa(s_o)}$ means the input element exemplar where the element sample s_o originates from (i.e. the element samples s_o and s_i have an identical exemplar id).

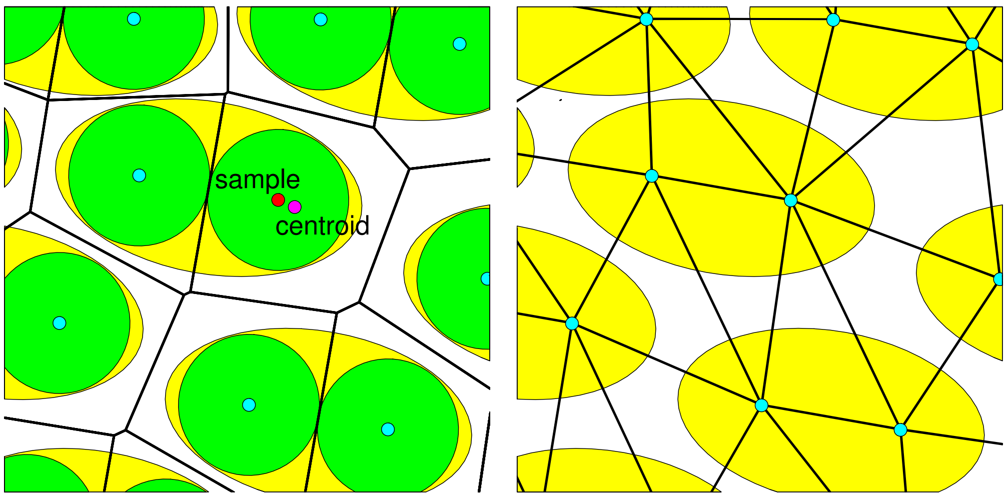
3.3.1.2 Procedural Process

As described above, the data-driven process can synthesize individual aggregate elements on the basis of the given element distribution references captured in the input element exemplars. However, the input element exemplars for our procedural approach only contain a unique aggregate element without providing extra distribution information, and thus to well distribute distinct types of aggregate elements within a given output domain, the procedural process aims to directly balance the interspaces between the aggregate elements by measuring the inter-distances between their element samples. To fulfill this, we thus apply an auxiliary structure, called the power diagram (Aurenhammer 1987), to effectively keep all the element samples in a balanced distance to each other over the entire output domain. As compared with former procedural algorithms (e.g. (Reinert *et al.* 2013; Saputra *et al.* 2018)), which aim to explicitly compute the balanced distances between the aggregate elements in 2D output domains, our procedural process takes a simple and yet efficient strategy that can not only achieve the comparably balanced distances between the synthesized elements across different output domains but also sufficiently offer more flexibility without expensive computation cost or heavy implementation workload.

More detailedly, as the power diagram is a generalized form of the Voronoi diagram and can be used to partition a given domain space into several power cells based on a set of point sites with weights, we can also utilize the power diagram to directly partition the output domain into a number of power cells based on the weighted element samples of aggregate elements and subsequently derive a potential position for each element sample from the centroid of its corresponding power cell. As illustrated in Figure 3.4, analogous to general Lloyd-like optimization methods, by iteratively moving each element sample to the potential position, well-distributed element samples can be gradually acquired, and the procedural sample distribution E_d^p term can be consequently formulated as:

$$E_d^p(X) = \sum_{s \in X} |\mathbf{p}(s) - \mathbf{centroid}(s)|^2, \tag{3.9}$$

where $\mathbf{p}(s)$ is the position of the element sample s that we want to solve, and $\mathbf{centroid}(s)$ treated as a constant here represents the centroid of the power cell of the element sample s. In addition, through further employing this power diagram to acquire the neighboring power cells of each power cell, we can accordingly determine a set of neighboring element samples $\mathbf{N}_s$ for each element sample s as well.



(a) *centroid* **(b)** *triangulation*

Figure 3.4: The power diagram. *Here each yellow ellipse also individually stands for an aggregate element, and each aggregate element is represented by two element samples with weights (green). It can be seen that by utilizing the power diagram to partition the output domain (a), a potential position (purple) for an element sample (red) can be derived from the centroid of the power cell of the element sample. A weighted Delaunay triangulation (b), which is the dual graph of the power diagram, can be further employed to construct the neighborhood information of each element sample.*

3.3.2 Conflict Check

Since mixtures consisting of aggregate elements with anisotropic shapes or various sizes (e.g. Figure 1.4a) may not have either the relevant element distribution references captured in the input element exemplars or the relevant domain space partitioned by the power diagram, in order to increase overall synthesis quality for arbitrary mixtures, the conflict check E_k term is exploited to effectively avoid the conflicts between the aggregate elements by checking the distances between the weighted element samples of the synthesized elements. As the distance between two element samples should not be less than the sum of their sample weights (which act like bounding spheres as illustrated in Figure 3.1), the conflict check E_k term can be formulated as follows:

$$E_k\left(X\right) = \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \mathbf{w}_k\left(s', s\right) \left|\hat{\mathbf{p}}\left(s', s\right) - \mathbf{w}_{\checkmark}\left(s', s\right) \check{\mathbf{p}}\left(s', s\right)\right|^2,$$

$$\mathbf{w}_k\left(s', s\right) = \begin{cases} 1 & \text{if } \mathbf{w}_{\checkmark}\left(s', s\right) > 1 \text{ and } s \in e, s' \in e', e \neq e' \\ \\ 0 & \text{if } \mathbf{w}_{\checkmark}\left(s', s\right) \leq 1 \text{ or } s, s' \in e \end{cases}, \tag{3.10}$$

$$\mathbf{w}_{\checkmark}\left(s', s\right) = \frac{\mathbf{c}(s')\mathbf{w}_s(s') + \mathbf{c}(s)\mathbf{w}_s(s)}{\left|\check{\mathbf{p}}\left(s', s\right)\right|},$$

where $\mathbf{N}_s$ denotes a set of all neighboring element samples of the element sample s as described in Section 3.3.1, $\hat{\mathbf{p}}(s', s)$ is the relative displacement between the element samples s' and s that we aim to solve, and $\check{\mathbf{p}}(s', s)$ stands for the current displacement between the element samples s' and s treated as a constant during the optimization process. Note that since the aggregate elements may be characterized by a few overlapping element samples, the conflict check E_k term ignores those element samples in the same aggregate element, and this can be straightforwardly determined by checking the element ids of the element samples. According to $\mathbf{w}_{\checkmark}(s', s)$, which is a ratio about whether the element samples s' and s can overlap, the conflict check E_k term can be correctly applied to those element samples which can conflict with their neighboring element samples through a conditional weight $\mathbf{w}_k(s', s)$.

3.3.3 Graph Similarity

In addition to the global inter-element distributions, the local intra-element connections should be properly taken into consideration as well. As a result, the graph structure of each aggregate element represented by multiple element samples has to be reconstructed as precisely as possible so as to reasonably maintain the shape of the aggregate element. To well reconstruct such a graph structure, we can minimize the graph distance $d(\mathbf{G}(s), \mathbf{G}'(s))$ between the graph structure $\mathbf{G}(s)$ that we want to retain and the original graph structure $\mathbf{G}'(s)$ for each element sample s in each aggregate element to consistently keep all the graph structures during the optimization process, and therefore the graph similarity E_g term can also be subsequently defined as follows:

$$\begin{gathered} E_g\left(X\right) = \sum_{s \in X} d\left(\mathbf{G}(s), \mathbf{G}'(s)\right), \\ d\left(\mathbf{G}(s), \mathbf{G}'(s)\right) = \sum_{s' \in \mathbf{G}_s} \left|\hat{\mathbf{p}}\left(s', s\right) - \mathbf{c}(s)\mathbf{o}(s)\hat{\mathbf{p}}'\left(s', s\right)\right|^2, \end{gathered} \tag{3.11}$$

where $\mathbf{G}_s$ denotes a set of all connected element samples of the element sample s in the element representation, $\hat{\mathbf{p}}(s', s)$ is the relative displacement between the element samples s' and s that we aim to solve as in Equation (3.10), and $\hat{\mathbf{p}}'(s', s)$ treated as a constant here means the displacement between the element samples s' and s in the original graph structure $\mathbf{G}'$.

3.4 Element Field Formulation

In addition to forming well-distributed aggregate elements, the synthesized elements should also match the specified scalar or direction field O while behaving reasonably well in the specified and unspecified portions of the output domain D. To achieve this, we thus design the field objective E_f to smoothly arrange distinct types of aggregate elements within the entire output domain and simultaneously fit all the synthesized elements with the given input field. This field objective E_f consists of a field alignment E_a term, a field continuity E_c term and an element rigidity E_r term for both scalar and direction fields, and each term is also formulated in element samples for consistency.

More specifically, the field alignment E_a term aims to effectively align the aggregate elements with the given input field in the specified areas, and the field continuity E_c term concentrates on adaptively smoothing inter-element relationships within both the specified and unspecified regions, while the element rigidity E_r term focuses on individually adjusting intra-element relationships based on the element rigidity of each aggregate element. In the element field formulation, both the field alignment E_a term and the field continuity E_c term are exploited to globally organize the element arrangements across the output domain, and the element rigidity E_r term is employed to locally constrain the deformability of each aggregate element represented by multiple element samples. In addition, as the local orientation $\mathbf{o}$ of an element sample is defined as a rotation matrix, in order to penalize the deviation of the orientation $\mathbf{o}$ to well keep a pure rotation matrix for each element sample during the optimization process, analogous to (Sumner *et al.* 2007; Xing *et al.* 2015), an additional rotation constraint E_o term has to be applied for the direction fields. As a result, the direction field objective E_f^d can be a summation of the field alignment E_a term, the field continuity E_c term, the element rigidity E_r term and the rotation constraint E_o term as:

$$E_f^d = E_a + \alpha^2 E_c + E_r + E_o. \tag{3.12}$$

Similarly, putting the field alignment E_a term, the field continuity E_c term and the element rigidity E_r term together, we can have the scalar field objective E_f^s as follows:

$$E_f^s = E_a + \beta^2 E_c + E_r. \tag{3.13}$$

Note that α and β (both default values $= 1$) are controllable parameters that can be tuned to optionally increase or decrease the effect of the field continuity E_c term for specific scenarios and applications (i.e. smoother or less smooth element fields).

As the scales and orientations of element samples are directly associated with the scalar and direction fields respectively, here we first define the distance between two scales $\mathbf{c}$ and $\mathbf{c}`$ for the scalar field objective $E_f^s$ and the distance between two orientation matrices $\mathbf{M}$ and $\mathbf{M}`$ for the direction field objective E_f^d as:

$$d\left(\mathbf{c}, \mathbf{c}`\right) = \left|\mathbf{c} - \mathbf{c}`\right|^2; \tag{3.14}$$

$$d\left(\mathbf{M}, \mathbf{M}`\right) = \sum_{i=1}^{Dim} \left|\mathbf{M}_i - \mathbf{M}`_i\right|^2, \tag{3.15}$$

where Dim stands for the dimension of the output domain, and $\mathbf{M}_i$ and $\mathbf{M}`_i$ represent the $i$-th column vectors of $\mathbf{M}$ and $\mathbf{M}`$ respectively. In the following sections, we first detail the field alignment E_a term in Section 3.4.1, the field continuity E_c term in Section 3.4.2 and the element rigidity E_r term in

Section 3.4.3 for both the scalar field objective E_f^s and the direction field objective E_f^d and specifically describe the rotation constraint E_o term in Section 3.4.4 for the direction field objective E_f^d at the end.

3.4.1 Field Alignment

Since each element sample s is associated with a scale $\mathbf{c}$ and an orientation $\mathbf{o}$, the scale and orientation of the element sample s in the specified output domain should be properly aligned with the scale and orientation of the specified input field O at the closest specified output domain point p to the element sample s, where an output domain point p indicates an output domain unit used to record information about the specified input field O (i.e. the values of scalars or the vectors of directions), and depending on the dimension of the output domain, the output domain unit can be a 2D pixel or a 3D voxel. Hence, for each element sample s in the specified output domain, the distance between $\mathbf{c}(s)$ and $O(p)$ and the distance between $\mathbf{o}(s)$ and $O(p)$ should be minimized accordingly, and the field alignment E_a term can be formulated as:

$$E_a\left(\mathbf{o}, O\right) = \begin{cases} \displaystyle\sum_{s \in X} \mathbf{w}_a\left(s\right) d\left(\mathbf{c}(s), O(p)\right) & \text{for } E_f^s \\ \displaystyle\sum_{s \in X} \mathbf{w}_a\left(s\right) d\left(\mathbf{o}(s), O(p)\right) & \text{for } E_f^d \end{cases},$$

$$\mathbf{w}_a\left(s\right) = \begin{cases} \dfrac{\pi\left(\mathbf{c}'\left(s\right)\mathbf{w}_s\left(s\right)\right)^2}{A} & \text{in 2D output domains} \\ \dfrac{\frac{4}{3}\pi\left(\mathbf{c}'\left(s\right)\mathbf{w}_s\left(s\right)\right)^3}{V} & \text{in 3D output domains} \end{cases}, \tag{3.16}$$

where $\mathbf{w}_a(s)$ is a confident weight which means a larger element sample dominating a larger output domain space should contribute more influence than a smaller element sample, $\mathbf{c}'(s)$ treated as a constant is the current scale of the element sample, and A and V are the 2D area and 3D volume of the output domain unit respectively.

3.4.2 Field Continuity

To adaptively smooth the inter-element relationships for aggregate elements with varying anisotropy and deformability within the entire output domain (including regions with or without the specified input field O), we can orient each aggregate element with its neighboring aggregate elements as similarly as possible, and this can be accomplished through measuring the distance between $\mathbf{c}(s)$ and $\mathbf{c}(s')$ and the distance between $\mathbf{o}(s)$ and $\mathbf{o}(s')$ for each pair of element samples s and s'. As a result, we can also formulate the field continuity E_c term as:

$$E_c(\mathbf{o}) = \begin{cases} \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \mathbf{w}_c(s', s)\, d(\mathbf{c}(s), \mathbf{c}(s')) & \text{for } E_f^s \\ \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \mathbf{w}_c(s', s)\, d(\mathbf{o}(s), \mathbf{o}(s')) & \text{for } E_f^d \end{cases}, \tag{3.17}$$

$$\mathbf{w}_c(s', s) = \frac{1}{1 + \left(\mathbf{w}_{\checkmark}(s', s)^{-1}\right)^2},$$

where $\mathbf{w}_c(s', s)$ is an inverse quadratic radial basis function used to relatively adjust the influence of the field continuity E_c term according to $\mathbf{w}_{\checkmark}(s', s)^{-1}$ in Equation (3.10) (i.e. the inter-distance between the element samples s and s' altered based on the current scales and weights of the element samples), and it means that each pair of element samples with a closer inter-distance $\mathbf{w}_{\checkmark}(s', s)^{-1}$ can have a relatively stronger effect upon the field continuity E_c term. Note that based on our experiment, $\mathbf{w}_c(s', s)$ can well reach a balanced relationship among each term in the element field formulation, and other radial basis functions (e.g. the Gaussian function) can also be considered for other scenarios or specific applications if there is need. In addition, for dimension reduction, in 2D output domains, we optimize the closest three neighboring element samples s' chosen from all the neighboring element samples $\mathbf{N}_s$ for each element sample s according to $\mathbf{w}_{\checkmark}(s', s)^{-1}$, while in 3D output domains, the closest four neighboring element samples s' are chosen and optimized similarly. Relying on this, the field continuity condition can be well tackled with reduced computation cost.

However, it can be observed that singularities in the field may cause undesirable artifacts such as misaligned aggregate elements and to explicitly determine the element arrangements across the singularities can also be tricky in arbitrary mixtures. Hence, in order to more properly arrange the aggregate elements around the singular points, we further utilize an extra adaptive weight to $d(\mathbf{o}(s), \mathbf{o}(s'))$ to accordingly adjust the inter-element relationships, and the field continuity E_c term for the direction field objective E_f^d can be subsequently reformulated as follows:

$$E_c(\mathbf{o}) = \begin{cases} \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \mathbf{w}_c(s', s)\, d(\mathbf{c}(s), \mathbf{c}(s')) & \text{for } E_f^s \\ \sum_{s \in X} \sum_{s' \in \mathbf{N}_s} \mathbf{w}_c(s', s) \sum_{i=1}^{Dim} \mathbf{w}_o(s', s, i) \left|\mathbf{o}_i(s) - \mathbf{o}_i(s')\right|^2 & \text{for } E_f^d \end{cases}, \tag{3.18}$$

$$\mathbf{w}_o(s', s, i) = \left(1 + \mathbf{o}'_i(s) \cdot \mathbf{o}'_i(s')\right),$$

where $\mathbf{w}_c(s', s)$ is the same as in Equation (3.17), $\mathbf{w}_o(s', s, i)$ is the extra weight which aims to adaptively increase or decrease the influence of the field continuity E_c term on the basis of the current orientations of element samples, $\mathbf{o}_i$ represents the i-th column vector of the orientation matrix $\mathbf{o}$ of the element sample that we want to solve, and $\mathbf{o}'_i$ treated as a constant here denotes the i-th column vector of the current orientation matrix $\mathbf{o}'$ of the element sample. As a consequence, when with $\mathbf{w}_o(s', s, i)$, each pair of element samples with similar orientations can also have a relatively stronger effect upon the field continuity E_c term, and by applying this extra adaptive weight $\mathbf{w}_o(s', s, i)$, the aggregate elements near the singular points can be better stabilized with their neighboring aggregate elements during the optimization process (see Figure 3.11). Note that since the values of the dot products of the i-th column vectors (i.e. $\mathbf{o}'_i(s) \cdot \mathbf{o}'_i(s')$) can be the same in 2D output domains but may be different in 3D output domains, the adaptive weight $\mathbf{w}_o(s', s, i)$ for each pair of the i-th column vectors has to be individually calculated in 3D output domains.

3.4.3 Element Rigidity

If the aggregate element e is rigid, all element samples in this aggregate element should have the same scales and orientations (i.e. $\mathbf{c}(s) = \mathbf{c}(s')$ and $\mathbf{o}(s) = \mathbf{o}(s'), \forall s, s' \in e$). If the aggregate element is deformable, the element samples of the aggregate element may have different scales and orientations. Therefore, to distinguish between rigid and deformable elements, the distance between $\mathbf{c}(s)$ and $\mathbf{c}(s')$ and the distance between $\mathbf{o}(s)$ and $\mathbf{o}(s')$ for each pair of element samples s and s' connected by an element graph in the aggregate element can be similarly measured, and the element rigidity E_r term can be consequently formulated as:

$$E_r(\mathbf{o}) = \begin{cases} \sum_{s \in X} \sum_{s' \in \mathbf{G}_s} \mathbf{w}_r(e)\, d\left(\mathbf{c}(s), \mathbf{c}(s')\right) & \text{for } E_f^s \\ \sum_{s \in X} \sum_{s' \in \mathbf{G}_s} \mathbf{w}_r(e)\, d\left(\mathbf{o}(s), \mathbf{o}(s')\right) & \text{for } E_f^d \end{cases}, \tag{3.19}$$

where $\mathbf{w}_r(e)$ is the weight of the element rigidity of the aggregate element e. In our element field formulation, we set the weight of the element rigidity to

100 for rigid elements, while the weight of the element rigidity can be set to 0 for deformable elements.

3.4.4 Rotation Constraint

Analogous to (Sumner *et al.* 2007; Xing *et al.* 2015), since each column vector of a rotation matrix must have a unit length and all the column vectors have to be orthogonal to each other, in order to correctly maintain the orientation $\mathbf{o}$ of each element sample during the optimization process, we can additionally define the rotation constraint E_o term for the direction field objective E_f^d as follows:

$$\begin{gathered} E_o\left(\mathbf{o}\right) = \sum_{s \in X} \mathbf{R}\left(\mathbf{o}(s)\right), \\ \mathbf{R}\left(\mathbf{o}(s)\right) = \sum_{i=1}^{Dim} \left(\mathbf{o}_i\left(s\right) \cdot \mathbf{o}_i\left(s\right) - 1\right)^2 + \sum_{1 \le i < j \le Dim} \left(\mathbf{o}_i\left(s\right) \cdot \mathbf{o}_j\left(s\right)\right)^2, \end{gathered} \tag{3.20}$$

where $\mathbf{o}_i(s)$ and $\mathbf{o}_j(s)$ are the i-th and j-th column vectors of the orientation matrix $\mathbf{o}$ of the element sample respectively. Note that this rotation constraint E_o term is a nonlinear function in the matrix entries.

3.5 Synthesizer

To solve the distribution objective E_e and the field objective E_f, our synthesizer directly optimizes all the synthesized elements with the given specifications on the basis of the positions, orientations and scales of their element samples across the entire output domain through an iterative optimization procedure, and the fundamental idea of our optimization procedure is that we separately approach the minimization of each objective by iterations so as to gradually optimize the individual objectives for the final output. The pseudocode of the optimization procedure in our synthesizer can be seen in Algorithm 1, and illustrations about the iteration process of our optimization procedure for both 2D and 3D cases are demonstrated in Figures 3.5 and 3.6 respectively.

3.5.1 Initialization

During the initialization, distinct types of aggregate elements from multiple input element exemplars are randomly placed into the output domain, and the orientations and scales of element samples of each aggregate element are initially set according to the given specifications (e.g. scalar or direction fields) captured in the closest specified output domain point to the aggregate element. To quickly obtain such a closest specified output domain point within the specified regions, we take advantage of a k-d tree as an acceleration data structure to facilitate the nearest neighbor searching for the set of the specified output domain points. This k-d tree can also be subsequently employed to quickly obtain relevant information about the specified input fields at the closest specified output domain points for the field alignment E_a term during the iterations. In order to handle boundary conditions, additional fixed samples with a unit weight are added to the output domain boundary in this stage. Thus, by similarly applying the conflict check E_k term (see Section 3.3.2) to these boundary samples and their neighboring element samples together as part of optimization, all the element samples can be effectively confined within the output domain. Furthermore, since the areas or volumes of individual aggregate elements can be calculated in preprocessing, the total number of aggregate elements placed into the output domain can be proportionally estimated on the basis of the total area or volume of the output domain. Related implementation details about the initialization are available in Section 4.1.2.

3.5.2 Iterations

In each iteration, we take a distribution step, an orientation step, and a scale step for different objectives (i.e. E_e, E_f^d, and E_f^s), and each step optimizes only one variable (i.e. the position $\mathbf{p}$, the orientation $\mathbf{o}$, and the scale $\mathbf{c}$) at a time. At the beginning, the neighborhood information of each element sample is constructed through the power diagram as illustrated in Figure 3.4b. Based on the neighborhood information of each element sample, we can first carry out the scale step for the scalar field objective E_f^s and then run the orientation step for the direction field objective E_f^d. Subsequently, the distribution step is fulfilled for the distribution objective E_e at the end of each iteration.

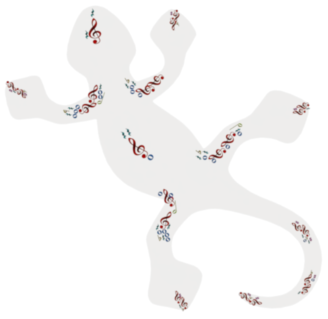
In the scale step, the scale $\mathbf{c}$ is the variable, and we can directly solve the scalar field objective E_f^s via linear least squares. While the orientation $\mathbf{o}$ is the variable in the orientation step, due to the rotation constraint E_o term (i.e. the nonlinear term), we optimize the direction field objective E_f^d via the Levenberg-Marquardt method (Madsen *et al.* 2004) to solve this non-linear least squares problem. Finally, in the distribution step, the position $\mathbf{p}$ is the variable, and as there can exist an irreconcilable conflict between the global inter-element distributions and the local intra-element connections, which may lead to a suboptimal result in terms of the positions of the element samples, we thus first solve the sample distribution E_d term and the conflict check E_k term together to optimally arrange the inter-element distributions via linear least squares. Then, for the aggregate elements represented by multiple element samples (like in Figure 3.1), as the current optimized positions of their element samples may not be properly located at the expected or reasonable places, to firmly reconstruct the intra-element connections for such aggregate elements, we can solve the graph similarity E_g term multiplied by the relative weight $\mathbf{w}_g$ on the basis of the current optimized positions, orientations and scales of their element samples via linear least squares as well. Note that the relative weight $\mathbf{w}_g$, which is set to 100 from our experiment, aims to increase the influence of the graph similarity E_g term so that the shapes of the synthesized elements can be precisely maintained according to the corresponding element samples and graphs.

3.5.3 Output

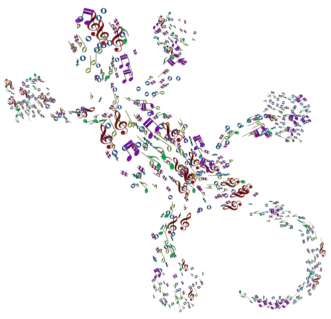
For the final output, the weighted element samples of the aggregate elements can be employed as the control points to appropriately regularize the source mesh vertices of the aggregate elements as in (Sumner *et al.* 2007), and at this final stage, these synthesized elements in the output can also be accordingly assigned related image textures and shaders captured in the corresponding input element exemplars.

```
Function $X \leftarrow$ SYNTHESIZER($\{I_1, \ldots, I_m\}, O$)
  // $X$: the output
  // $\{I_1, \ldots, I_m\}$: the input element exemplars
  // $O$: the partially or fully specified scalar or direction field
  $X \leftarrow$ INITIALIZATION($\{I_1, \ldots, I_m\}, O$)
  ITERATIONS($O, X$)
  OUTPUT($\{I_1, \ldots, I_m\}, X$)
Return $X$

Function ITERATIONS($O, X$)
  Iterate until convergence or enough # of iterations reached
    $\forall s \in X$, $\mathbf{N}_s \leftarrow$ building the power diagram
    SCALE($O, X$)
    ORIENTATION($O, X$)
    DISTRIBUTION($X$)
  End
End

Function SCALE($O, X$)
  $\forall s \in X$, $\mathbf{c}(s) \leftarrow$ solving $E_f^s$ with $O$
End

Function ORIENTATION($O, X$)
  $\forall s \in X$, $\mathbf{o}(s) \leftarrow$ solving $E_f^d$ with $O$
End

Function DISTRIBUTION($X$)
  $\forall s \in X$, $\mathbf{p}(s) \leftarrow$ solving $E_d$ and $E_k$ together
  $\forall s \in X$, $\mathbf{p}(s) \leftarrow$ solving $\mathbf{w}_g E_g$ from the current optimized $\mathbf{p}(s)$, $\mathbf{o}(s)$ and $\mathbf{c}(s)$
End
```

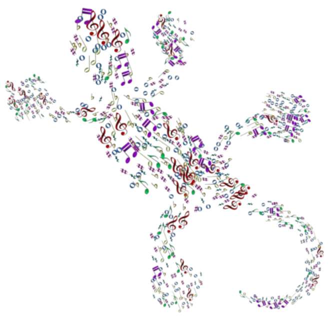
Algorithm 1: The pseudocode of the optimization procedure in our synthesizer. *Our synthesizer iteratively optimizes the individual objectives to observe the input element exemplars $\{I_1, \ldots, I_m\}$ and the given specified input field O for the output X.*



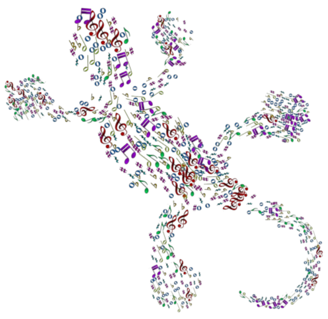
(a) *partially specified input*



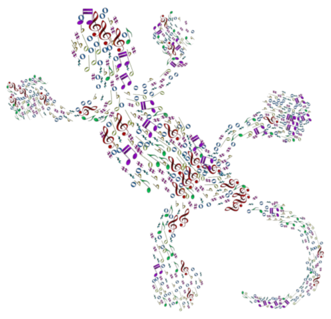
(b) *initialization*



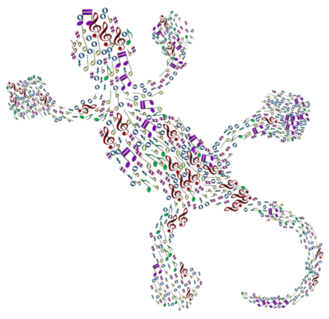
(c) *iteration 1*



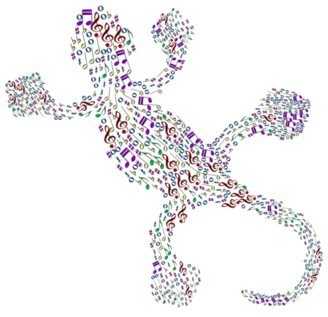
(d) *iteration 2*



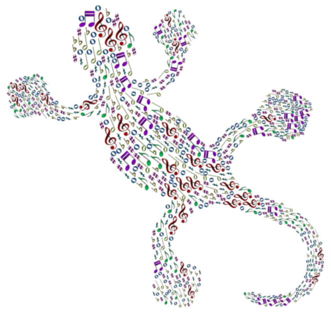
(e) *iteration 4*



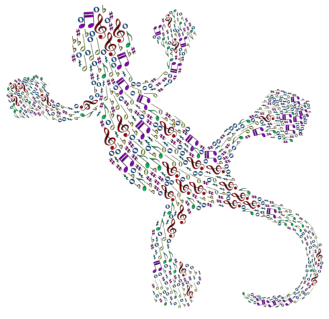
(f) *iteration 8*



(g) *iteration 20*

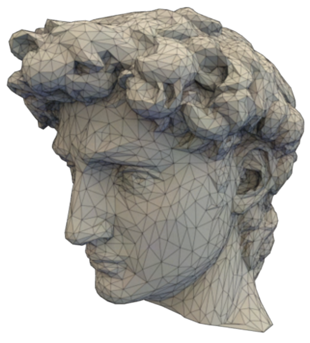


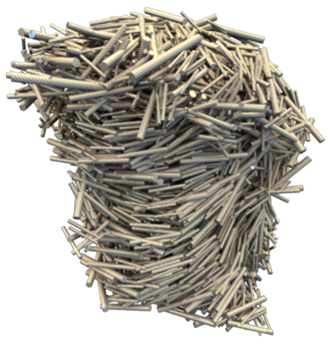
(h) *iteration 50*

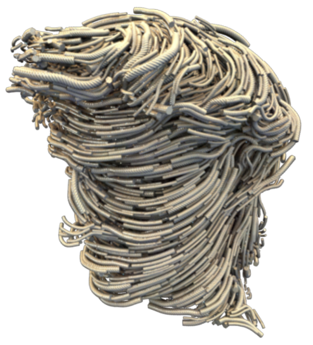


(i) *iteration 100*

Figure 3.5: The iteration process of our optimization procedure for the 2D case. *Distinct types of music symbols are randomly placed into the output domain and initially aligned with a partially specified input (a) during the initialization (b), and our synthesizer respectively optimizes the positions, orientations and scales of all the element samples via an iterative manner (i.e. (c) to (i)). Note that the input (a) is just to imply the intended element orientations and scales instead of being optimized together (see Section 4.1.3.2). According to the number of aggregate elements, the complexity of mixture, the completeness of input fields, and the output domain's size, shape and dimension, the total number of iterations can range from 5 to 100.*


(a) *input mesh*

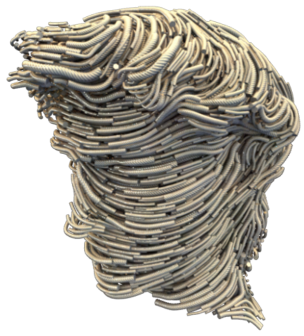

(b) *initialization*

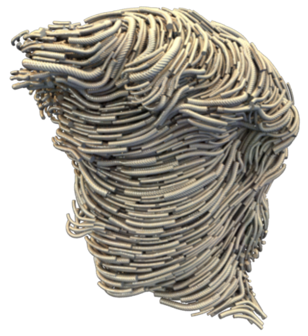

(c) *iteration 1*

(d) *iteration 2*

(e) *iteration 3*

(f) *iteration 5*


(g) *iteration 10*


(h) *iteration 20*

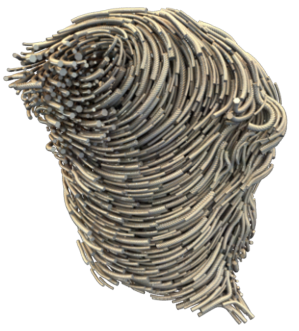

(i) *cross-sectional view of (h)*

Figure 3.6: The iteration process of our optimization procedure for the 3D case. *We compute an extrinsically smooth direction field (Huang and Ju 2016) over a sparse set of mesh vertices (a) as the partially specified input and employ our synthesizer to automatically compute a full volumetric output (i.e. not just over the surface). The cross-sectional view (i) shows that the full outcome (h) can be smoothly constructed from only the surface input field (i.e. a very small subset relative to the entire volume) without the need to predefine the whole resolution of the input field and interpolate the full input field.*

3.6 Formulation Evaluation

In this section, we evaluate the element distribution formulation as well as the element field formulation in terms of the distribution manipulation and the field manipulation respectively. Below we utilize several representative examples under specific situations for better illustrations, and more complete outcomes about practical applications can be seen in Chapter 4.

3.6.1 Distribution Manipulation

In our element distribution formulation, aggregate elements with anisotropic shapes and various sizes can be distributed via either the data-driven method (Hsu *et al.* 2018) or the procedural approach (Hsu *et al.* 2020), and thus here we first make a comparison between our data-driven method and our procedural approach in terms of synthesis quality and performance. Since the data-driven process demands to search for the best matches for all pairs of element samples, the size of each input element exemplar (i.e. the number of aggregate elements in the input element exemplar) has to be restricted in order to speed up the search process for better performance. Therefore, the data-driven method cannot reasonably deal with the element distributions for arbitrary mixtures and consequently compromise the synthesis quality by reason that the input element exemplars with limited sizes may not sufficiently provide relevant distribution information for element synthesis. In contrast, as the procedural process can directly distribute the aggregate elements on the basis of the power cells of their element samples, the synthesized elements can be uniformly distributed without requiring expensive computation cost. As demonstrated in Figure 3.7, it can be seen that our procedural approach can significantly outperform our data-driven method in both synthesis quality and performance. In consequence, unless noted otherwise, we majorly generate the output results shown in this thesis through our procedural approach due to the remarkable advantage in terms of the output standard and efficiency.

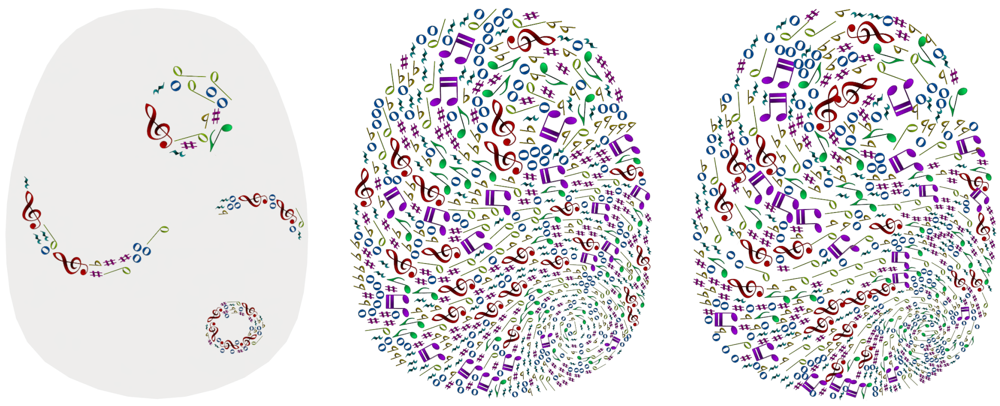

(a) *partially specified input* **(b)** *Hsu* et al. *(2020)* **(c)** *Hsu* et al. *(2018)*

Figure 3.7: The comparison between the data-driven process and the procedural process. *When given the same partially specified input (a), our procedural approach (b) can more uniformly distribute distinct types of aggregate elements in better performance than the data-driven method (c). The outputs in (b) and (c) contain the same 689 aggregate elements and 1598 element samples, and the synthesis times of (b) and (c) are 9 and 44 seconds respectively in 100 iterations, and the CPU we use is Intel® Xeon® E5-1650 3.20GHz. Note that we employ the input element exemplars in Figures 3.2a to 3.2h to generate the output in (c).*

In addition, as our data-driven method relies on the given element distribution references captured in the input element exemplars to synthesize aggregate elements, the flexibility of manipulating the distributions of the aggregate elements can also be significantly confined owing to the factor that relevant input element exemplars have to be accordingly prepared for corresponding element distributions in advance. Similarly, because of algorithmic limitations or expensive computation cost, existing data-driven methods or other procedural approaches do not take the manipulation of element distributions into account either. By comparison, our procedural approach can flexibly cope with diverse output compositions with dense, sparse or spatially varying element distributions under the same element synthesis process, and such a flexibility can further give a clear demonstration that our procedural approach can definitely offer more functionality than our data-driven method and prior algorithms.

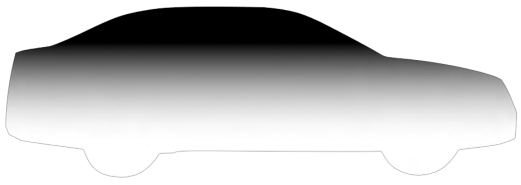
More detailedly, it can be straightforward for our procedural approach to uniformly form dense and sparse element distributions through directly controlling the total number of aggregate elements placed within the output domain. While taking spatially varying element distributions into consideration, our procedural approach can incorporate an additional density field, which is also a scalar field, to spatially vary the distributions of the aggregate elements. To fulfill this, we can directly modify the weights of element samples of each aggregate element by the given density field during the optimization process but do not change the actual sizes of the synthesized elements for the final output as follows:

$$\mathbf{w}'_s\left(s\right) = \left(1 + \rho\left(s\right)\right)\mathbf{w}_s\left(s\right), \tag{3.21}$$

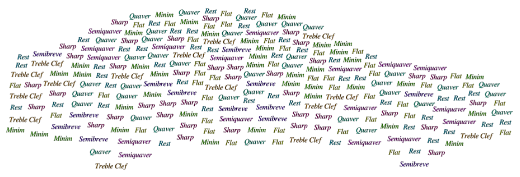
where $\rho(s)$ is the density with a value between 0 and 1 in the density field. This density field can also be optimized independently via the same formulation and step with the scalar field objective E^s_f during the iterations, and through directly replacing $\mathbf{w}_s(s)$ by $\mathbf{w}'_s(s)$ in our proposed formulations, our procedural approach can not only densely or sparsely distribute aggregate elements but also spatially vary the overall element distributions according to demand as exemplified in Figure 3.8. Note that by alternatively taking advantage of a control map or a density function, the additional density field can be generated either partially or fully.

(a) *dense element distributions*

(b) *sparse element distributions*



(c) *control map*



(d) *spatially varying element distributions*

Figure 3.8: The manipulation of element distributions. *Through directly controlling the total number of aggregate elements placed into the output domain, the aggregate elements (i.e. words) can be distributed densely (a) or sparsely (b). While optionally taking advantage of a control map (c) as the density field, the distributions of the synthesized elements can be varied spatially (d).*

Since a majority of existing practices either only work for rigid elements (e.g. (Reinert *et al.* 2013; Kwan *et al.* 2016)) or need extra schemes such as physics solvers for deformable elements (e.g. (Ma *et al.* 2011; Saputra *et al.* 2018)), the capability of freely mixing specific aggregate elements is usually confined. In contrast, based on our devised element representation, our element distribution formulation can effectively synthesize aggregate elements with varying anisotropy and deformability in a satisfactory output standard and directly bring a variety of novel mixtures into existence without the need to incorporate any additional solvers or processes. Analogous to common color mixing, this capability of being able to freely mix multiple rigid and deformable elements can definitely provide more flexibility for the creation of element aggregations as well. Furthermore, while the specified input field O is not given (i.e. only optimizing the element distributions), the element arrangements in the output can be entirely regular as exemplified in Figures 3.8a and 3.8b (i.e. the same element orientations and scales). In order to enhance visual diversity of artistic works, we can utilize extra local orientations and scales to individually manipulate the initial orientations and scales of aggregate elements, and thus different styles of mixtures composed of the aggregate elements with regular or irregular orientations or scales can be accordingly produced as demonstrated in Figure 3.9.

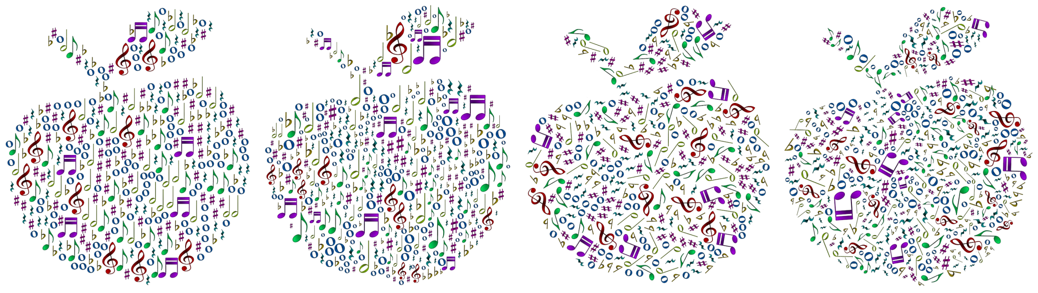
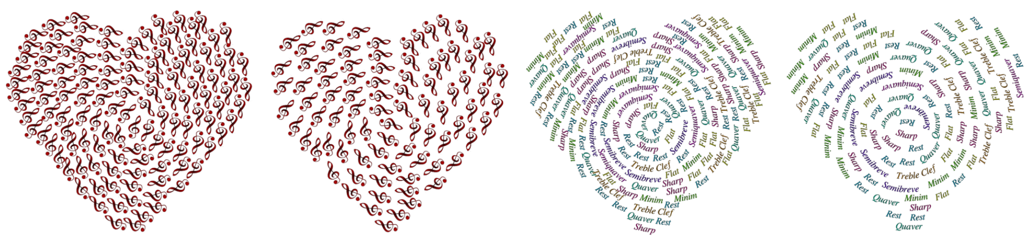
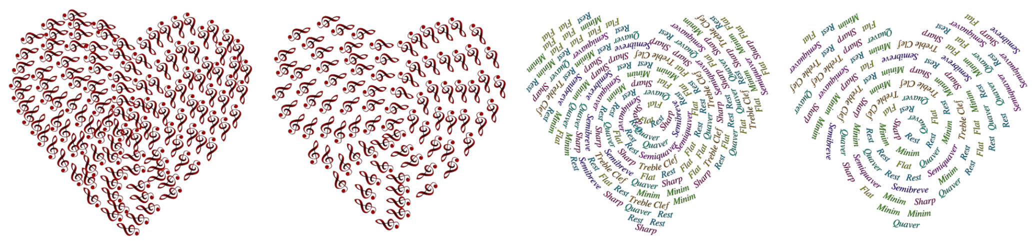

(a) *regular both* **(b)** *random scales* **(c)** *random orientations* **(d)** *random both*

Figure 3.9: The enhancement of visual diversity of artistic works. *Through regularly or randomly initializing the orientations or scales of elements, multiple types of music symbols with regular scales and orientations (a), random scales and regular orientations (b), regular scales and random orientations (c), or random scales and orientations (d) can be correspondingly synthesized for different output appearances.*

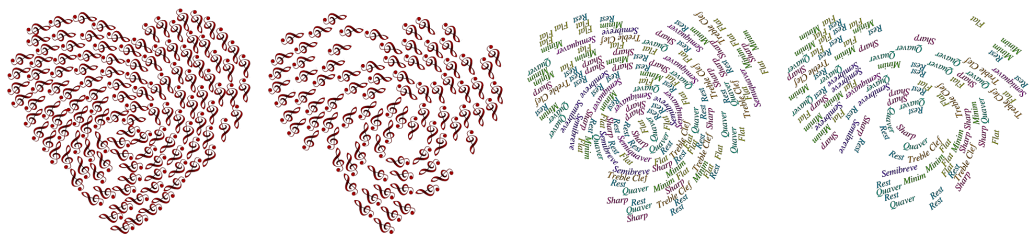
At the end, as illustrated in Figure 3.10, we further perform an ablation study about the sample distribution E_d term and the conflict check E_k term in our element distribution formulation to clarify their individual effects upon the distributions of aggregate elements. Note that as previously described and exemplified in Figure 3.7, our data-driven method may not distribute the aggregate elements uniformly enough, and this can consequently affect the ablation study result. Therefore, in order to better demonstrate the effect differences between the two terms, we employ the procedural sample distribution E_d^p term (i.e. the procedural process) as the sample distribution E_d term in our ablation study.



(a) E_d *on,* E_k *on* **(d)** E_d *on,* E_k *on* **(g)** E_d *on,* E_k *on* **(j)** E_d *on,* E_k *on*



(b) E_d *on,* E_k *off* **(e)** E_d *on,* E_k *off* **(h)** E_d *on,* E_k *off* **(k)** E_d *on,* E_k *off*

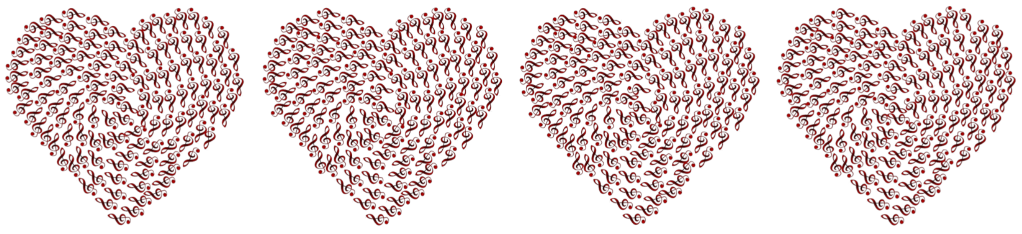


(c) E_d *off,* E_k *on* **(f)** E_d *off,* E_k *on* **(i)** E_d *off,* E_k *on* **(l)** E_d *off,* E_k *on*

Figure 3.10: The ablation study. *For better clarification, we densely (i.e. (a), (b), (c), (g), (h), and (i)) and sparsely (i.e. (d), (e), (f), (j), (k), and (l)) distribute rigid (i.e. treble clefs) and deformable (i.e. words) elements respectively and study the sample distribution* E_d *term and the conflict check* E_k *term in different cases to demonstrate their individual effects. It can be observed that while aggregate elements are densely distributed without the conflict check* E_k *term as shown in (b) and (h), the synthesized elements may overlap, especially near the singular point (i.e. the center of the heart). When without the sample distribution* E_d *term, the synthesized elements do not overlap, but both dense and sparse element distributions can be non-uniform as illustrated in (c), (f), (i) and (l).*

3.6.2 Field Manipulation

In our element field formulation, we utilize rotation matrices instead of radians to indicate local orientations because of the need to specifically utilize the adaptive weight $\mathrm{w}_o(s', s, i)$ in Equation (3.18) for the direction field objective E_f^d to better stabilize aggregate elements with their neighboring aggregate elements around singular points. In Figure 3.11, we first show an example to demonstrate the effect of the adaptive weight $\mathrm{w}_o(s', s, i)$.


(a) *outputs with the weight $\mathrm{w}_o(s', s, i)$ via 70, 80, 90 and 100 iterations from left to right*

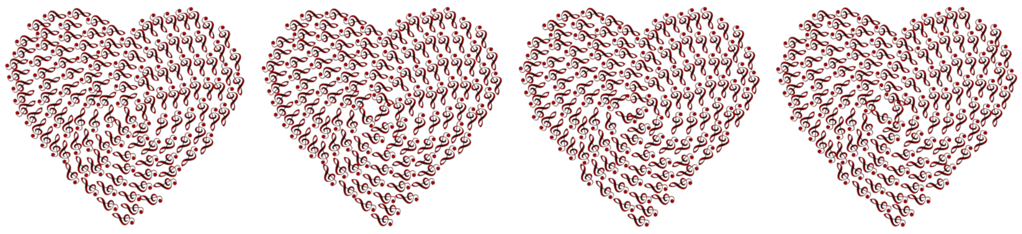
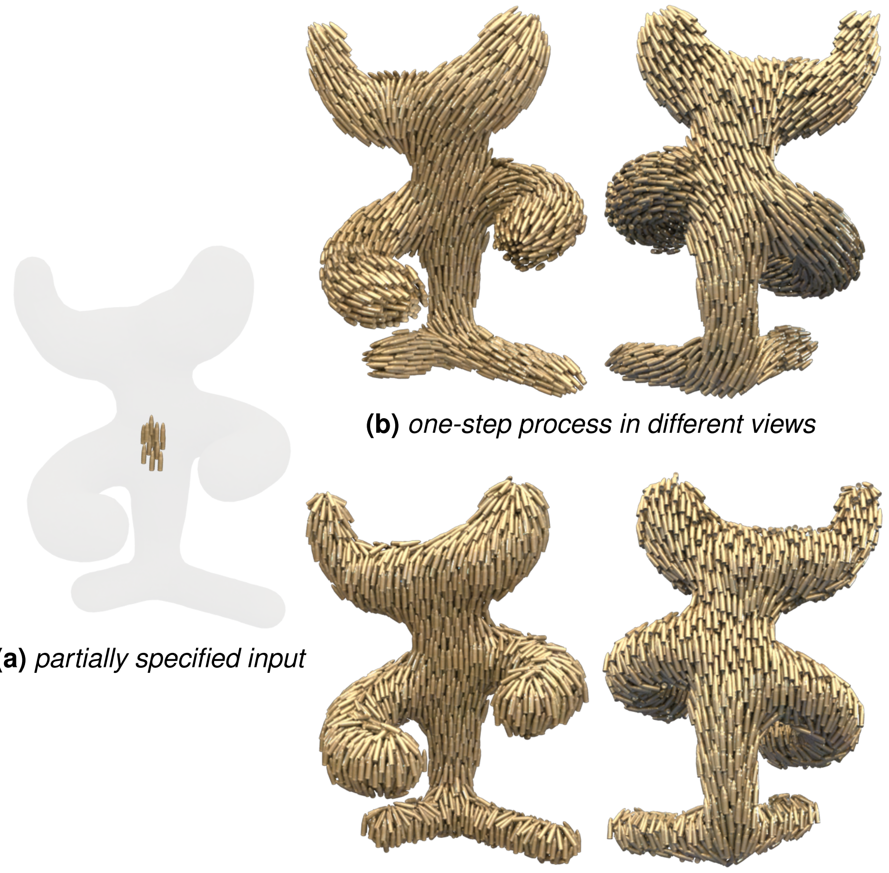

(b) *outputs without the weight $\mathrm{w}_o(s', s, i)$ via 70, 80, 90 and 100 iterations from left to right*

Figure 3.11: The effect of the adaptive weight $\mathrm{w}_o(s', s, i)$. *It can be seen that with the adaptive weight $\mathrm{w}_o(s', s, i)$ (a), the aggregate elements near the singularity (i.e. the center of the heart) can be better stabilized after 70 iterations, whereas without the adaptive weight $\mathrm{w}_o(s', s, i)$ (b), the synthesized elements around the singularity can still remain unsteady throughout. Please refer to* ***Appendix A*** *for the illustrations of in-between iterations.*

In addition, using rotation matrices can also offer more freedom and versatility as individual rotation axes can be separately aligned with specified input directions for specific purposes. For example, for aggregate elements near the output domain boundary, we can directly align the up axes of the rotation matrices of their element samples with the normals of the output domain boundary to generate corresponding results, as a normal at an output domain point near the output domain boundary is a direction vector. As illustrated in Figure 3.12, the element field formulation can effectively integrate a given partially specified input with the normals of the output domain boundary as multiple partial input fields to more smoothly match all the synthesized elements with both the output domain boundary and the given specification across the output domain.

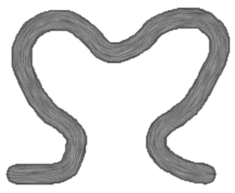


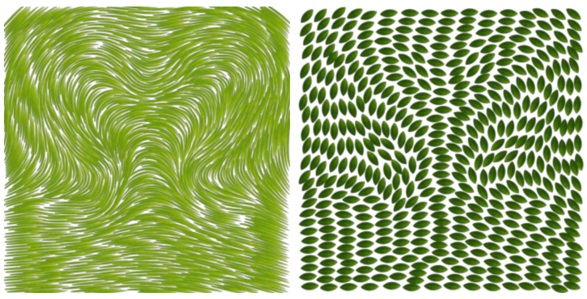
(b) *one-step process in different views*
(a) *partially specified input*
(c) *two-step process in different views*

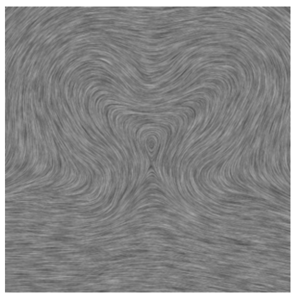
Figure 3.12: Multiple partial input fields. *By directly combining a small partially specified input (a) and the normals of the output domain boundary together, our one-step automatic optimization process (b) can effectively generate smoother element fields, whereas the two-step process (c) may lead to discontinuous element fields.*

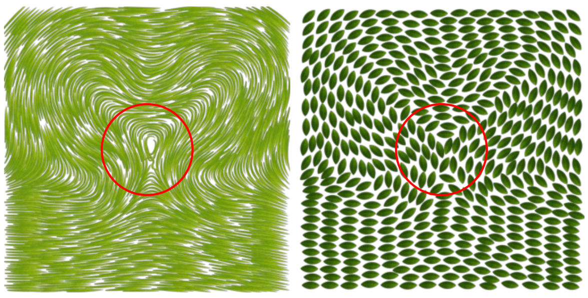
As exemplified in Figure 1.3, by directly optimizing the element distributions with the given specification together, our one-step automatic optimization process can more adaptively orient distinct types of aggregate elements and more smoothly compute complete element fields than the existing two-step process. In Figure 3.13, we show more examples about the comparison between our one-step process and the two-step process. Moreover, in addition to stabilizing the synthesized elements near the singular points (i.e. Figure 3.11), when there exist undesired singularities in the field, our element field formulation is able to directly smooth out the element arrangements around the undesired singularities without needing to change the underlying field. As demonstrated in Figure 3.14, by gradually increasing the value of the parameter α in Equation (3.12) to further enhance the influence of the field continuity E_c term for the direction field objective E_f^d, multiple singularities in the field can be effectively hidden while the overall topology of element fields can still be reasonably maintained.

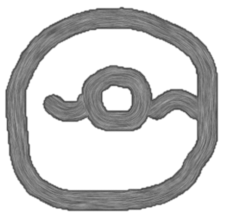
While the given input field is chaotic, we can also consider our field continuity E_c term as a denoise filter to smoothly regularize all the synthesized elements without incorporating any field preprocessing (e.g. field smoothing). As illustrated in Figure 3.15, analogous to Figure 3.14, by directly enhancing the effect of the field continuity E_c term for the direction field objective E_f^d, the degree of chaos of the overall element arrangements can be reduced correspondingly as well. Eventually, when given a sequence of dynamic input fields, our element field formulation can also well synthesize aggregate elements following such dynamic input fields frame by frame. The overall procedure for the dynamic input fields is that at the beginning, we generate the output for the first frame and utilize this output as the initialization for the next frame to compute the next output, and by adopting the same strategy, the outputs for the rest of the frames can be similarly computed via the same element synthesis process.

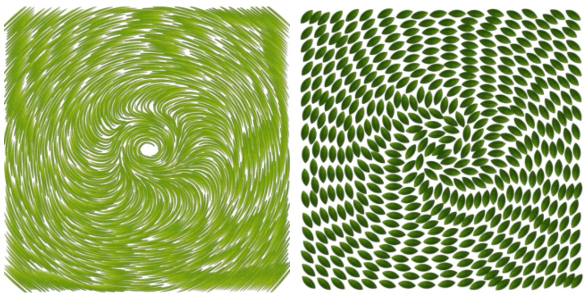

(a) *partially specified input*

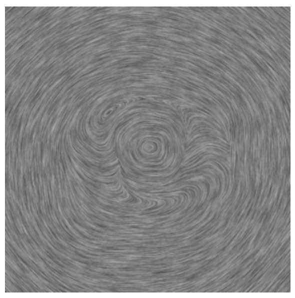

(b) *element fields computed from (a)*


(c) *full input field from (a)*


(d) *element fields computed from (c)*


(e) *partially specified input*


(f) *element fields computed from (e)*


(g) *full input field from (e)*

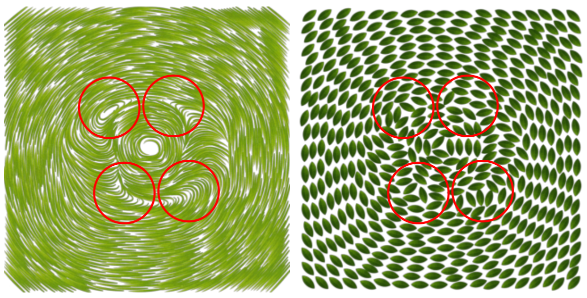
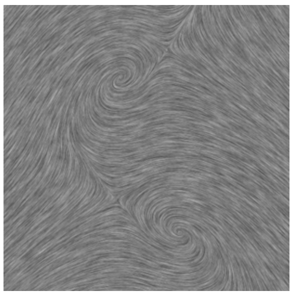
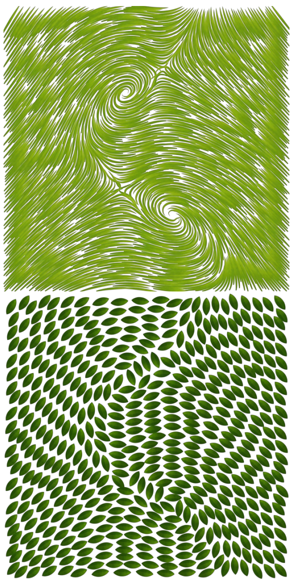

(h) *element fields computed from (g)*

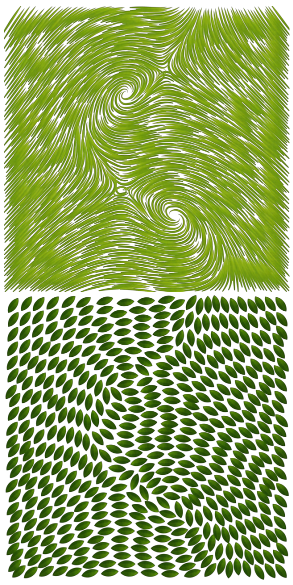
Figure 3.13: The comparison between the one-step and two-step processes. *Our one-step process can directly compute smooth element fields (b) and (f) from partially specified inputs (a) and (e), while the two-step process may create less smooth results (d) and (h). Note that the full input fields (c) and (g) are preprocessed via Laplacian smoothing as in Figure 1.3c, and the problematic regions in (d) and (h) are highlighted by red circles.*



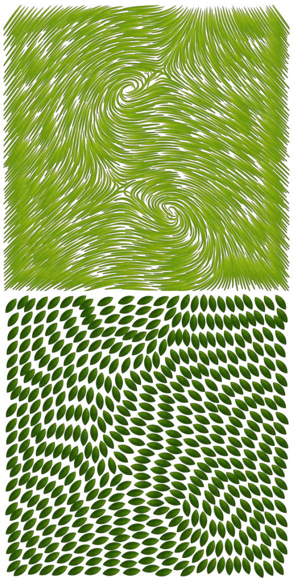
(a) *full input field*



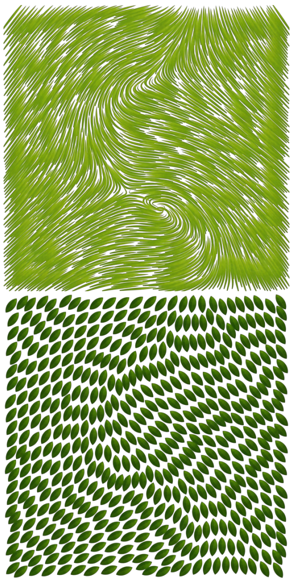
(b) α *= 0*



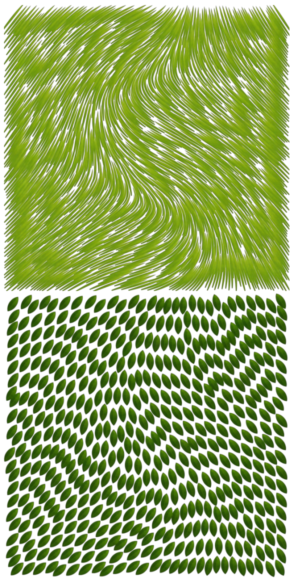
(c) α *= 5*



(d) α *= 10*

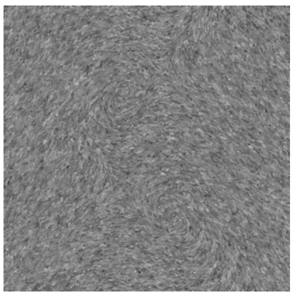


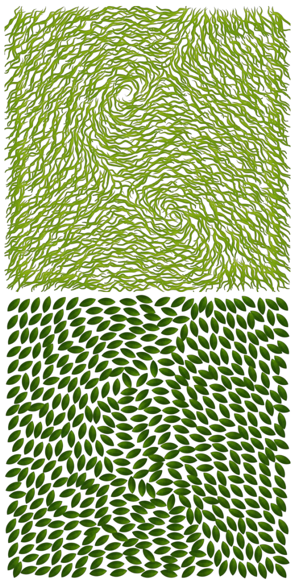
(e) α *= 15*

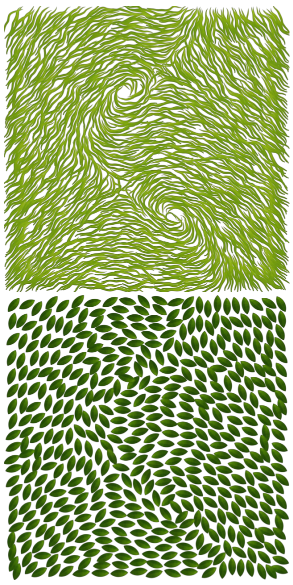


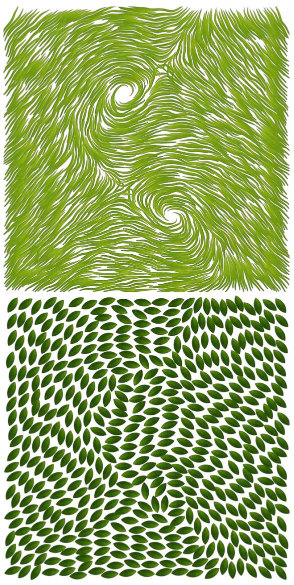
(f) α *= 20*

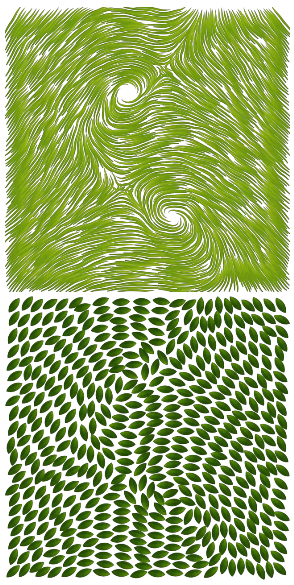
Figure 3.14: The singularity handling. *Given a full input field (a) with a few singular points, by directly tuning the value of the parameter* α *in Equation* (3.12) *to increase the effect of the field continuity* E_c *term for the direction field objective* E_f^d*, our element field formulation can directly smooth out the element arrangements across the singular points without changing the underlying field. Note that the orientations of the synthesized elements near the output domain boundary can still remain almost the same.*


(a) *full input field*


(b) α *= 0*


(c) α *= 1*


(d) α *= 2*


(e) α *= 3*

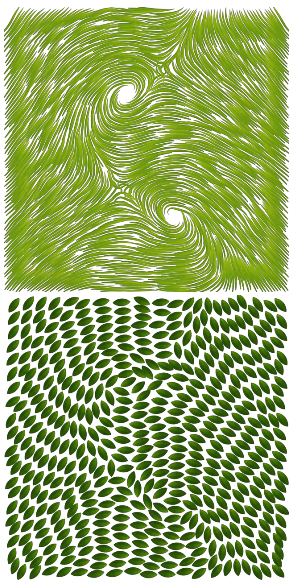
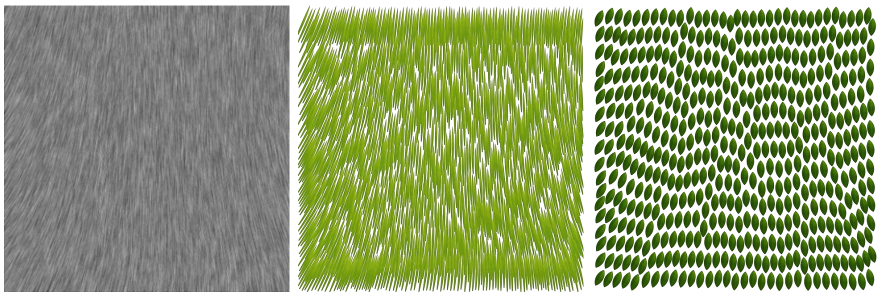
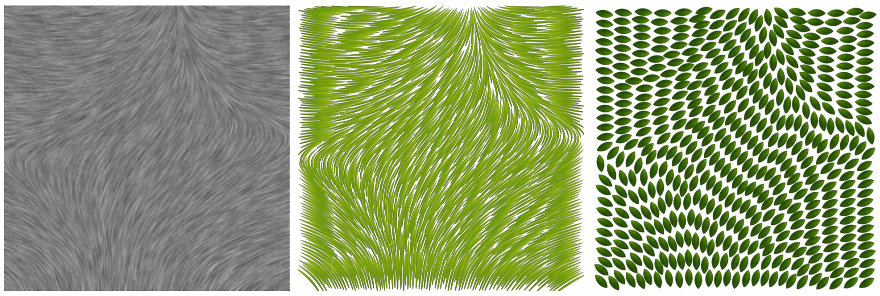
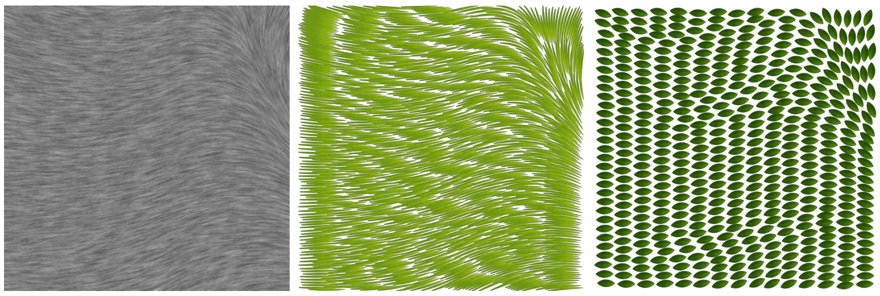

(f) α *= 4*

Figure 3.15: The chaotic input field. *The full input field (a) here is generated by directly adding random noise to the input field in Figure 3.14a. Analogous to Figure 3.14, through gradually increasing the value of the parameter* α *in Equation* (3.12) *to accordingly relief the effect of the random noise, corresponding element fields with noise reduction can be appropriately computed to comparably match the original input field in Figure 3.14a.*


(a) *frame 0*


(b) *frame 40*


(c) *frame 80*

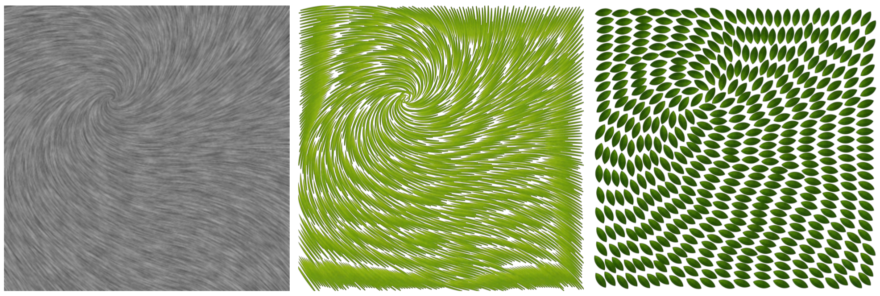

(d) *frame 120*

Figure 3.16: Dynamic input fields. *Our element field formulation can also cope with dynamic input fields. In this case, we directly compute the output for the first frame in 100 iterations while the outputs for the rest of the frames are generated in 20 iterations. Please refer to* ***Appendix B*** *for the illustrations of in-between frames.*

4 Interactive Synthesis System

In order to enrich visual diversity of artistic works and streamline work procedures for the creation of element aggregations, we aim to develop an interactive synthesis system with a palette-based brushing interface for users to iteratively design diverse output compositions for practical applications via an enjoyable and effective manner. To carry out this, we thus incorporate the idea of our autocomplete element fields (i.e. both the element distribution formulation and the element field formulation) into the interactive synthesis system and establish an intuitive and user-friendly brushing workflow for interactive authoring of aggregate elements across different output domains. In the following sections, we first detail the development specification, which includes the design rationale, software engineering and user interface of our developed system, in Section 4.1, evaluate the prototype system in Section 4.2, show our pilot user study in Section 4.3 and then demonstrate other applied results in Section 4.4.

4.1 Development Specification

Below we first recap the design rationale of our interactive synthesis system in terms of both element synthesis and interactive authoring in Section 4.1.1 and then elaborate the software engineering about implementation details in Section 4.1.2. Subsequently, the user interface of the developed system is introduced in Section 4.1.3.

4.1.1 Design Rationale

Manual placement of aggregate elements can offer users full authoring freedom, but the users can not only demand to possess essential knowledge of artistic skills and production workflows in advance but also make heavy efforts to repetitively organize individual aggregate elements with intended scales and orientations over the given output domain. During the process of iterative design, the users can consequently take significant production time and energy to completely produce satisfactory results by hand. While automatic batch computation can assist the users in the creation of element aggregations within 2D or 3D output domains, it cannot supply the users with sufficient interactivity and flexibility to appropriately transform their conceptual ideas into final design outcomes owing to algorithmic limitations or expensive computation cost, and therefore the ability and freedom of adequately generating desirable output formations according to user intention can be confined significantly.

In addition, for better element arrangements, existing element synthesis practices demand to incorporate full scalar or direction fields with their algorithms as part of optimization through a two-step process. Nevertheless, for ordinary users, it can be difficult or inconvenient to manually specify such full input fields across arbitrary output domains, especially in 3D environments. Even though the full input fields can be preprocessed by existing field interpolation algorithms, the final outputs created via the two-step process might not either properly reach the expected synthesis quality or reasonably reflect the original user intention as exemplified in Figures 1.3d, 3.12c, 3.13d and 3.13h by reason that these existing field interpolation algorithms are not specifically developed for element synthesis and do not take both inter- and intra-element relationships into consideration. While optionally operating additional field design systems for the generation of the fully specified input fields, the users may be able to ideally design the full input fields for desired element arrangements. However, it can require significant learning time from the users to obtain essential understanding of these field design systems, and this can not only break the natural artist workflow but also hinder novice users from producing artistic works.

As a majority of preceding interfaces for interactive design are mostly tailored for specific applications in 2D planes or 3D surfaces without supplying enough input interactivity and output diversity, it can be seen that to this day, there is still a lack of general user interfaces for interactive authoring of aggregate elements with varying anisotropy and deformability for multiple applications across different output domains. Moreover, in order to adequately handle these preceding interfaces, users can require a broad background of technical expertise in relevant work procedures, and the overall process of interactive authoring can also be tedious and time-consuming due to the lack of an intuitive and user-friendly production workflow. Even though it is possible to take advantage of physically-based simulation to facilitate the placement of aggregate elements (e.g. (Gay 2016a)), the users can still demand to globally specify and deploy each individual aggregate element all over the output domain by fully manual authoring. To encourage and benefit more creators, especially ordinary novices, in the creation of element aggregations, a well-developed framework that can achieve more interactive responses and simultaneously offer versatile flexibility is desirable and worth establishing, and the ideal user interface for interactive authoring of aggregate elements has to be intuitive and user-friendly enough so that the creators are able to effectively generate a variety of output formations from their design intention via an efficient and handy manner without compromising their authoring freedom.

As a result, in order to properly overcome the aforementioned challenges for better usability and interactivity, we thus develop the interactive synthesis system that aims to not only retain full user controls but also provide an efficient and yet versatile solution for users to iteratively design and explore diverse output compositions with controllable element distributions for various applications. The developed system, which combines the concept of our autocomplete element fields, can take both the inter- and intra-element relationships into account to reduce the total production workload of the users and enhance the overall synthesis quality of the outcomes via the one-step automatic optimization process. Through our palette-based brushing interface, ordinary users can easily familiarize themselves with our developed system in a short time and flexibly manipulate the distributions of aggregate elements with partial authoring without the need to learn any additional field design systems like (Palacios *et al.* 2017). With our intuitive and user-friendly brushing workflow similar to conventional painting workflows, the interactive synthesis system can indeed more naturally fit element synthesis with interactive authoring as compared with preceding approaches and better assist users in creating more interesting and novel element aggregations in high production standard without the requirement of a great amount of manual labor and technical expertise.

4.1.2 Software Engineering

There is no doubt that it can be complicated and time-consuming for developers to completely carry out a standalone interactive synthesis system from the beginning. In addition, it can also be troublesome to promote and share such a standalone system to relevant public communities, and this can more or less hinder the system developers from receiving useful and constructive feedback from existing user forums for further improvement. As a consequence, to simplify the development process of our interactive synthesis system and facilitate the promotion of the concept of our auto-complete element fields, we directly implement the proposed formulations along with the palette-based brushing interface as an external plugin into Autodesk Maya, which is a 3D computer graphics application software, to leverage its whole system manipulations, such as camera control, and functions like rendering. Overall, in our implementation, we utilize the Maya Embedded Language (MEL) for the development of our palette-based brushing interface and employ C++ with the Maya API for the fulfillment of the auto-complete element fields.

It is clear that one of the essential components for interactive authoring is to create a general brushing canvas for objects with specific shapes and sizes within the display viewport of Autodesk Maya. To achieve it, we can directly voxelize the selected object as the brushing canvas (i.e. a planar layer of voxels for 2D cases or a fully voxelized solid object for 3D cases) in the developed system, and as inspired by Kim *et al.* (2018), a sparse octree can be subsequently exploited to represent all the voxels of the selected object. Analogous to color information stored in the pixels of a 2D image, these voxels can be used to capture relevant information about given user specifications such as stroke directions and brush radii, and each voxel can also be utilized to stand for an output domain point. Note that the directions of user-specified brush strokes can be employed to denote a direction field, while the brush radii of the user-specified brush strokes can be exploited to indicate a scalar field.

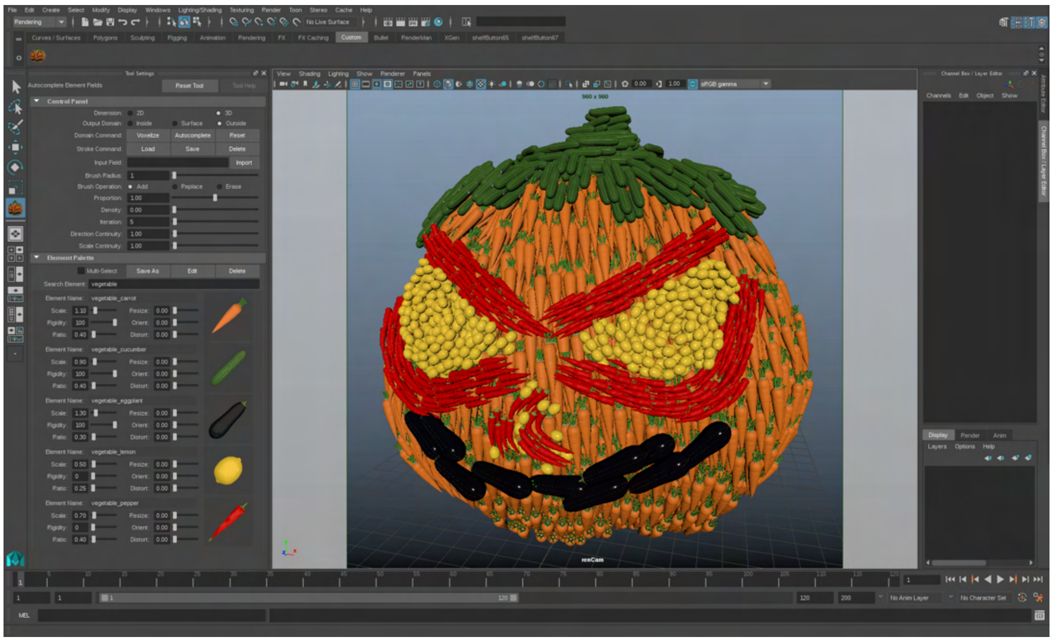
Furthermore, by adding additional fixed samples with a unit weight (i.e. the unit size of the voxel) to the boundary voxels of the selected object, we can apply the conflict check E_k term (see Section 3.3.2) to these additional boundary samples and the element samples of their neighboring aggregate elements as part of optimization. Therefore, boundary conditions for arbitrary output domains with desired shapes and sizes can be effectively dealt with under the same element synthesis process. The boundary voxels of the selected object can be similarly used to record information about the normals of the output domain boundary for further usage as exemplified in Figure 3.12. Note that in order to quickly obtain such boundary normals, instead of accurately calculating the normals of the output domain boundary from the selected object, we can adopt the Sobel filter to efficiently compute an approximation of the boundary normals according to the voxels of the output domain. Since our proposed formulations can automatically construct complete and smooth element fields on the basis of partial input fields as exemplified in Figure 3.6, such an approximation of the boundary normals can be directly handled by our synthesizer without needing to incorporate any extra processes.

Similar to a conventional brush cursor, which can be discretized by pixels in 2D images, the brush cursor with a solid sphere shape in our prototype system can be discretized by voxels as well. By adding a flag (e.g. 1 for inside, 0 for boundary, and -1 for outside) to the voxels of the brush cursor to determine the domain scope via a boolean operator, users are able to interactively brush aggregate elements within the inside, boundary or outside voxels of the output domain. More detailedly, as the brush cursor's center is located over the output domain, there can exist a portion of the brush cursor's voxels inside the output domain and another portion outside the output domain. If the flag is set to 1, aggregate elements can be directly synthesized in the brush cursor's voxels inside the output domain, while if the flag is set to -1, the users are enabled to place the synthesized elements within only the brush cursor's voxels outside the output domain. Note that it is also feasible to support a tablet with a stylus for the users to control the brush cursor in the interactive synthesis system.

Moreover, since we can similarly voxelize individual aggregate elements in preprocessing as well and record the initial number of each aggregate element's voxels in the corresponding input element exemplar, the total number of aggregate elements placed into the output domain can be proportionally estimated on the basis of the total number of the voxels of the output domain. In consequence, by alternatively increasing or decreasing the voxel count of each aggregate element, the proportion of aggregate elements synthesized in the output domain can be effectively controlled to generate sparse or dense element distributions correspondingly. About the construction of the power diagram (see Figure 3.4), we extend Voro++ (Rycroft 2009) to compute the power cells of element samples of all the synthesized elements in parallel and simultaneously establish the neighborhood information of each element sample.

4.1.3 User Interface

As illustrated in Figure 4.1, we directly develop the interactive synthesis system with our proposed formulations into Autodesk Maya, which is a popular application software in the animation and film industry, for interactive authoring of aggregate elements within 2D and 3D output domains. In the following, the major components of our palette-based brushing interface, including a brushing canvas, brush operations, a control panel and an element palette, are introduced respectively.



File Edit Create Select Modify Display Windows Lighting/Shading Texturing Render Toon
Rendering
Curves / Surfaces Polygons Sculpting Rigging Animation Rendering FX
Tool Settings
Reset Tool
Tool Help
View
MEL

Autocomplete Element Fields

Control Panel

Dimension: ○ 2D ● 3D
Output Domain: ○ Inside ○ Surface ● Outside
Domain Command: Voxelize | Autocomplete | Reset
Stroke Command: Load | Save | Delete
Input Field: | Import
Brush Radius: 1
Brush Operation: ● Add ○ Replace ○ Erase
Proportion: 1.00
Density: 0.00
Iteration: 5
Direction Continuity: 1.00
Scale Continuity: 1.00

Element Palette

Multi-Select | Save As | Edit | Delete
Search Element: vegetable

Element Name	Scale	Rigidity	Ratio	Resize	Orient	Distort
vegetable_carrot	1.10	100	0.40	0.00	0.00	0.00
vegetable_cucumber	0.90	100	0.40	0.00	0.00	0.00
vegetable_eggplant	1.30	100	0.30	0.00	0.00	0.00
vegetable_lemon	0.50	0	0.25	0.00	0.00	0.00
vegetable_pepper	0.70	0	0.40	0.00	0.00	0.00

Figure 4.1: The user interface. *The palette-based brushing interface is integrated with Autodesk Maya. Both the element palette and the control panel are on the left, and the brushing canvas is in the middle.*

4.1.3.1 Brushing Canvas

Similar to common painting interfaces, a brushing canvas is established for users to interactively arrange user-picked element collections within the domain scope, and as described in Section 4.1.2, the brushing canvas (i.e. the voxelized object) can be a 2D plane, a 3D surface, or a 3D volume with a specific domain shape and size in our developed system. As a result, aggregate elements are able to be intuitively brushed by the users over the brushing canvas to create results that can well match the given domain shape and size.

4.1.3.2 Brush Operations

In the developed system, we offer *add*, *erase*, and *replace* brush operations for interactive authoring. The *add* brush operation enables users to handily place a collection of aggregate elements over the output domain, while the *erase* brush operation can remove the already placed aggregate elements. The *replace* brush operation is a combination of both the *erase* and *add* brush operations. For automatic completion, the partially user-specified brush strokes can be employed to imply the intended element orientations and scales for the whole synthesized elements, and instead of keeping these brush strokes unchanged, our interactive synthesis system globally optimizes all the element distributions, orientations and scales together according to their inter- and intra-element relationships to better produce smooth element fields while adequately observing the original user intention. Based on our experiment, this can more effectively utilize the entire output domain space to improve overall synthesis quality. For example, when there exists a small gap space between two brush strokes, if the brush strokes are fixed, such a gap space may not be properly covered by the synthesized elements, and thus the overall element distributions may not reach satisfactory output standard.

4.1.3.3 Control Panel

The control panel aims to offer users essential parameters and functions for global element synthesis controls as demonstrated in Figure 4.2. For instance, the users can freely adjust the brush radius to proportionally enlarge or shrink the overall element sizes of all the selected input element exemplars, set the total number of iterations used in the optimization process or directly press an autocomplete button to generate complete outputs from partially specified inputs via the one-step process. Additionally, through this control panel, the users are able to specifically determine the brush operations, the domain scope (i.e. the output domain's inside, boundary or outside voxels) and the total number of aggregate elements placed into the output domain as well.

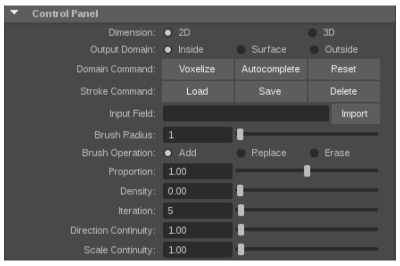
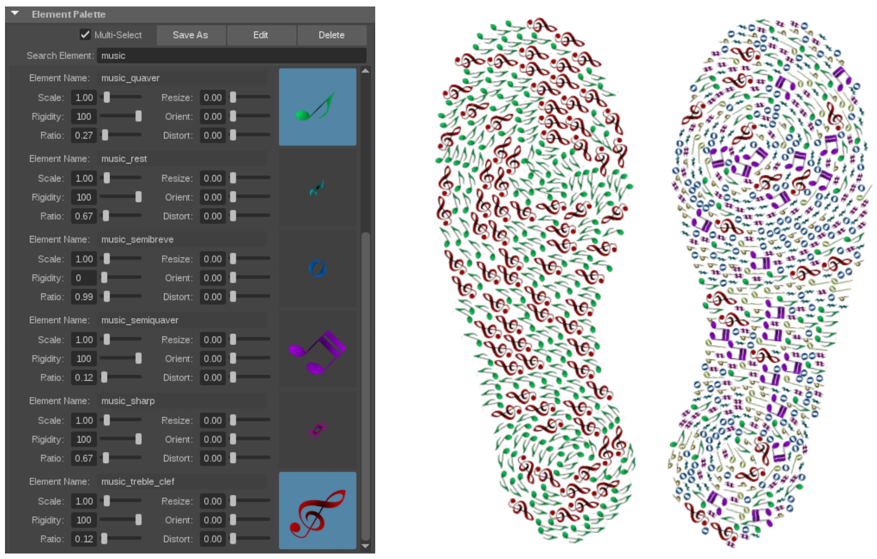

Control Panel
Dimension: [✓ 2D] 3D
Output Domain: [✓ Inside] Surface Outside
Domain Command: Voxelize Autocomplete Reset
Stroke Command: Load Save Delete
Input Field: Import
Brush Radius: 1
Brush Operation: [✓ Add] Replace Erase
Proportion: 1.00
Density: 0.00
Iteration: 5
Direction Continuity: 1.00
Scale Continuity: 1.00

Figure 4.2: The control panel. *A number of controllable parameters and relevant functions for global element synthesis controls are provided in the control panel.*

The controllable parameters for manipulating α in Equation (3.12) and β in Equation (3.13) are also supplied in the control panel, and thus the influence of the field continuity E_c term in the element field formulation (see Section 3.4.2) can be optionally increased or decreased by users in the interactive synthesis system. As exemplified in Figure 3.14, by directly tuning the related parameters from the control panel, it can encourage the users to flexibly produce outcomes with smoother or less smooth element orientations and scales on the basis of the same user specifications without the necessity of any additional field preprocessing. Moreover, with the control panel, the users can further delete multiple undesired brush strokes, reset the entire output domain or even import a predefined input field for element synthesis if there is specific demand.

4.1.3.4 Element Palette

As illustrated in Figure 4.3, through directly clicking the image icons of individual aggregate elements from the element palette, a variety of input element exemplars listed in the element palette can be freely selected and mixed by users.



[✓ Multi-Select]
Search Element: music

Element Name: music_quaver
Scale: 1.00 Resize: 0.00
Rigidity: 100 Orient: 0.00
Ratio: 0.27 Distort: 0.00

Element Name: music_rest
Scale: 1.00 Resize: 0.00
Rigidity: 100 Orient: 0.00
Ratio: 0.67 Distort: 0.00

Element Name: music_semibreve
Scale: 1.00 Resize: 0.00
Rigidity: 0 Orient: 0.00
Ratio: 0.99 Distort: 0.00

Element Name: music_semiquaver
Scale: 1.00 Resize: 0.00
Rigidity: 100 Orient: 0.00
Ratio: 0.12 Distort: 0.00

Element Name: music_sharp
Scale: 1.00 Resize: 0.00
Rigidity: 100 Orient: 0.00
Ratio: 0.67 Distort: 0.00

Element Name: music_treble_clef
Scale: 1.00 Resize: 0.00
Rigidity: 100 Orient: 0.00
Ratio: 0.12 Distort: 0.00

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

Figure 4.3: The element palette. *Through the element palette (a), several kinds of input element exemplars (i.e. music symbols) can be chosen by users as the input primitives to form a shoeprint consisting of only two types (b) or eight types (c). For each input element exemplar, the users can respectively set its initial element size and deformability by adjusting the* scale *and* rigidity *parameters and individually manage the total proportion of this kind of aggregate elements placed into the output domain via the* ratio *parameter. In addition, while manipulating the* resize *and* orient *parameters, the scales and orientations of the synthesized elements can be accordingly randomized as exemplified in Figure 3.9. To further enhance visual diversity of artistic works, the* distort *parameter can be tuned to arbitrarily twist the element shapes (see Figure 4.10c).*

During the iterative design procedure, users can choose one of the input element exemplars to generate simple outcomes consisting of the same kind of aggregate elements or mix multiple input element exemplars to create novel mixtures composed of different kinds of aggregate elements. Analogous to common color palettes (like in (Shugrina *et al.* 2017)), the combination of user-picked element collections can be subsequently saved as another new input entry into the element palette for further reuse and remix, so the users are able to customize their own element palettes in an intuitive and flexible manner. Like tunable color properties (e.g. hue, intensity and saturation) in a common color palette, in our element palette, there are also a few manageable element properties (e.g. initial element size and deformability) for the users to freely control, and through optionally manipulating the properties of individual aggregate elements, it can encourage the users to produce more interesting element aggregations. Furthermore, in order to quickly search and choose relevant aggregate elements from the element palette, the users are enabled to automatically filter irrelevant input element exemplars from the element palette by directly entering the corresponding names of the desired aggregate elements into a text field.

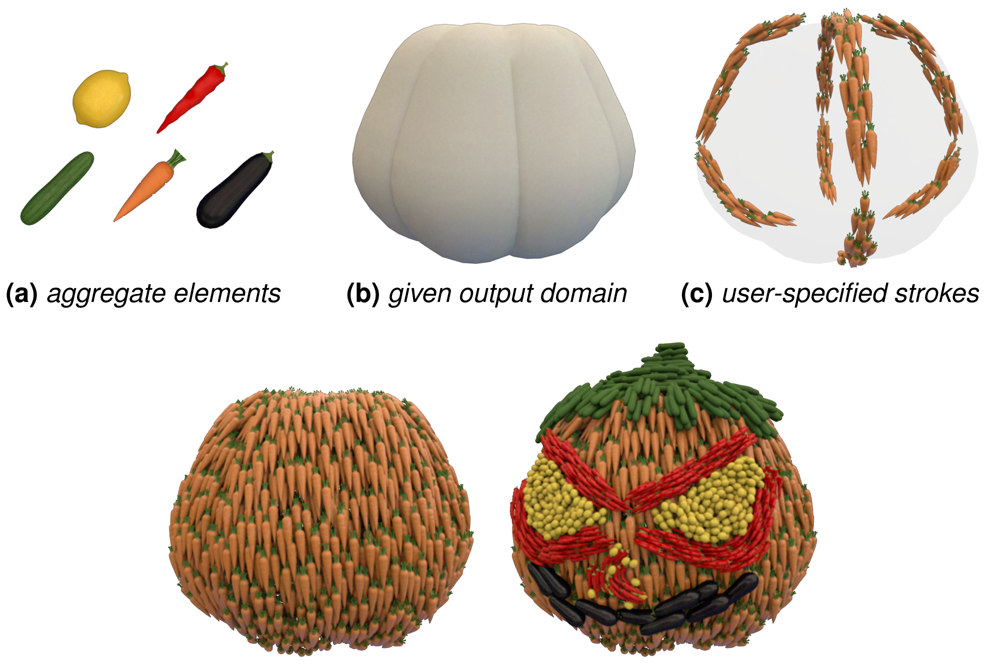
Finally, one thing to note here is that in our current prototype, the input element exemplars listed in the element palette are stored in a designated folder, and the developed system can automatically import all the input element exemplars from the folder into the element palette. To add new input element exemplars to the element palette, users have to manually collect essential information about new aggregate elements (i.e. source models, image textures and shaders) into the designated folder. Obviously, it can be desirable to develop an advanced interface that can simplify the process of the addition and removal of different input element exemplars or even assist the users in classifying and exploring existing input element exemplars like in (Chen *et al.* 2016), and we consider this as a potential future task.

4.2 System Evaluation

In this section, we first describe the iterative design procedure about the creation of element aggregations and simultaneously evaluate the versatility of our interactive synthesis system by demonstrating more diverse and compelling results. Subsequently, the performance of element synthesis about our representative results and the limitations of the developed system are reported and discussed respectively.

4.2.1 Iterative Design Procedure

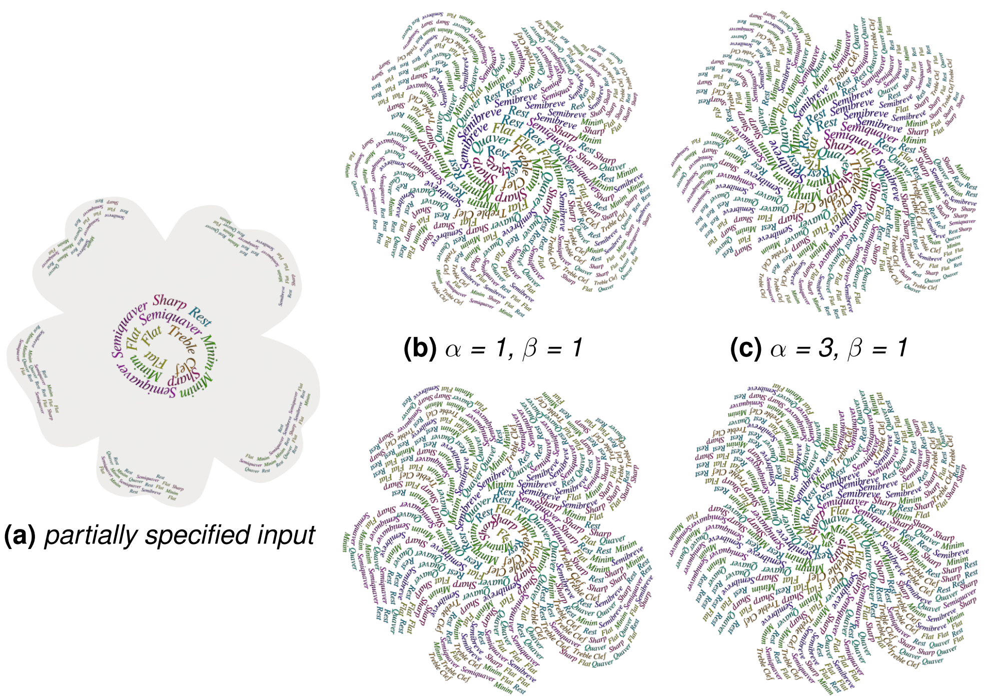
During the process of iterative design, users can freely choose single or mix multiple aggregate elements from the element palette, optionally tune relative parameters from the control panel, and directly perform corresponding brush operations to interactively arrange the user-picked element collections over the brushing canvas for the production of artistic works. In general, for better production efficiency, at the beginning, the users can synthesize a certain number of aggregate elements and adopt fewer iterations (e.g. 5 to 20 iterations) to quickly generate basic sample layouts as initial previews on the basis of given user specifications. Following that, when the sample layouts can reach the users' expected design outcomes, the users can then specify a desired proportion of the synthesized elements and utilize more iterations (e.g. 50 to 100 iterations) to optimally produce the final intended results. At the end of the process of iterative design, further arrangements such as the removal or replacement of undesired aggregate elements can be subsequently carried out. Additionally, if there is need, far more iterations can also be available for the users to cope with more complicated scenarios. In the interactive synthesis system, the fundamental brushing workflow about the creation of element aggregations can be seen in Figure 4.4.


(a) *aggregate elements* **(b)** *given output domain* **(c)** *user-specified strokes*

(d) *automatic completion* **(e)** *final arrangement*

Figure 4.4: The brushing workflow. *Users can select desired aggregate elements (a) from the element palette, directly brush the selected elements within the boundary voxels of a given output domain (b) and interactively see the corresponding results (c). Based on the stroke directions and brush radii derived from the partial user specifications (c), the developed system can automatically complete the full output (d) with intended element orientations and scales. Subsequently, the users can further arrange the aggregate elements over the outside voxels of the output domain for the final outcome (e) via corresponding brush operations.*

As our control panel provides the controllable parameters α and β for users to optionally manipulate the influence of the field continuity E_c term for both the direction field objective E_f^d and the scalar field objective E_f^s, the users can flexibly produce output compositions with smoother or less smooth element orientations and scales according to their personal preferences. As illustrated in Figure 4.5, when given partially user-specified brush strokes, by directly tuning the values of the parameters α and β from the control panel, the final intended results with different appearances can be correspondingly generated on the basis of the same user specifications. This feature of being able to let users manipulate the topology of element fields can not only supply more flexibility for the iterative design procedure but also significantly assist the users in designing more satisfactory artworks without breaking the natural artist workflow. As a result, through our interactive synthesis system, ordinary users neither need to learn any additional field design systems (like in (Zhang *et al.* 2007; Palacios *et al.* 2017)) nor become algorithmic experts in the interpolation process of full input fields.



(a) *partially specified input*

(b) α *= 1,* β *= 1*

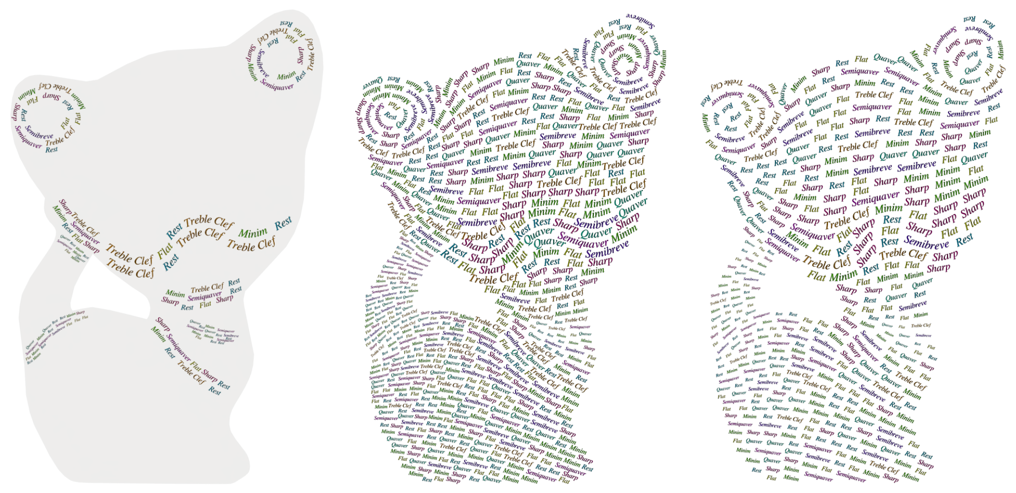
(c) α *= 3,* β *= 1*

(d) α *= 1,* β *= 10*

(e) α *= 3,* β *= 10*

Figure 4.5: The manipulation of the influence of the field continuity E_c term. *By reasonably increasing the values of the parameters* α *and* β *via the control panel, it can encourage smoother element orientations and scales. Our interactive synthesis system can observe the original user intention (a) (i.e. partially user-specified brush strokes) to optimize overall synthesis quality with the tuned parameter values accordingly.*

While iteratively designing a large output, it can be desirable and even essential for users to be able to flexibly manipulate the distributions of aggregate elements for the final output and alternatively acquire a variety of visual effects and artistic styles, such as compact or loose output formations. Since the total number of aggregate elements placed into the output domain can be directly determined by the users via the control panel, our interactive synthesis system can enable the users to straightforwardly generate various output compositions with dense or sparse element distributions via the same element synthesis process. As demonstrated in Figure 4.6, after partially brushing a few strokes of aggregate elements over the output domain, the developed system can alternatively create compact or loose element aggregations from the same partially user-specified strokes according to the users' favorites.

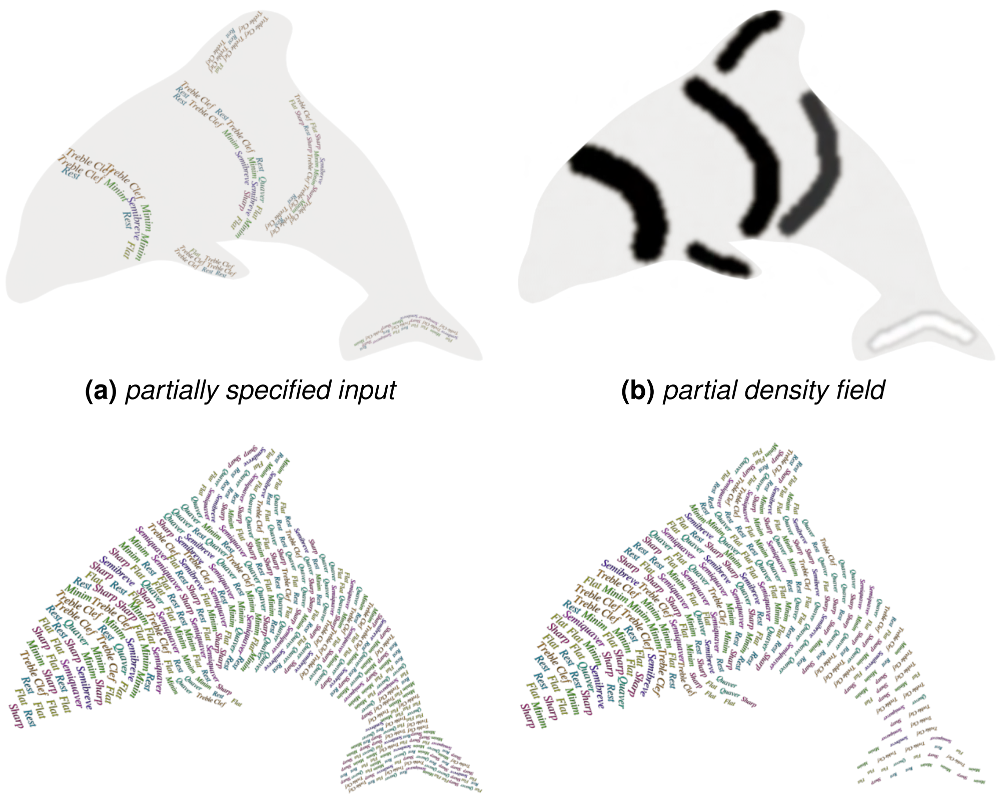


(a) *partially specified input* **(b)** *dense words* **(c)** *sparse words*

Figure 4.6: The manipulation of dense and sparse element distributions. *Based on the same partially user-specified brush strokes (a), by directly adjusting the total number of aggregate elements placed into the output domain, our interactive synthesis system can densely (b) or sparsely (c) distribute the aggregate elements (i.e. words) across the entire output domain and simultaneously well match all the synthesized elements with the given user specifications.*

Moreover, as exemplified in Figure 3.8d, by incorporating an additional density field as part of optimization, our element distribution formulation can effectively deal with output compositions with spatially varying element distributions. However, similar to the fully specified direction or scalar fields, to acquire such a complete density field within different output domains can be difficult or inconvenient for users as well. Therefore, in order to streamline work procedures for better usability and interactivity, the interactive synthesis system can further enable the users to interactively specify partial density fields across different output domains via the same brush operations under the same brushing workflow. To accomplish this, in our current prototype system, each brush stroke can be optionally equipped with a density value among 0 and 1, and the users can interactively brush multiple strokes with specified density values over the output domain to acquire a partial density field and then let our synthesizer spatially vary the distributions of aggregate elements according to the partially specified density field. Like stroke directions and brush radii, this extra information about the density values of brush strokes can also be directly captured in the voxels of the output domain, and the density value of each brush stroke can be individually determined by the users via the control panel.

As demonstrated in Figure 4.7, through directly equipping individual brush strokes with relative density values from the control panel, our interactive synthesis system can allow users to interactively specify a partial density field within the given output domain and then correspondingly generate an output composition with spatially varying element distributions based on the partially user-specified density field in an efficient and handy manner. In consequence, in terms of the iterative design procedure, this manipulation ability can not only provide more usability and interactivity for users to produce desired element aggregations but also encourage the users in enriching visual diversity of artistic works and increase creativity and productivity.

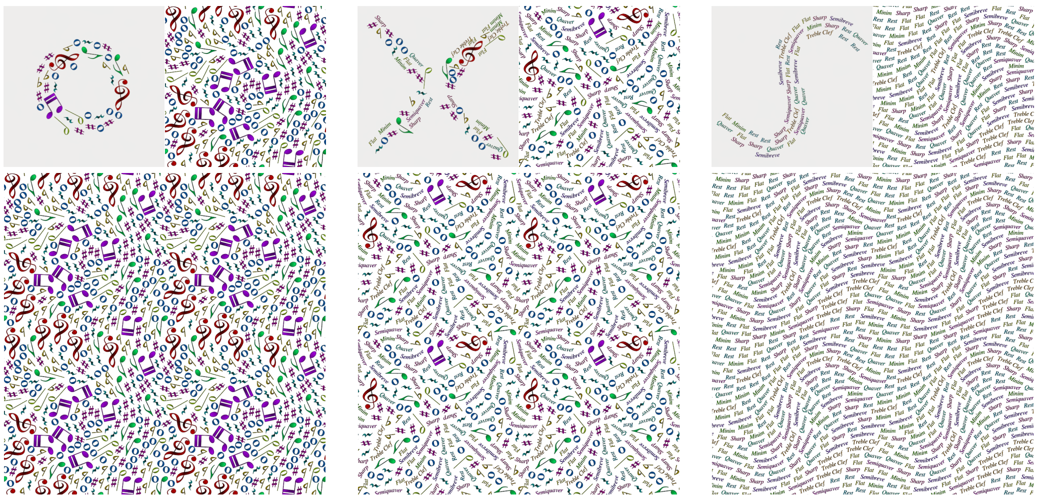


(a) *partially specified input* **(b)** *partial density field*

(c) *uniform element distributions* **(d)** *spatially varying element distributions*

Figure 4.7: The manipulation of spatially varying element distributions. *Users can interactively specify brush strokes (a) with density values to acquire a partial density field (b) over the output domain. Without adopting the density field (b), aggregate elements can be distributed uniformly (c). While incorporating the density field (b) as part of optimization, the distributions of the synthesized elements can be varied spatially (d).*

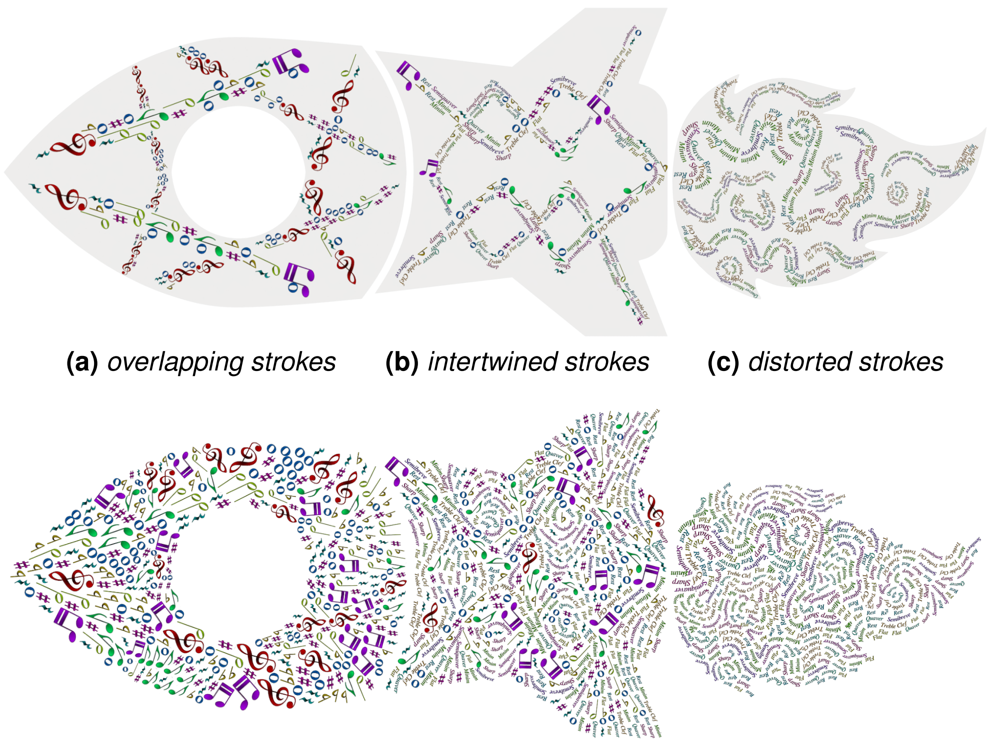
While taking toroidal boundary conditions into consideration, our synthesizer can be further applied to the generation of 2D and 3D element tiles that can seamlessly tile a large output domain. As illustrated in Figure 4.8, rigid and deformable elements with anisotropic shapes and various sizes can be properly synthesized to form mixtures with tile-based element distributions according to given user specifications. It is obvious that the capability of directly manipulating the distributions of aggregate elements via our developed system can offer users higher functionality since existing practices do not take user-specified element distributions into account, and thanks to the versatility of our interactive synthesis system, it can also be achievable for the users to interactively create these tileable output compositions with dense, sparse or spatially varying element distributions via the same element distribution manipulation.



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

Figure 4.8: The manipulation of tile-based element distributions. *Aggregate elements with varying anisotropy and deformability can be properly mixed to generate seamless element tiles on the basis of the partially user-specified brush strokes. Note that in (b), the rigid elements (i.e. music symbols) are randomly aligned, while the deformable elements (i.e. words) follow the stroke directions.*

In addition to the manipulation of user-specified element distributions described above, the freedom of interactive authoring can also be crucial, as it can directly reflect design intention and affect final output formations. Since our element field formulation can effectively handle chaotic input fields as exemplified in Figure 3.15, by taking advantage of this ability, the interactive synthesis system can enable users to arbitrarily specify brush strokes with incoherent directions or sizes to produce stylized or notable results without compromising their brushing behavior. As illustrated in Figure 4.9, different kinds of incoherent user strokes can be freely specified by the users within the output domain, and our synthesizer can still appropriately match all the synthesized elements with such incoherent user strokes to smoothly generate incoherent element fields. This characteristic can indeed offer users sufficient freedom of interactive authoring and clearly demonstrate that our interactive synthesis system can outperform existing practices in terms of usability and interactivity since previous methods (e.g. (Maharik *et al.* 2011; Saputra *et al.* 2017)) only deal with coherent inputs (e.g. fully smooth input fields) for element arrangements without taking incoherent user specifications into consideration.



(a) *overlapping strokes* **(b)** *intertwined strokes* **(c)** *distorted strokes*

(d) *incoherent element fields computed from (a), (b) and (c) respectively*

Figure 4.9: Incoherent user strokes. *Our interactive synthesis system can allow users to freely specify overlapping (a), intertwined (b) or even distorted (c) brush strokes with incoherent directions or sizes to produce stylized or notable results (d) without restricting their authoring freedom, whereas it can be difficult if not impossible for the users to effectively generate such results through existing practices.*

Furthermore, while user specifications (i.e. brush strokes or input fields) are not supplied, our synthesizer is still able to generate diverse output compositions by randomly manipulating the initial orientations and scales of aggregate elements as exemplified in Figure 3.9. Through our element palette, users can arbitrarily orient, resize or even twist the selected element collections and let our interactive synthesis system automatically create complete element aggregations. As demonstrated in Figure 4.10, the developed system can properly form a variety of novel mixtures composed of aggregate elements with random orientations or scales in high synthesis quality without the need to incorporate any additional schemes (e.g. deformation simulation). Although there may exist a few overlaps between the synthesized elements, it can be observed that the overall inter-element penetrations are not visually obvious in the outputs.

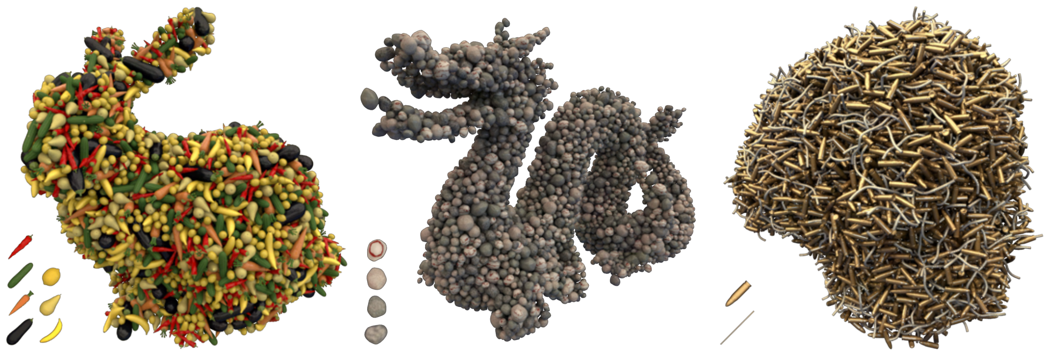
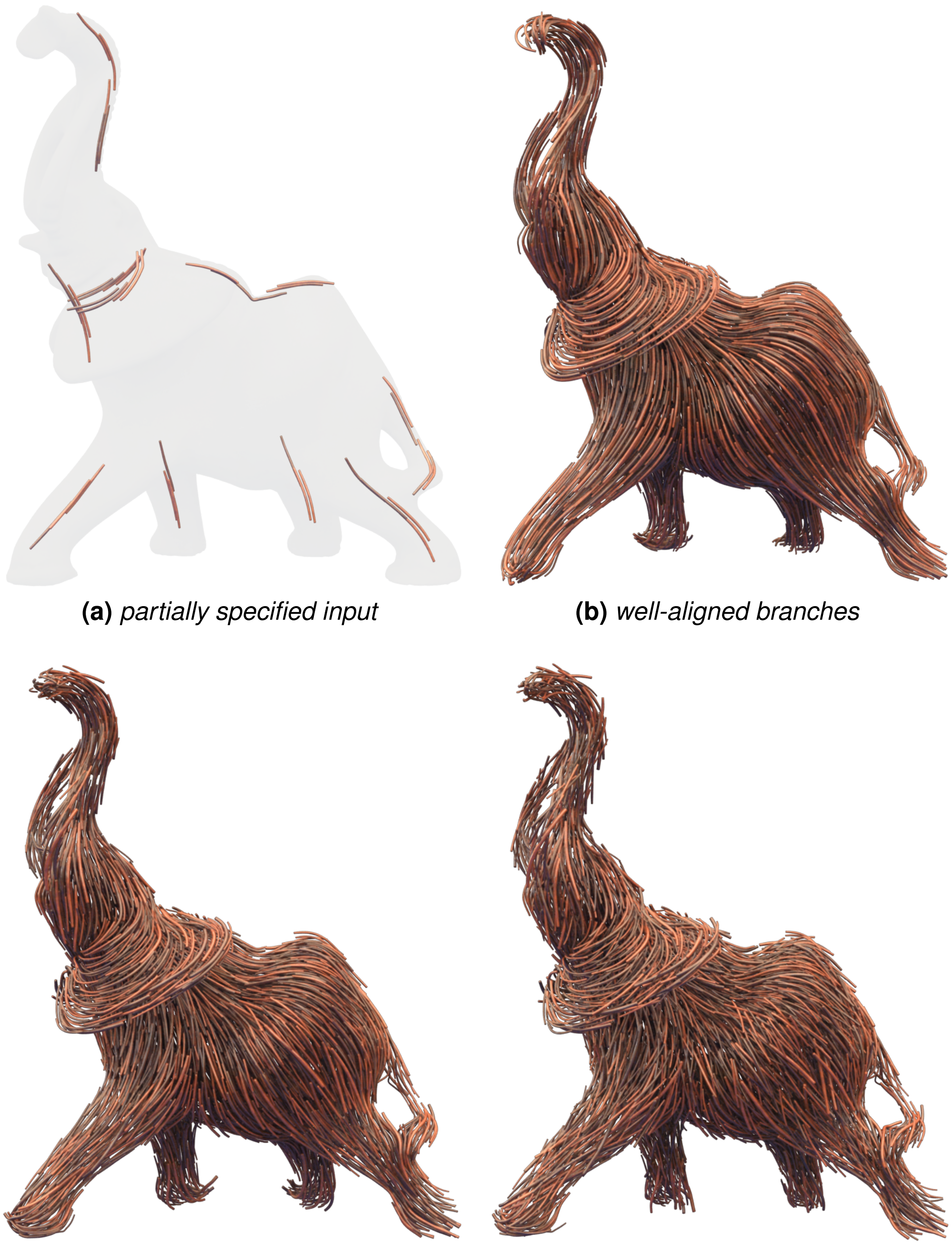

(a) *aggregate vegetables* **(b)** *piled pebbles* **(c)** *tangled metals*

Figure 4.10: The manipulation of the initial orientations and scales of aggregate elements. *Through individually manipulating the initial orientation and scale of each selected aggregate element from the element palette, compelling mixtures consisting of multiple aggregate elements with chaotic alignments (a), arbitrary orientations and scales (b) or even twisted shapes (i.e. metal wires) (c) can be correspondingly formed by users in 3D volumes. Note that in (c), to twist the shapes of the metal wires, each element sample of the metal wires is further assigned another extra random orientation, which can also be determined by the users via our element palette.*

This feature can significantly relieve the demand of collecting or modeling a host of aggregate elements, assist users in broadening the formations of element aggregations, and more increase the versatility of our interactive synthesis system. In particular, to our best knowledge, there is a lack of efficient and straightforward avenues for the creation of such mixtures from user-specified aggregate elements as well. Moreover, we can also extend this feature to further enhance the visual effects of output formations. For instance, as illustrated in Figure 4.11, when given user specifications, users can still slightly randomize the initial orientations of aggregate elements through the element palette, and our interactive synthesis system can automatically compute more or less misaligned element fields across the entire output domain and accordingly match all the synthesized elements with the original user intention at the same time.

To sum up, it is evident that the developed system can supply users with functional manipulations to facilitate the iterative design procedure, and due to the intuitiveness and user-friendliness of our brushing workflow, the users are enabled to produce compelling outcomes with controllable element distributions, orientations and scales as demonstrated above. While our current prototype system is generalized enough to effectively cope with a variety of phenomena and output formations, this property can make further extension and potential enhancement more practicable for particular scenarios and applications. For example, being able to directly place a few fixed key aggregate elements in the output domain as partial inputs for optimization can be considered as an alternative production workflow, and to allow the users to design specific element alignments around the output domain boundary (e.g. only part of the synthesized elements perpendicular to the boundary normals) is also worth developing. Thus, to more leverage our proposed formulations, it is considerable to incorporate more design constraints into our interactive synthesis system for future improvement.



(a) *partially specified input* **(b)** *well-aligned branches*

(c) *less-misaligned branches* **(d)** *more-misaligned branches*

Figure 4.11: The manipulation of misaligned element fields. *When given several partially user-specified brush strokes (a), well-aligned aggregate elements (b) can be directly synthesized by our developed system across the entire 3D volume. While slightly randomizing the initial orientations of the individual aggregate elements from the element palette, the synthesized elements can be misaligned less (c) or more (d) but still correspondingly follow the original user intention.*

4.2.2 Performance

The synthesis times of our representative results shown in this thesis are reported for reference in Table 4.1. In our current prototype system, there exists room to improve the performance. For example, as we extend Voro++ (Rycroft 2009), which is developed for 3D output domains, for the construction of the power diagram, it should be able to specifically optimize the code framework of Rycroft (2009) to reduce the computation of the power cells of element samples in 2D output domains. We leave it for future work.

2D case	# elements	# samples	time	# iterations
Figure 1.4a	725	1656	9s	100
Figure 4.5b	303	1276	15s	200
Figure 4.6b	634	2629	144s	1000
Figure 4.6c	436	1810	94s	1000
Figure 4.7c	381	1560	8s	100
Figure 4.7d	274	1139	7s	100
Figure 4.8a	187	402	4s	200
Figure 4.8b	194	650	6s	200
Figure 4.8c	206	838	8s	200
Figure 4.9a	367	849	5s	100
Figure 4.9b	516	1675	18s	200
Figure 4.9c	355	1477	28s	400

3D case	# elements	# samples	time	# iterations
Figure 1.4b	3994	19970	28s	5
Figure 1.4c	6280	43960	170s	20
Figure 4.10a	5272	15931	60s	50
Figure 4.10b	6158	16966	101s	50
Figure 4.10c	5344	26344	145s	50
Figure 4.11b	4888	48880	150s	10

Table 4.1: The synthesis timing. *The CPU we use is Intel® Xeon® E5-1650 3.20GHz. Note that Figures 4.5b to 4.5e have similar synthesis times and the same numbers of aggregate elements and element samples, Figures 4.9a to 4.9c here represent the corresponding outputs in Figure 4.9d respectively, and Figures 4.11b to 4.11d also have similar synthesis times as well as the same numbers of aggregate elements and element samples.*

4.2.3 Limitations

Since our synthesizer randomly places distinct types of aggregate elements into the output domain for fast initialization, the optimization process for the distributions of aggregate elements might be trapped in a local minimum when the synthesized elements are highly anisotropic and the output domain is irregular. For instance, as shown in Figure 1.4a, the numbers of aggregate elements within the four feet of the gecko (i.e. the output domain) are slightly different from each other, but this can be relieved via a progressive initialization like (Davison *et al.* 2019) or a teleportation scheme like (Cohen-Steiner *et al.* 2004) for the dynamic addition and removal of aggregate elements. In addition, our current prototype system cannot entirely avoid inter-element penetrations, but the issue does not result in visually obvious artifacts to our outcomes shown in this thesis. If there is requirement, similar to (Hsu and Keyser 2010, 2012), it is also feasible to incorporate additional physics solvers (e.g. (Coumans 2013)) into the developed system to overcome this issue.

Unlike (Xing *et al.* 2014, 2015; Peng *et al.* 2018), which only demand to deal with a few number of sketched or geometric elements within a small region at once, our interactive synthesis system aims to automatically compute a complete element field and thus may lack interactive speed for large outputs (e.g. Figure 4.11). One possibility is to display the intermediate iteration progress of the optimization process (like in (Saputra *et al.* 2018)) instead of only the final result at the end. Furthermore, while the synthesizer is currently carried out based on CPU computing, to investigate GPU computing for further speedup can be an alternative solution for more immediate interactivity.

4.3 User Study

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

4.3.1 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.

4.3.1.1 Warm-up Session

This session aims to assist the novice users in learning basic understanding and manipulation of Autodesk Maya (e.g. camera control) and familiarize all the participants with our interactive synthesis system. The process consists of interactive authoring and automatic completion for given objects in both 2D and 3D output domains. Through the session, we supplied the participants with relevant assistance, and thus each participant could properly experience our brushing workflow and realize relative parameters as well as essential functions.

4.3.1.2 Target Brush Session

In this session, we aim to measure the usability of the palette-based 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. Although the 2D target (Figure 4.13b) can be achieved by Adobe Photoshop or Illustrator (e.g. Levis (2014); VideoLot (2016)) and the 3D target (Figure 4.14a) can be fulfilled by PhysX Painter (e.g. Gay (2016b)) plus manual placements, it can still require significant artistic skills and manual labor from users in a time-consuming procedure. As a consequence, instead of asking the participants to accomplish these designated results by existing tools, we let our participants directly create outcomes through our developed system under two conditions: autocomplete off (i.e. synthesizing aggregate elements only under the user strokes) and autocomplete on. We then recorded the production time and the total number of user strokes for each task respectively.

4.3.1.3 Open Brush Session

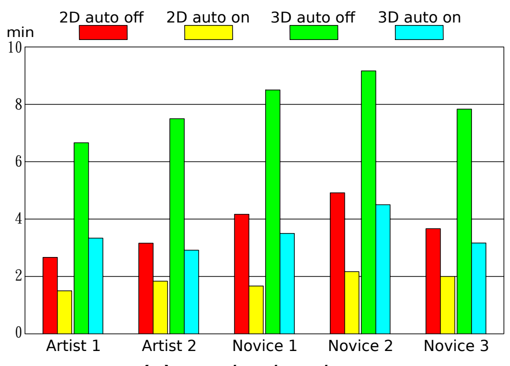
In order to identify possible enhancement for better usability and interactivity, the participants were encouraged to freely create other element aggregations during this period, and at the same time, we observed their brushing behavior and production workflow throughout the process. In the session, we also offered the participants necessary supports to assist them in exploring more interesting experiments.

4.3.1.4 Final Interview Session

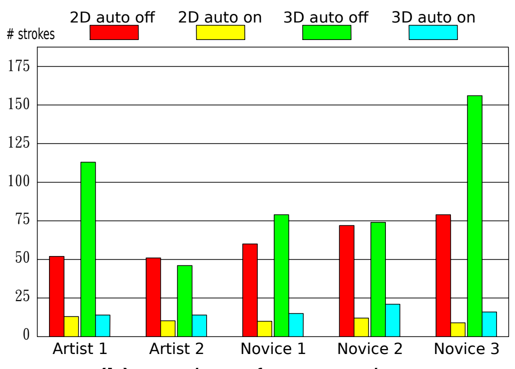
In the final interview, we discussed conceivable usage and potential improvement about our interactive synthesis system with the participants, and each participant was also asked a few user feedback questions, such as desirable manipulations for the autocomplete element fields and other probable applications about the developed system. At the end, we requested our participants to mark the interactive synthesis system in terms of *utility*, *easy to use*, *quality*, *efficiency*, and *satisfaction* respectively.

4.3.2 Outcome

Figure 4.12 offers quantitative measures of production time and stroke counts for each target task. The outcome statistics demonstrate that our interactive synthesis system can significantly reduce both the production time and the number of user strokes for either artists or novices. Please refer to Figures 4.13 and 4.14 for the sample user study outputs. Below we highlight the participants' authoring behaviors from our observation.



(a) *production time*

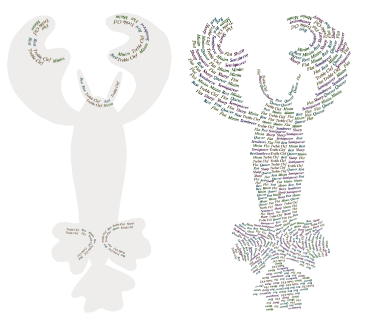


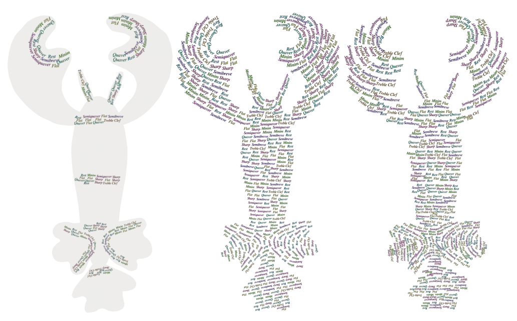
(b) *number of user strokes*

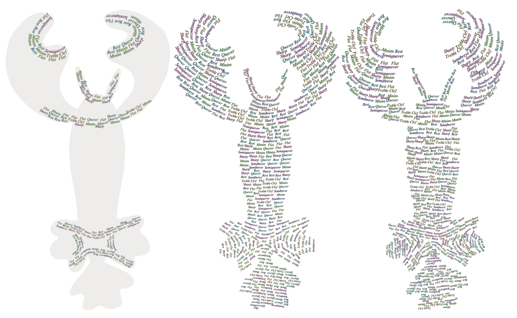
Figure 4.12: The outcome statistics. *The production time (a) and the total number of user strokes (b) for each target were measured individually.*

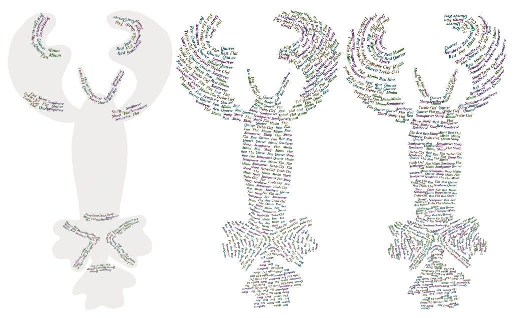
In terms of the 2D target, without the autocomplete mode, the participants had to repetitively adjust the brush radius and carefully place each brush stroke one by one so as to fill the entire output domain and properly match the element alignments in the 2D target. However, even though they could quickly and intuitively arrange the brush strokes all over the output domain via the palette-based brushing interface, analogous to common sketching and painting, to perfectly place these brush strokes side by side was not straightforward to them by reason that the brushing paths could be a little tilted to some extent. Thus, the outcomes regarding fully manual brushing can contain some obvious gaps amid the user strokes, and the orientations and scales of aggregate elements are not smooth enough either. While with the autocomplete mode, the participants could place a few strokes of aggregate elements around the singularity and some other evenly-spaced brush strokes within the remaining output domain for automatic completion. It can be seen that the results created via the one-step automatic optimization process can have smoother element distributions, orientations and scales as well as similar appearances to the 2D target even though the partially user-specified brush strokes are not totally identical. From our observation, some of the user strokes might follow the 2D target's outline but the outline did not actually affect the participants' authoring strategies in general, while the element alignments in the 2D target could affect the participants' decisions about where to place the brush strokes for automatic completion.

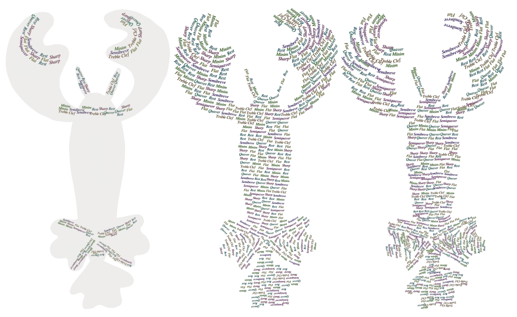
Similar results to the 3D target could be made via both without and with the autocomplete mode as the participants did not need to consider the element alignments for the body shape of the 3D target (i.e. lemons and bananas), but it can be noted that with the autocomplete mode, the outcomes can have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


(a) *user input* **(b)** *2D target*


(c) *user input* **(d)** *auto on* **(e)** *auto off*


(f) *user input* **(g)** *auto on* **(h)** *auto off*


(i) *user input* **(j)** *auto on* **(k)** *auto off*


(l) *user input* **(m)** *auto on* **(n)** *auto off*

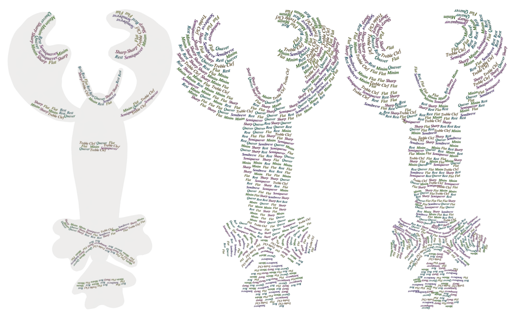
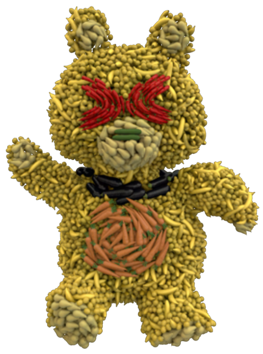

(o) *user input* **(p)** *auto on* **(q)** *auto off*

Figure 4.13: The sample user study outputs for the 2D target. *Each group of the sample user study outputs contains the partially specified input for automatic completion, the result automatically computed from the user input (i.e. auto on), and the output created by fully manual brushing (i.e. auto off).*


(a) *3D target*

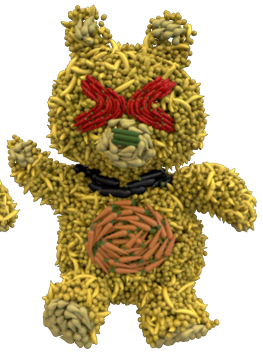
(b) *auto on*

(c) *auto off*

(d) *auto on*

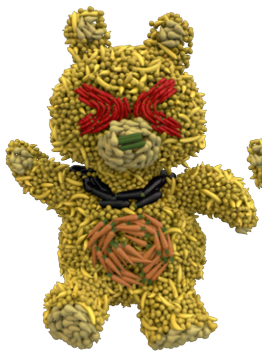
(e) *auto off*

(f) *auto on*


(g) *auto off*

(h) *auto on*

(i) *auto off*


(j) *auto on*

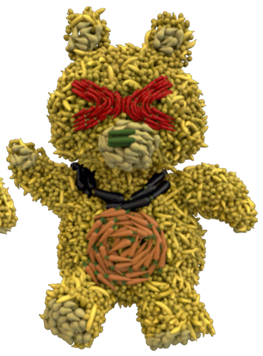

(k) *auto off*

Figure 4.14: The sample user study outputs for the 3D target. *Each group of the sample user study outputs contains the result computed via automatic completion plus partially manual brushing and the output created by fully manual brushing. With the autocomplete mode (i.e. (b), (d), (f), (h) and (j)), the lemons and bananas were automatically synthesized within the output domain via our interactive synthesis system. Without the autocomplete mode (i.e. (c), (e), (g), (i) and (k)), the lemons and bananas were interactively brushed by the participants over the output domain.*

While comparing the brushing behavior between the professional artists and the novice users, thanks to our intuitive and user-friendly brushing workflow, there was no obvious behavior difference between the participants based on our observation in terms of partially specifying a few brush strokes for automatic completion. About fully manual brushing, in the main, the novice users tended to brush the output domains with comparatively short brush strokes and sometimes some of their brushing paths might be a little slightly zigzag, whereas the professional artists could more flexibly utilize brush strokes with different lengths to generate the outputs. Note that the slightly zigzag user strokes did not lead to visual artifacts in their results since our proposed formulations can directly compute smooth element fields even from incoherent brush strokes as exemplified in Figure 4.9.

4.3.3 User Feedback

Figure 4.15 shows a summarization of the user feedback marked by each participant. Overall, the participants were content with our interactive synthesis system and commented that they can easily learn and understand the brushing workflow and directly produce desirable results without requiring a great amount of practice and expertise, as our palette-based brushing interface can fit the natural artist workflow. They also said that with the interactive synthesis system, they only demand to design a few specific brush strokes for automatic completion instead of arranging all the element collections across the output domain during the process of iterative design, and this concept is fairly creative and novel to them. When asked about comparing manual placement with automatic completion, the participants stated that the one-step automatic optimization process can indeed help them to reduce input workload and obtain smooth output compositions in an efficient and handy manner, and it can significantly encourage them to explore more interesting experiments. The participants were happy to see that our developed system can improve existing production pipelines for professional artists and benefit more novice users in the production of artistic works.

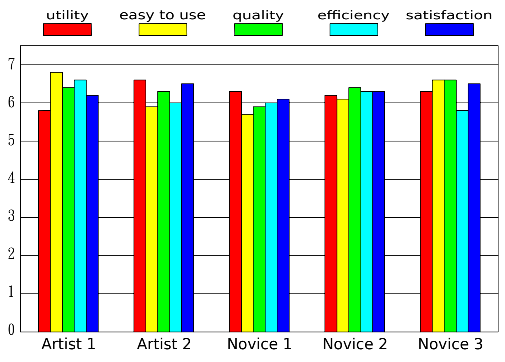


Figure 4.15: The user feedback. *The participants were asked to mark our interactive synthesis system in terms of utility, easy to use, quality, efficiency, and satisfaction, and all quantities are expressed in a 7-point Likert scale.*

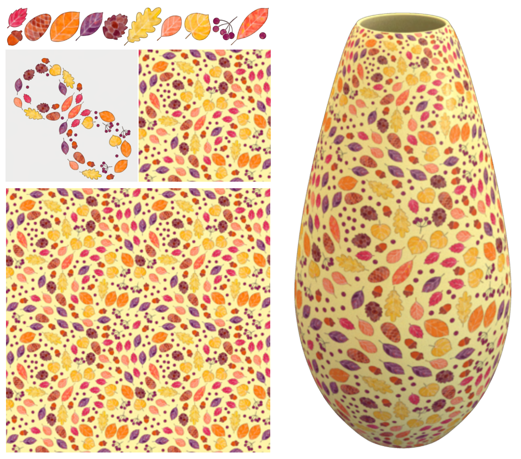
Our participants also made comments related to possible enhancement and conceivable usage for our interactive synthesis system. The artists recommended that it can be desirable to have more advanced brush controls such as a single stroke with varying brush radii for more varied effects, and being able to save the designed brush strokes as presets for further reuse (i.e. relocating a set of predefined brush strokes at a desired region) may be handier. Some participants expressed that after automatic completion, to let users slightly adjust a few local aggregate elements with specified orientations or scales as partial inputs for local optimization can be an alternative avenue for local field manipulation, as it can still work under our element field formulation, and this may benefit some scenarios which require more precise control of flow directions. Since our current prototype system is developed to provide a general user interface for interactive texturing across different output domains, we believe that their advice can be integrated into the developed system for tailored applications in accordance with requirements.

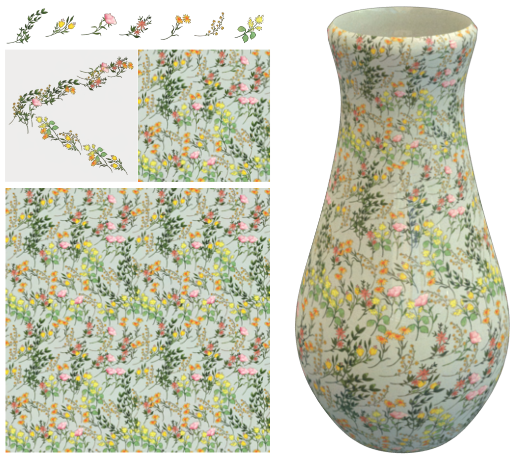
4.4 Other Applied Results

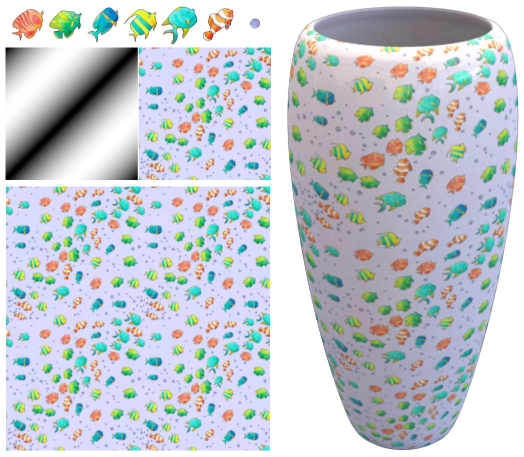
In addition to graphic design, artistic collage, and aggregate modeling we demonstrate in the previous sections, here we show more applications of our interactive synthesis system in pattern design, solid texturing, and field visualization as follows.

4.4.1 Pattern Design

There can be no doubt that element arrangements are important for pattern design (Loi *et al.* 2017), but it can still be difficult or inconvenient for ordinary users to effectively design tile-based patterns composed of aggregate elements with varying anisotropy and deformability. Since existing tools often require users to manually arrange individual aggregate elements for the generation of repeating patterns in a tedious and time-consuming process as exemplified in (Cunningham 2015; Purdy 2019), the users can notably take plenty of mental energy as well as production time to iteratively design such patterns with different styles (e.g. compact or loose output formations). As our interactive synthesis system can supply the capability of flexibly manipulating the distributions of aggregate elements, creators are enabled to directly form diverse output compositions with dense, sparse, spatially varying or evenly overlapping element distributions without requiring a great amount of manual labor and technical expertise. As a result, analogous to Figure 4.8, through toroidal boundary conditions, our developed system can significantly assist the creators in interactively designing a variety of tileable patterns with intended element distributions, orientations and scales via an efficient and handy manner as illustrated in Figure 4.16.


(a) *colorful pattern*


(b) *floral pattern*


(c) *fish pattern*

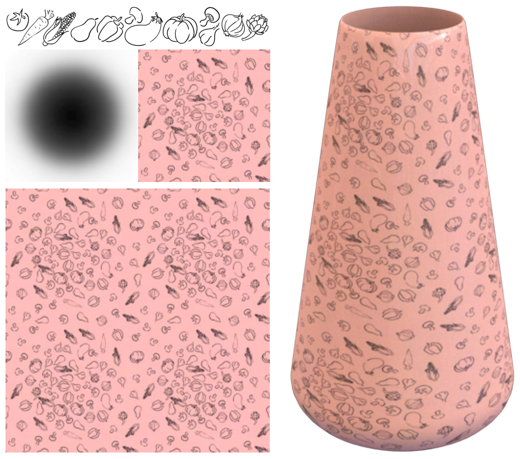

(d) *vegetable pattern*

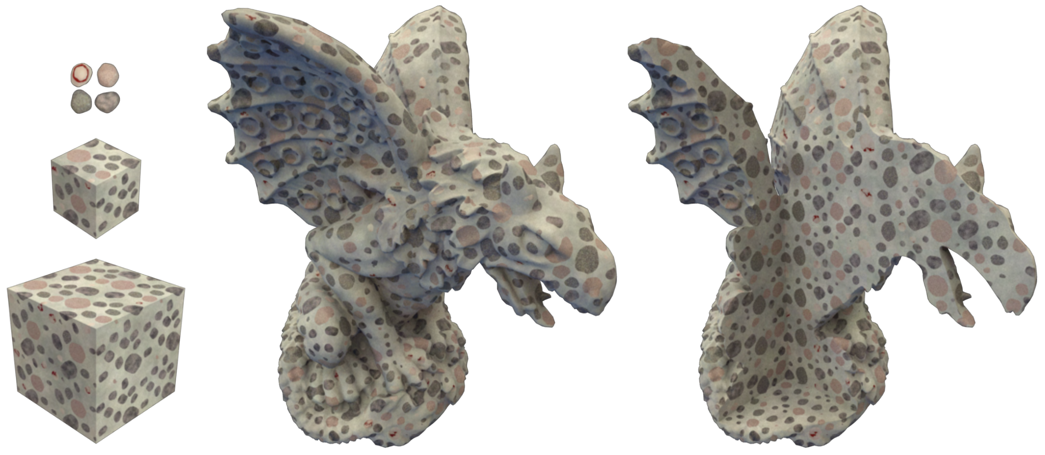
Figure 4.16: Tileable pattern design. *When given partially user-specified brush strokes, the tileable patterns with dense (a) or evenly overlapping (b) element distributions can be properly generated. Note that in (b), we overly increase the total number of aggregate elements placed into the output domain to form slightly overlapping but comparatively uniform element distributions. While taking advantage of control maps (i.e. (c) and (d)), the distributions of aggregate elements in the tileable patterns can be spatially varied. Note that in (c), we utilize the partially user-specified brush stroke in Figure 4.8c and a tileable density map as the given inputs for optimization, and in (d), we directly randomize the initial orientations of aggregate elements and also match the synthesized elements with an extra density map. The aggregate elements we use here are designed by Freepik (2019).*

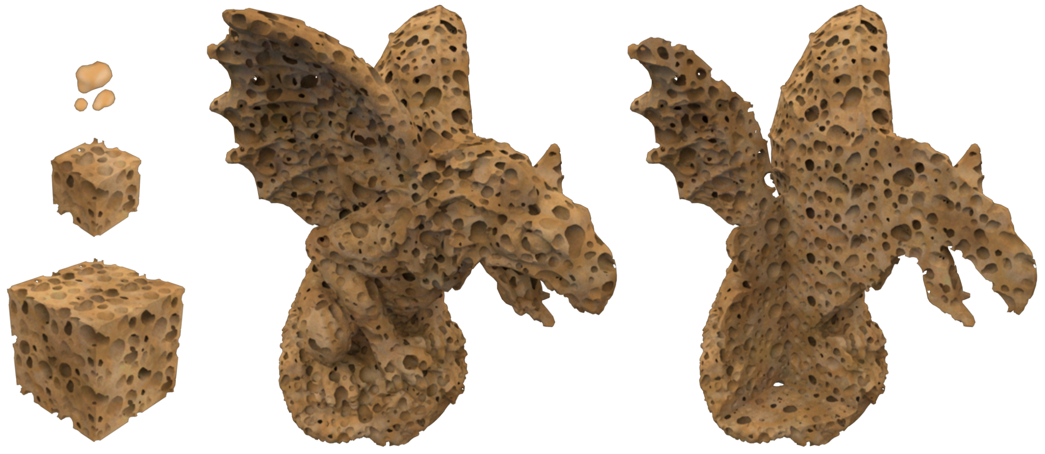
4.4.2 Solid Texturing

The creation of solid textures is another application of our interactive synthesis system. Since most proposed methods for solid texturing (e.g. (Jagnow *et al.* 2004; Qin and Yang 2007; Kopf *et al.* 2007; Du *et al.* 2013)) create the solid textures from 2D input exemplars such as photos, the input exemplars with specific structures might not be properly synthesized. To our best knowledge, there is also a lack of algorithms which can generate tile-based solid textures from user-specified aggregate elements. In contrast, our developed system can not only create 3D element tiles from specific aggregate elements but also well maintain the element configuration. Users can directly choose a few types of aggregate elements from the element palette and take relevant manipulations to create tileable solid textures, and then these tile-based solids can be subsequently reused to fully construct arbitrary objects as shown in Figure 4.17. An extra benefit is that instead of using a dense 3D array to store relevant texture information in traditional voxel-based algorithms, based on our devised element representation, we only need to record the source models of aggregate elements and corresponding information about the positions, orientations and scales of element samples of the synthesized elements to entirely reconstruct the target objects without compromising texture resolution and storage capability.

4.4.3 Field Visualization

To visualize orientation fields can be a direct application of our developed system. As illustrated in Figure 4.18, by properly distributing and aligning anisotropic elements to depict underlying fields, it can be considered as an alternative geometry-based technique for field visualization.


(a) *pebble cement*


(b) *beehive rock*

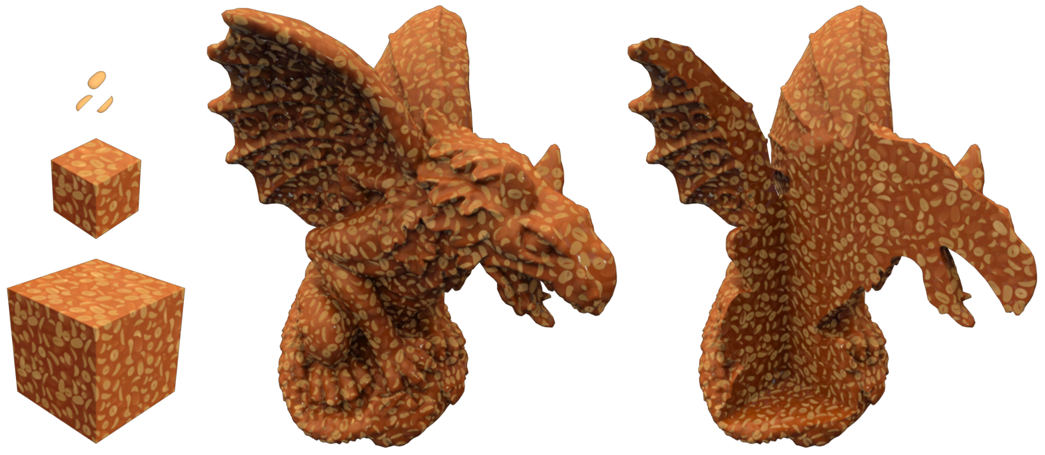
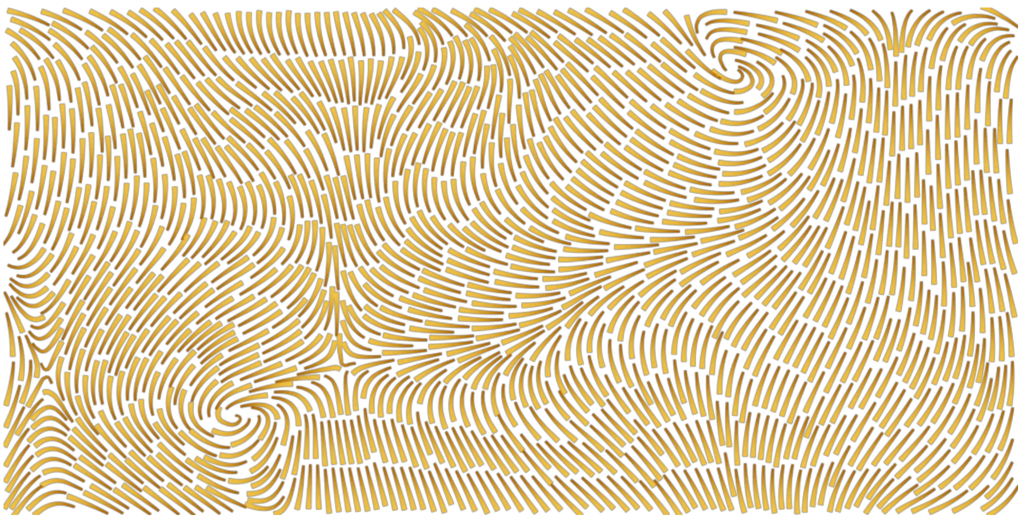
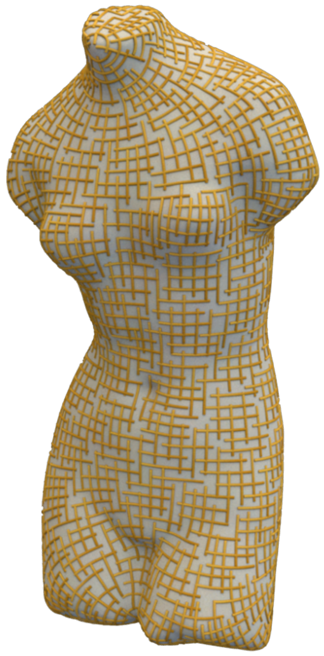

(c) *peanut caramel*

Figure 4.17: Solid texturing. *By randomly resizing and orienting a few aggregate elements from the element palette, tile-based solid textures can be created for concrete (a), eroded (b) and colloidal (c) structures. Note that in (b), the aggregate elements are used as cutting objects to trim models.*


(a) *2D plane*


(b) *3D surface*

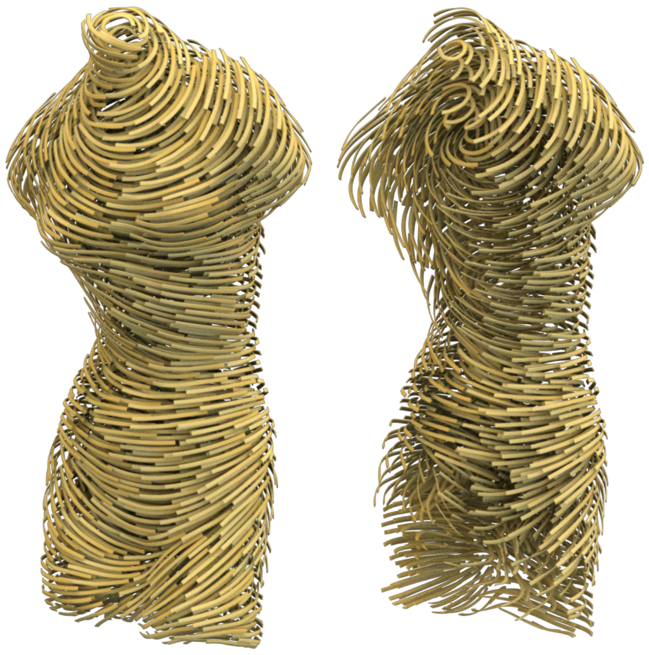

(c) *3D volume*

Figure 4.18: Field visualization. *To more clearly depict the underlying field, aggregate elements can be properly assigned color-ramped textures (a) for the enhancement of the visual effect of flow directions. By directly mapping elongated elements onto polygonal surfaces (b), cross fields can be visualized over the surfaces. Analogous to (Palacios* et al. *2017), anisotropic elements can also be exploited to individually indicate the orientations of field axes for the visualization of volumetric tensor fields (c).*

5 Conclusion and Future Work

5.1 Conclusion

It is obvious that the arrangement of aggregate elements is important and popular for artists and designers, as these well-organized element aggregations can be widely employed for a variety of practical applications, such as graphic design, artistic collage, and aggregate modeling. Nevertheless, a certain number of creators, especially ordinary novices, often find it difficult or ineffective to appropriately transform their conceptual ideas into concrete design outcomes through existing practices. Additionally, to well produce satisfactory element aggregations, those creators not only have to possess essential production abilities in advance but also demand heavy efforts in most cases by reason that there is still a lack of efficient algorithms and general user interfaces for interactive authoring of aggregate elements with varying anisotropy and deformablility. As a consequence, the desire for a well-developed element synthesis framework that can flexibly cope with diverse output compositions for pragmatic usage, especially in 3D environments, can be significant. In order to effectively overcome this challenging issue and reasonably satisfy the demand from users in the iterative design procedure, in this thesis, we thus specifically formulate the autocomplete element fields that can properly deal with a variety of phenomena and output formations across different output domains and further develop the interactive synthesis system to facilitate the creation of element aggregations with high production quality and efficiency.

In terms of the key contributions of our research, the element distribution formulation, which includes the data-driven method (Hsu *et al.* 2018) and the procedural approach (Hsu *et al.* 2020), focuses on adequately distributing aggregate elements with anisotropic shapes and various sizes to form a variety of novel mixtures. As compared with existing element synthesis practices, based on our devised element representation, the element distribution formulation can directly synthesize and mix distinct types of aggregate elements in high output standard without the need to integrate any additional physics solvers. In addition, the capability of being able to manipulate the distribution of aggregate elements can also demonstrate a clear advantage that our procedural approach can provide more functionality for element synthesis, since a majority of the proposed algorithms do not take user-specified element distributions into account owing to algorithmic limitations or expensive computation cost. This notable advantage can be crucial for the production of artistic works, as it can offer more freedom of designing diverse output compositions.

Moreover, the element field formulation, which is our second key contribution, concentrates on smoothly organizing the overall element arrangements on the basis of inter- and intra-element relationships such that complete element fields can be automatically computed according to given specifications (e.g. scalar or direction fields). Since existing field interpolation algorithms or field design systems are not developed for the purpose of orienting aggregate elements, in contrast to the two-step process, our one-step automatic optimization process can more adaptively orient all the aggregate elements to better reflect user intention and even effectively avoid undesirable artifacts, such as element misalignments near singular points. Based on our element field formulation, rigid and deformable elements can be straightforwardly distinguished without needing to incorporate any additional schemes (e.g. deformation simulation), and by directly adjusting the field continuity condition, the topology of element fields can also be accordingly manipulated without requiring any extra field preprocessing (e.g. field smoothing). Clearly, this remarkable flexibility can not only significantly streamline work procedures but also further enrich visual diversity of artistic works without breaking the production workflow.

In addition to the proposed formulations, another key contribution of the research is the development of the interactive synthesis system, as our developed system, centered on the concept of our autocomplete element fields, is the first for interactive authoring of aggregate elements with varying anisotropy and deformability for multiple applications. During the process of iterative design, through our palette-based brushing interface, creators can directly brush different output domains with chosen element collections to flexibly deal with diverse output compositions under an intuitive and user-friendly brushing workflow without requiring a great deal of manual labor and technical expertise. When with the automatic completion of full outcomes based on partially user-specified inputs, the interactive synthesis system can dramatically reduce input workload and simultaneously enhance output quality to properly assist both professional artists and novice users in the production of desired element aggregations without compromising their authoring freedom. Our current prototype system as an extension of traditional painting systems can indeed more naturally fit element synthesis with interactive authoring and sufficiently offer users more usability and interactivity than existing practices.

5.2 Future Work

As described in Section 3.1, distinct types of aggregate elements are characterized by a set of element samples and graphs in our devised element representation, and currently we set these element samples and graphs by hand. Even though the automatic computation of element representation can be fulfilled by utilizing a few relevant algorithms as mentioned in Figure 3.1, it may not ideally fit user design intention or reach operational standard for element synthesis, and thus further adjustments, which include moving, resizing or removing the related element samples and graphs, are needed in order to better approach the intended element representation. As a result, to simplify the procedure, a user-interactive editing tool for efficiently specifying the element samples and graphs can be considered as a potential future task. Furthermore, as our proposed formulations can effectively handle aggregate elements with general shapes, distributions and alignments, there can exist several interesting research directions worth investigating and we individually list them as follows:

1. **Geometric feature brushes:** While considering geometric features (e.g. fish scales or bird feathers) as aggregate elements, it should be possible to interactively brush such geometric features over polygonal surfaces, and this can be achieved by integrating (Schmidt *et al.* 2006; Takayama *et al.* 2011) into our interactive synthesis system to process the geometric features over the surfaces.

2. **3D printable element aggregations:** As inspired by Dumas *et al.* (2018), by properly adding extra constraints to make sure that all the synthesized elements can slightly contact with each other, we believe that our element synthesis framework can be directly extended for 3D printing.

3. **VR brushing:** Recently, VR brushing has received significant attention, as it can provide an immersive environment for interactive painting and modeling like (Kim *et al.* 2018; Xing *et al.* 2019). We plan to implement our proposed formulations under VR and offer a user interface for interactive authoring of aggregate elements in immersive environments.

4. **Continuous element synthesis:** Since our current prototype system does not take continuous elements (Roveri *et al.* 2015) into consideration due to the complexity of brush controls in 3D environments and the difficulty of arbitrarily mixing such continuous elements, a potential research direction is to extend our proposed formulations with (Lu *et al.* 2014; Zhou *et al.* 2014) for interactive synthesis of continuous artistic patterns following user specifications.

5. **Dynamic element synthesis:** As demonstrated in Figure 3.16, our proposed formulations can also handle dynamic input fields. Therefore, we also plan to extend this feature for motion graphics like (Kazi *et al.* 2014; Xing *et al.* 2016) and element animations like (Ma *et al.* 2013; Milliez *et al.* 2014, 2016) as our future work. While to properly visualize unsteady input fields is still an open problem, especially in 3D output domains, we are going to more fully explore possible solutions to efficiently deal with time-varying input fields and moving elements for dynamic field visualization like (Jobard *et al.* 2012; Hu *et al.* 2019) as well.

Finally, to enhance our interactive synthesis system, we would like to overcome the limitation issues described in Section 4.2.3 and carry out more desirable user controls mentioned in Section 4.3.3 for more specific usage. Since the developed system is directly fulfilled into an external Autodesk Maya plugin, we are planning to share the plugin to relevant public communities so as to obtain useful and constructive feedback from potential users for the future improvement.

Bibliography

AlMeraj, Z., Kaplan, C. S. and Asente, P., 2013. Patch-based geometric texture synthesis. In *Proceedings of the Symposium on Computational Aesthetics*, CAE '13, 15–19.

Amenta, N., Choi, S. and Kolluri, R. K., 2001. The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, SMA '01, 249–266.

Aurenhammer, F., 1987. Power diagrams: Properties, algorithms and applications. *SIAM J. Comput.*, 16(1), 78–96.

Bendsoe, M. and Sigmund, O., 2004. *Topology Optimization: Theory, Methods and Applications*.

Bridson, R., 2007. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07.

Buron, C., Marvie, J.-E., Guennebaud, G. and Granier, X., 2015. Dynamic on-mesh procedural generation. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, 17–24.

Chen, G., Kwatra, V., Wei, L.-Y., Hansen, C. D. and Zhang, E., 2012. Design of 2d time-varying vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(10), 1717–1730.

Chen, W., Ma, Y., Lefebvre, S., Xin, S., Martínez, J. and wang, w., 2017. Fabricable tile decors. *ACM Trans. Graph.*, 36(6), 175:1–175:15.

Chen, Y., Fu, H. and Au, K. C., 2016. A multi-level sketch-based interface for decorative pattern exploration. In *SIGGRAPH ASIA 2016 Technical Briefs*, SA '16.

Cho, J. H., Xenakis, A., Gronsky, S. and Shah, A., 2007. Anyone can cook – inside ratatouille's kitchen. In *SIGGRAPH 2007 Courses*.

Cohen-Steiner, D., Alliez, P. and Desbrun, M., 2004. Variational shape approximation. *ACM Trans. Graph.*, 23(3), 905–914.

Comaniciu, D. and Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5), 603–619.

Coumans, E., 2013. Bullet physics library. Available from: `https://bulletphysics.org` [Accessed 15 March 2020].

Cunningham, T., 2015. How to create seamless patterns in illustrator. Available from: `https://www.youtube.com/watch?v=ITRZ75OKrG0` [Accessed 15 March 2020].

Dalal, K., Klein, A. W., Liu, Y. and Smith, K., 2006. A spectral approach to npr packing. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '06, 71–78.

Davison, T., Samavati, F. and Jacob, C., 2019. Interactive example-palettes for discrete element texture synthesis. *Computers & Graphics*, 78, 23 – 36.

Doyle, L., Anderson, F., Choy, E. and Mould, D., 2019. Automated pebble mosaic stylization of images. *Computational Visual Media*, 5, 33–44.

Du, S.-P., Hu, S.-M. and Martin, R. R., 2013. Semiregular solid texturing from 2d image exemplars. *IEEE Transactions on Visualization and Computer Graphics*, 19(3), 460–469.

Dumas, J., Martínez, J., Lefebvre, S. and Wei, L.-Y., 2018. Printable aggregate elements. *CoRR*, abs/1811.02626.

Emilien, A., Vimont, U., Cani, M.-P., Poulin, P. and Benes, B., 2015. Worldbrush: Interactive example-based synthesis of procedural virtual worlds. *ACM Trans. Graph.*, 34(4), 106:1–106:11.

Freepik, , 2019. Graphic resources for everyone. Available from: `https://www.freepik.com/` [Accessed 15 March 2020].

Frehse, L., 2018. Automodeller pro. Available from: `http://www.automodeller.com/` [Accessed 15 March 2020].

Fu, H., Wei, Y., Tai, C.-L. and Quan, L., 2007. Sketching hairstyles. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, SBIM '07, 31–36.

Gal, R., Sorkine, O., Popa, T., Sheffer, A. and Cohen-Or, D., 2007. 3d collage: Expressive non-realistic modeling. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, 7–14.

Gay, C., 2016a. Physx painter. Available from: `http://www.scriptspot.com/3ds-max/scripts/physx-painter` [Accessed 15 March 2020].

Gay, C., 2016b. Physx painter teaser. Available from: `https://vimeo.com/162046605` [Accessed 15 March 2020].

Guérin, E., Galin, E., Grosbellet, F., Peytavie, A. and Génevaux, J.-D., 2016. Efficient modeling of entangled details for natural scenes. *Computer Graphics Forum*, 35(7), 257–267.

Hausner, A., 2001. Simulating decorative mosaics. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 573–580.

Hertzmann, A. and Zorin, D., 2000. Illustrating smooth surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, 517–526.

Hsu, C.-Y., Wei, L.-Y., You, L. and Zhang, J. J., 2018. Brushing element fields. In *SIGGRAPH Asia 2018 Technical Briefs*, SA '18.

Hsu, C.-Y., Wei, L.-Y., You, L. and Zhang, J. J., 2020. Autocomplete element fields. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 1–13.

Hsu, S.-W. and Keyser, J., 2010. Piles of objects. *ACM Trans. Graph.*, 29 (6), 155:1–155:6.

Hsu, S.-W. and Keyser, J., 2012. Automated constraint placement to maintain pile shape. *ACM Trans. Graph.*, 31(6), 150:1–150:6.

Hu, W., Chen, Z., Pan, H., Yu, Y., Grinspun, E. and Wang, W., 2016. Surface mosaic synthesis with irregular tiles. *IEEE Transactions on Visualization and Computer Graphics*, 22(3), 1302–1313.

Hu, Z., Xie, H., Fukusato, T., Sato, T. and Igarashi, T., 2019. Sketch2vf: Sketch-based flow design with conditional generative adversarial network. *Computer Animation and Virtual Worlds*, 30(3-4), e1889.

Huang, H., Zhang, L. and Zhang, H.-C., 2011. Arcimboldo-like collage using internet images. *ACM Trans. Graph.*, 30(6), 155:1–155:8.

Huang, Z., Wang, J., Fu, H. and Lau, R. W. H., 2014. Structured mechanical collage. *IEEE Transactions on Visualization and Computer Graphics*, 20 (7), 1076–1082.

Huang, Z. and Ju, T., 2016. Extrinsically smooth direction fields. *Computers & Graphics*, 58, 109–117.

Ijiri, T., Mêch, R., Igarashi, T. and Miller, G., 2008. An example-based procedural system for element arrangement. *Computer Graphics Forum*, 27 (2), 429–436.

Interrante, V., 1997. Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, 109–116.

Jagnow, R., Dorsey, J. and Rushmeier, H., 2004. Stereological techniques for solid textures. *ACM Trans. Graph.*, 23(3), 329–335.

Jobard, B. and Lefer, W., 1997. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing '97*, 43–56.

Jobard, B., Ray, N. and Sokolov, D., 2012. Visualizing 2d flows with animated arrow plots. *arXiv preprint arXiv:1205.5204*.

Kalogerakis, E., Nowrouzezahrai, D., Breslav, S. and Hertzmann, A., 2012. Learning hatching for pen-and-ink illustration of surfaces. *ACM Trans. Graph.*, 31(1), 1:1–1:17.

Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S. and Fitzmaurice, G., 2014. Draco: Bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, 351–360.

Kazi, R. H., Igarashi, T., Zhao, S. and Davis, R., 2012. Vignette: Interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, 1727–1736.

Kim, J. and Pellacini, F., 2002. Jigsaw image mosaics. *ACM Trans. Graph.*, 21(3), 657–664.

Kim, Y., Kim, B. and Kim, Y. J., 2018. Dynamic deep octree for high-resolution volumetric painting in virtual reality. *Computer Graphics Forum*, 37(7), 179–190.

Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D. and Wong, T.-T., 2007. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph.*, 26(3).

Kuhn, H. W., 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.

Kwan, K. C., Sinn, L. T., Han, C., Wong, T.-T. and Fu, C.-W., 2016. Pyramid of arclength descriptor for generating collage of shapes. *ACM Trans. Graph.*, 35(6), 229:1–229:12.

Lagae, A. and Dutré, Ph., 2006. Poisson sphere distributions. In *Vision, Modeling, and Visualization*, 373–379.

Landes, P.-E., Galerne, B. and Hurtut, T., 2013. A shape-aware model for discrete texture synthesis. In *Proceedings of the Eurographics Symposium on Rendering*, EGSR '13, Aire-la-Ville, Switzerland, Switzerland. 67–76.

Levis, M., 2014. Tutorial on how to do a typography design of a woman's face in photoshop cc. Available from: `https://www.youtube.com/watch?v=LcZFp1s1AQI` [Accessed 15 March 2020].

Li, P., Wang, B., Sun, F., Guo, X., Zhang, C. and Wang, W., 2015. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Trans. Graph.*, 35(1), 8:1–8:16.

Li, Y., Bao, F., Zhang, E., Kobayashi, Y. and Wonka, P., 2011. Geometry synthesis on surfaces using field-guided shape grammars. *IEEE Transactions on Visualization and Computer Graphics*, 17(2), 231–243.

Loi, H., Hurtut, T., Vergne, R. and Thollot, J., 2013. Discrete texture design using a programmable approach. In *SIGGRAPH '13 Talks*, 43:1–43:1.

Loi, H., Hurtut, T., Vergne, R. and Thollot, J., 2017. Programmable 2d arrangements for element texture design. *ACM Trans. Graph.*, 36(3), 27:1–27:17.

Lu, J., Barnes, C., DiVerdi, S. and Finkelstein, A., 2013. Realbrush: Painting with examples of physical media. *ACM Trans. Graph.*, 32(4), 117:1–117:12.

Lu, J., Barnes, C., Wan, C., Asente, P., Mech, R. and Finkelstein, A., 2014. Decobrush: Drawing structured decorative patterns by example. *ACM Trans. Graph.*, 33(4), 90:1–90:9.

Lu, J., Yu, F., Finkelstein, A. and DiVerdi, S., 2012. Helpinghand: Example-based stroke stylization. *ACM Trans. Graph.*, 31(4), 46:1–46:10.

Lukáč, M., Fišer, J., Asente, P., Lu, J., Shechtman, E. and Sýkora, D., 2015. Brushables: Example-based edge-aware directional texture painting. *Comput. Graph. Forum*, 34(7), 257–267.

Lukáč, M., Fišer, J., Bazin, J.-C., Jamriška, O., Sorkine-Hornung, A. and Sýkora, D., 2013. Painting by feature: Texture boundaries for example-based image creation. *ACM Trans. Graph.*, 32(4), 116:1–116:8.

Ma, C., Wei, L.-Y., Lefebvre, S. and Tong, X., 2013. Dynamic element textures. *ACM Trans. Graph.*, 32(4), 90:1–90:10.

Ma, C., Wei, L.-Y. and Tong, X., 2011. Discrete element textures. *ACM Trans. Graph.*, 30(4), 62:1–62:10.

Madsen, K., Nielsen, H. B. and Tingleff, O., 2004. Methods for non-linear least squares problems (2nd ed.). Available from: `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3215` [Accessed 15 March 2020].

Maharik, R., Bessmeltsev, M., Sheffer, A., Shamir, A. and Carr, N., 2011. Digital micrography. *ACM Trans. Graph.*, 30(4), 100:1–100:12.

Meng, J., Papas, M., Habel, R., Dachsbacher, C., Marschner, S., Gross, M. and Jarosz, W., 2015. Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph.*, 34(4), 49:1–49:13.

Milliez, A., Guay, M., Cani, M.-P., Gross, M. and Sumner, R. W., 2016. Programmable animation texturing using motion stamps. *Comput. Graph. Forum*, 35(7), 67–75.

Milliez, A., Noris, G., Baran, I., Coros, S., Cani, M.-P., Nitti, M., Marra, A., Gross, M. and Sumner, R. W., 2014. Hierarchical motion brushes for animation instancing. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, NPAR '14, 71–79.

Muller, T., Papas, M., Gross, M., Jarosz, W. and Novak, J., 2016. Efficient rendering of heterogeneous polydisperse granular media. *ACM Trans. Graph.*, 35(6).

Palacios, J., Roy, L., Kumar, P., Hsu, C.-Y., Chen, W., Ma, C., Wei, L.-Y. and Zhang, E., 2017. Tensor field design in volumes. *ACM Trans. Graph.*, 36 (6).

Palacios, J. and Zhang, E., 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3).

Peng, M., Xing, J. and Wei, L.-Y., 2018. Autocomplete 3d sculpting. *ACM Trans. Graph.*, 37(4), 132:1–132:15.

Peytavie, A., Galin, E., Grosjean, J. and Mérillou, S., 2009. Procedural generation of rock piles using aperiodic tiling. *Computer Graphics Forum*, 28(7), 1801–1809.

Prasad, L., 1997. Morphological analysis of shapes. *CNLS Newsletter*, 139, 1–18.

Purdy, C., 2019. Make it, sell it: Repeating patterns in adobe illustrator. Available from: `https://create.adobe.com/2019/4/2/make_it_sell_it_repe.html` [Accessed 15 March 2020].

Qin, X. and Yang, Y.-H., 2007. Aura 3d textures. *IEEE Transactions on Visualization and Computer Graphics*, 13(2), 379–389.

Ramshaw, L. and Tarjan, R. E., 2012. On minimum-cost assignments in unbalanced bipartite graphs.

Reinert, B., Ritschel, T. and Seidel, H.-P., 2013. Interactive by-example design of artistic packing layouts. *ACM Trans. Graph.*, 32(6), 218:1–218:7.

Ritter, L., Li, W., Curless, B., Agrawala, M. and Salesin, D., 2006. Painting with texture. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, 371–376.

Roveri, R., Öztireli, A. C., Martin, S., Solenthaler, B. and Gross, M., 2015. Example based repetitive structure synthesis. *Comput. Graph. Forum*, 34 (5), 39–52.

Rycroft, C., 2009. Voro++: A three-dimensional voronoi cell library in c++. Available from: `http://math.lbl.gov/voro++/` [Accessed 15 March 2020].

Sakurai, K. and Miyata, K., 2014. Modelling of non-periodic aggregates having a pile structure. *Comput. Graph. Forum*, 33(1), 190–198.

Santoni, C. and Pellacini, F., 2016. gtangle: A grammar for the procedural generation of tangle patterns. *ACM Trans. Graph.*, 35(6), 182:1–182:11.

Saputra, R., Kaplan, C. and Asente, P., 2018. Repulsionpak: Deformation-driven element packing with repulsion forces. In *Proceedings of Graphics Interface 2018*, GI 2018, 10 – 17.

Saputra, R. A., Kaplan, C. S., Asente, P. and Měch, R., 2017. Flowpak: Flow-based ornamental element packing. In *Proceedings of the 43rd Graphics Interface Conference*, GI '17, 8–15.

Schiftner, A., Höbinger, M., Wallner, J. and Pottmann, H., 2009. Packing circles and spheres on surfaces. *ACM Trans. Graph.*, 28(5), 139:1–139:8.

Schmidt, R., Grimm, C. and Wyvill, B., 2006. Interactive decal compositing with discrete exponential maps. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, 605–613.

Schroeder, D., Coffey, D. and Keefe, D., 2010. Drawing with the flow: A sketch-based interface for illustrative visualization of 2d vector fields. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, 49–56.

Schwarz, M., Isenberg, T., Mason, K. and Carpendale, S., 2007. Modeling with rendering primitives: An interactive non-photorealistic canvas. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, 15–22.

Shugrina, M., Lu, J. and Diverdi, S., 2017. Playful palette: An interactive parametric color mixer for artists. *ACM Trans. Graph.*, 36(4), 61:1–61:10.

Sumner, R. W., Schmid, J. and Pauly, M., 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3).

Takayama, K., Okabe, M., Ijiri, T. and Igarashi, T., 2008. Lapped solid textures: Filling a model with anisotropic textures. *ACM Trans. Graph.*, 27 (3), 53:1–53:9.

Takayama, K., Schmidt, R., Singh, K., Igarashi, T., Boubekeur, T. and Sorkine, O., 2011. Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum*, 30(2), 613–622.

Thiery, J.-M., Guy, E. and Boubekeur, T., 2013. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Trans. Graph.*, 32(6), 178:1–178:12.

VideoLot, , 2016. Adobe illustrator cs6 — typography portrait — bruno mars. Available from: `https://www.youtube.com/watch?v=_kdhB-8tNeM` [Accessed 15 March 2020].

Wang, R., Zhou, K., Snyder, J., Liu, X., Bao, H., Peng, Q. and Guo, B., 2006. Variational sphere set approximation for solid objects. *Vis. Comput.*, 22 (9), 612–621.

Wei, L.-Y. and Levoy, M., 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 355–360.

Xing, J., Chen, H.-T. and Wei, L.-Y., 2014. Autocomplete painting repetitions. *ACM Trans. Graph.*, 33(6), 172:1–172:11.

Xing, J., Kazi, R. H., Grossman, T., Wei, L.-Y., Stam, J. and Fitzmaurice, G., 2016. Energy-brushes: Interactive tools for illustrating stylized elemental dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, 755–766.

Xing, J., Nagano, K., Chen, W., Xu, H., Wei, L.-Y., Zhao, Y., Lu, J., Kim, B. and Li, H., 2019. Hairbrush for immersive data-driven hair modeling. In *UIST 2019*.

Xing, J., Wei, L.-Y., Shiratori, T. and Yatani, K., 2015. Autocomplete hand-drawn animations. *ACM Trans. Graph.*, 34(6), 169:1–169:11.

Xu, J. and Kaplan, C. S., 2007. Calligraphic packing. In *Proceedings of Graphics Interface 2007*, GI '07, 43–50.

Xu, X., Zhang, L. and Wong, T.-T., 2010. Structure-based ascii art. *ACM Trans. Graph.*, 29(4), 52:1–52:10.

Zehnder, J., Coros, S. and Thomaszewski, B., 2016. Designing structurally-sound ornamental curve networks. *ACM Trans. Graph.*, 35(4), 99:1–99:10.

Zhang, E., Hays, J. and Turk, G., 2007. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(1), 94–107.

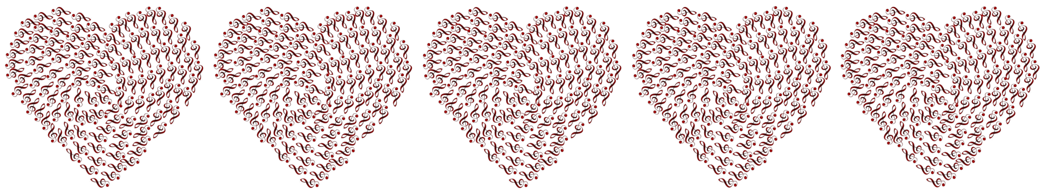
Zhang, E., Mischaikow, K. and Turk, G., 2006. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4), 1294–1326.

Zhang, G.-X., Du, S.-P., Lai, Y.-K., Ni, T. and Hu, S.-M., 2011. Sketch guided solid texturing. *Graphical Models*, 73(3), 59–73.

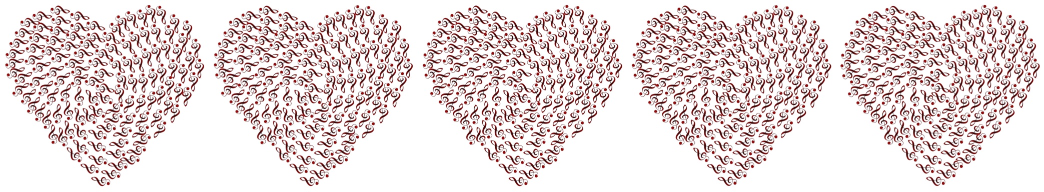
Zhou, S., Jiang, C. and Lefebvre, S., 2014. Topology-constrained synthesis of vector patterns. *ACM Trans. Graph.*, 33(6), 215:1–215:11.

Zou, C., Cao, J., Ranaweera, W., Alhashim, I., Tan, P., Sheffer, A. and Zhang, H., 2016. Legible compact calligrams. *ACM Trans. Graph.*, 35 (4), 122:1–122:12.

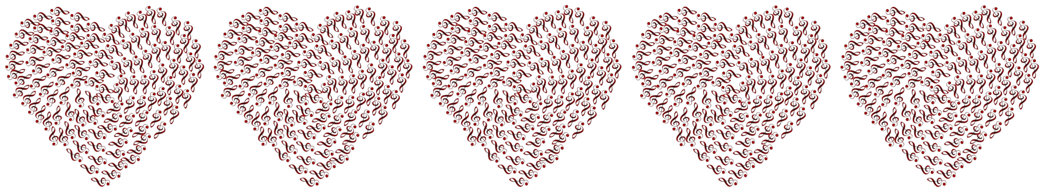
Appendix A



(a) *iteration 72* **(b)** *iteration 74* **(c)** *iteration 76* **(d)** *iteration 78* **(e)** *iteration 80*

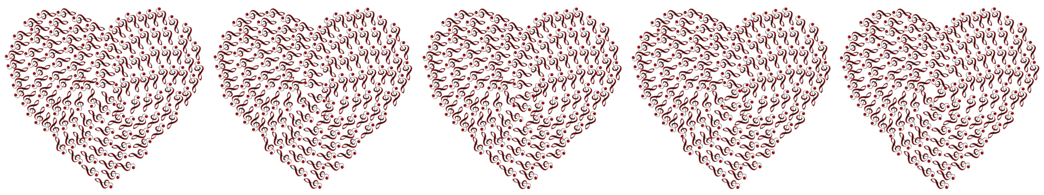


(f) *iteration 82* **(g)** *iteration 84* **(h)** *iteration 86* **(i)** *iteration 88* **(j)** *iteration 90*

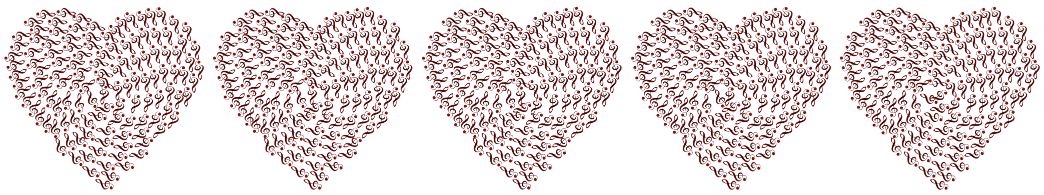


(k) *iteration 92* **(l)** *iteration 94* **(m)** *iteration 96* **(n)** *iteration 98* **(o)** *iteration 100*

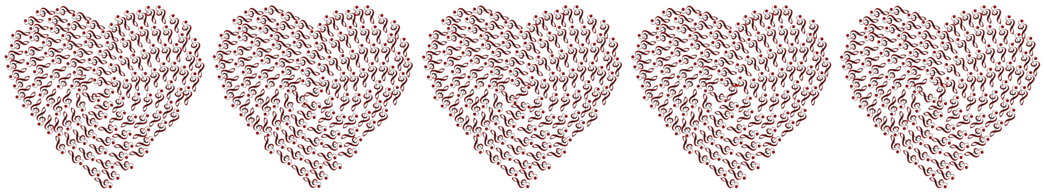
Figure A.1: The iteration process with the weight $\mathrm{w}_o(s', s, i)$.



(a) *iteration 72* **(b)** *iteration 74* **(c)** *iteration 76* **(d)** *iteration 78* **(e)** *iteration 80*



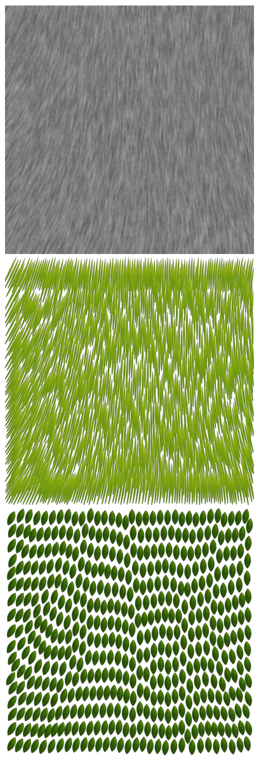
(f) *iteration 82* **(g)** *iteration 84* **(h)** *iteration 86* **(i)** *iteration 88* **(j)** *iteration 90*

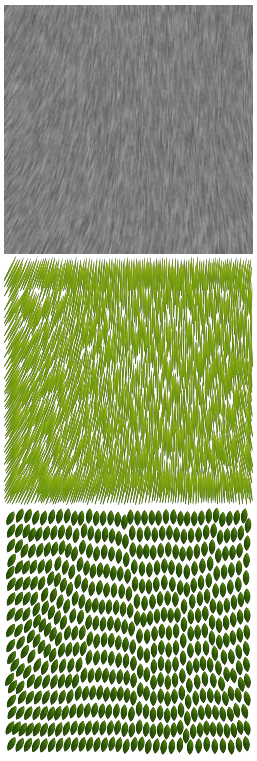


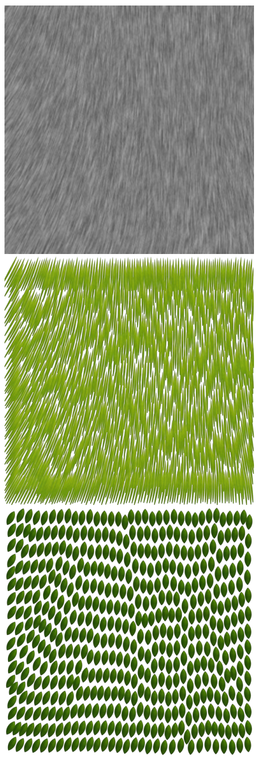
(k) *iteration 92* **(l)** *iteration 94* **(m)** *iteration 96* **(n)** *iteration 98* **(o)** *iteration 100*

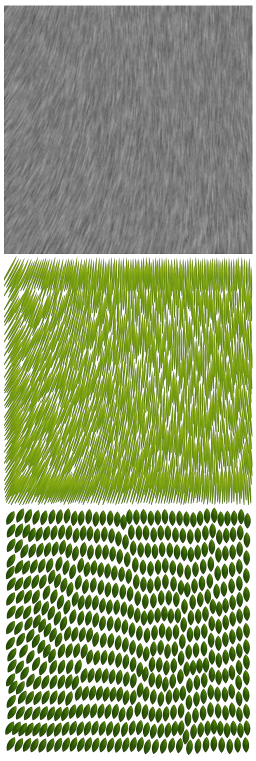
Figure A.2: The iteration process without the weight $\mathrm{w}_o(s', s, i)$.

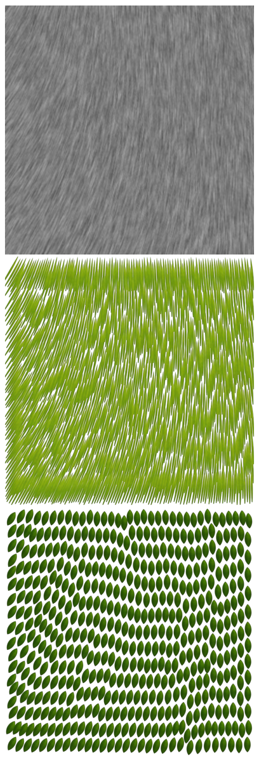
Appendix B

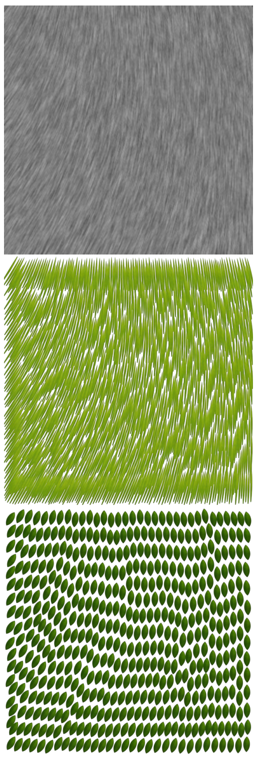

(a) *frame 1*

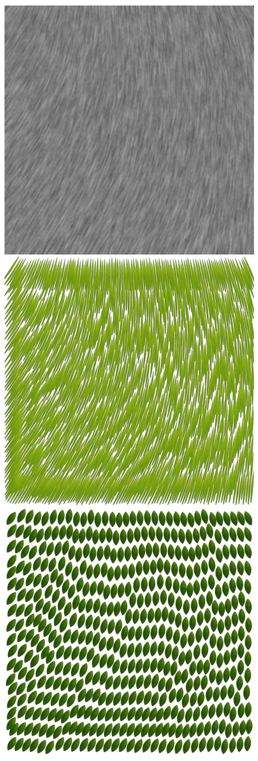

(b) *frame 2*


(c) *frame 3*


(d) *frame 4*


(e) *frame 5*


(f) *frame 6*


(g) *frame 7*

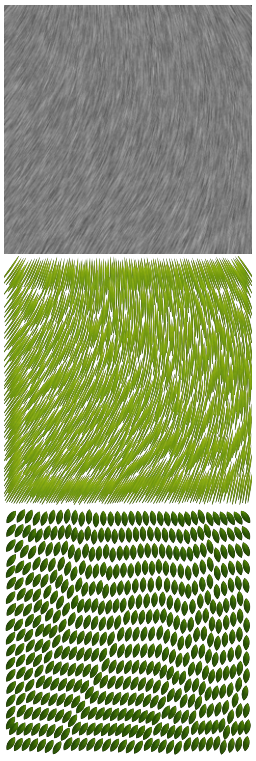
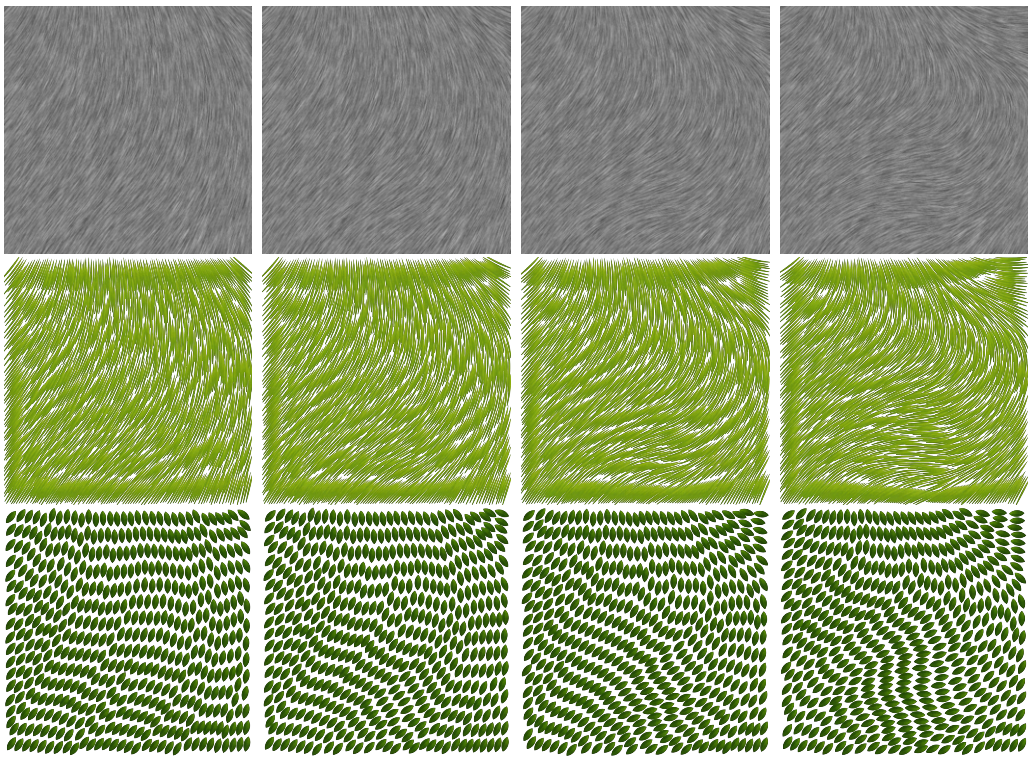

(h) *frame 8*

Figure B.1: Dynamic input fields from 1 to 8 frames.


(a) *frame 9* **(b)** *frame 10* **(c)** *frame 11* **(d)** *frame 12*

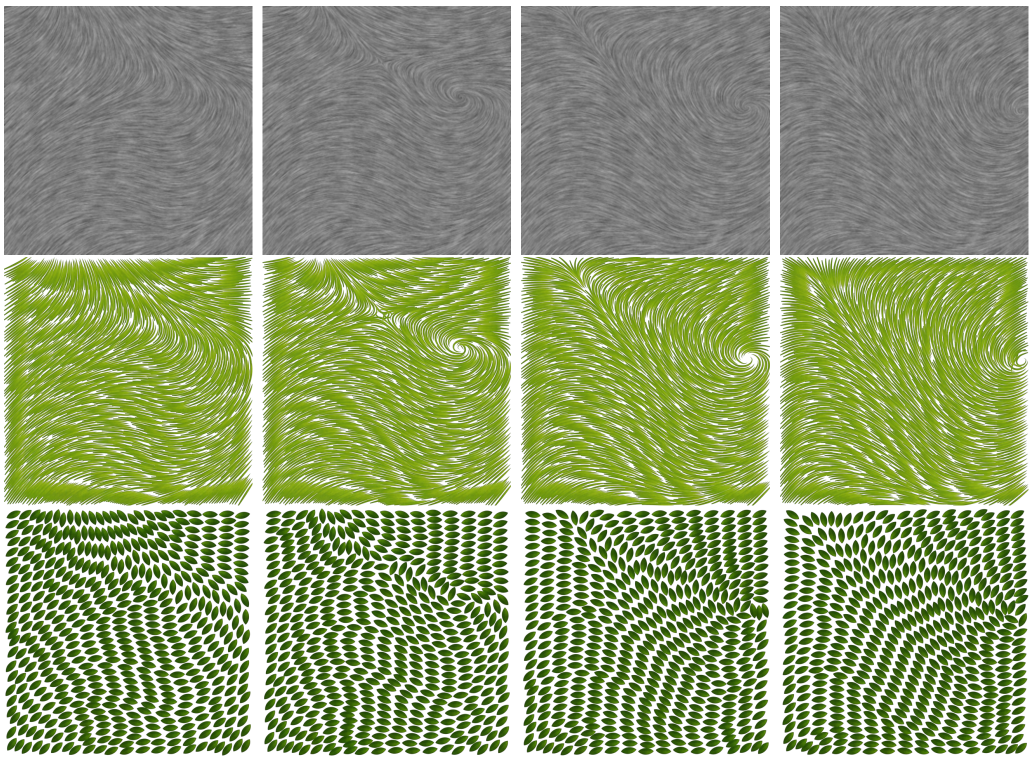
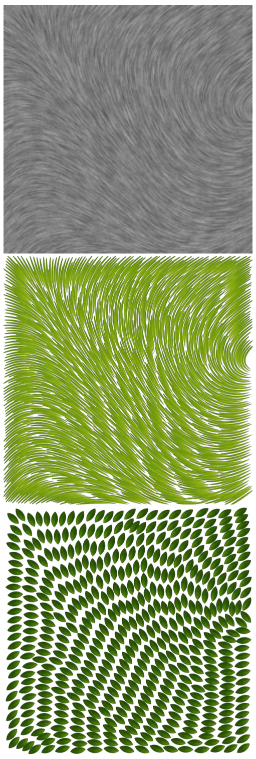
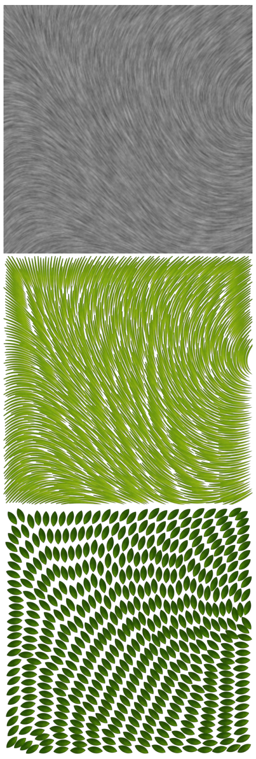
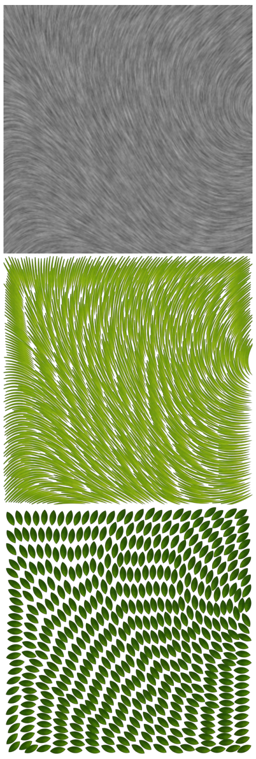
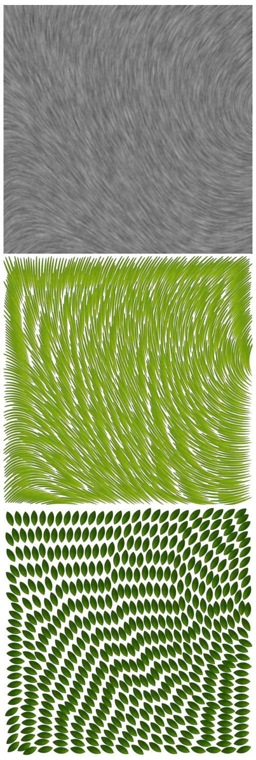

(e) *frame 13* **(f)** *frame 14* **(g)** *frame 15* **(h)** *frame 16*

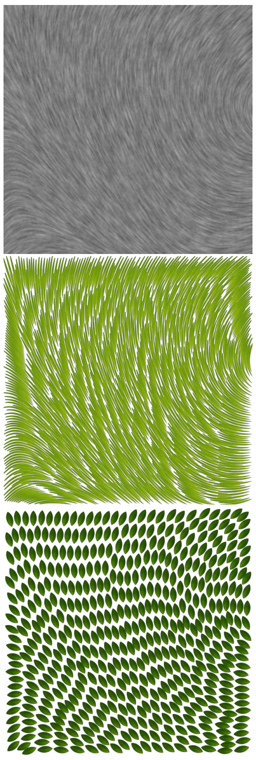
Figure B.2: Dynamic input fields from 9 to 16 frames.

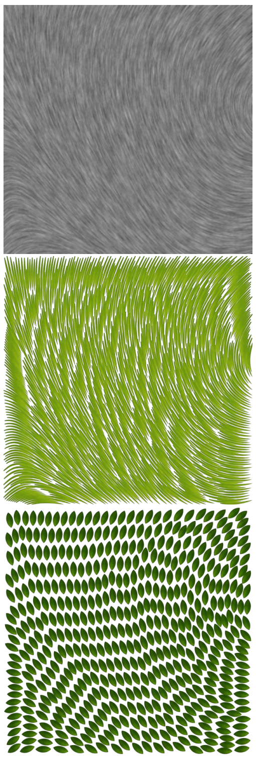

(a) *frame 17*

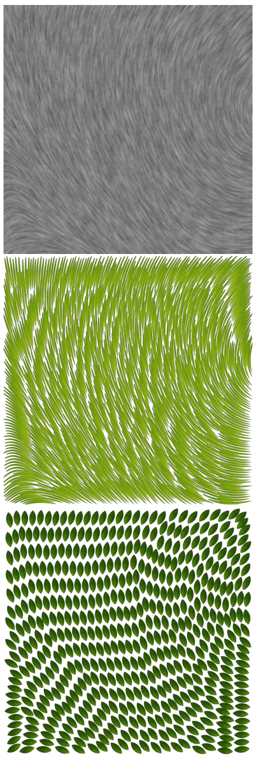

(b) *frame 18*


(c) *frame 19*


(d) *frame 20*


(e) *frame 21*


(f) *frame 22*


(g) *frame 23*

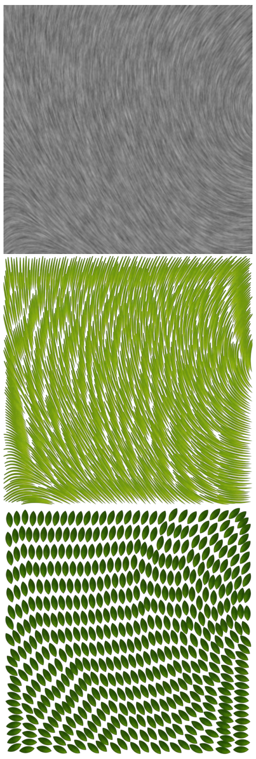
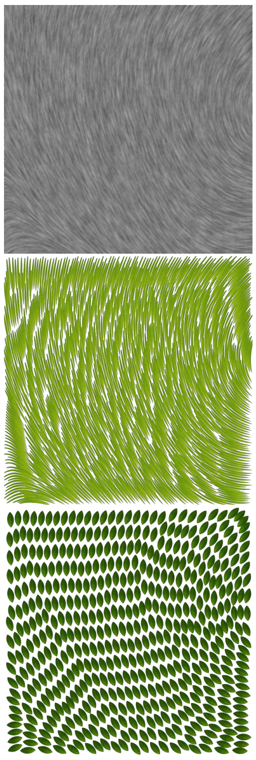
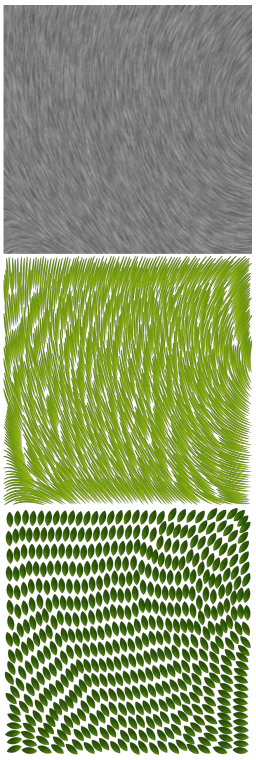
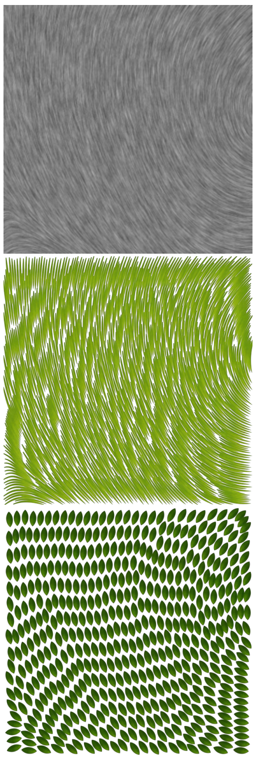
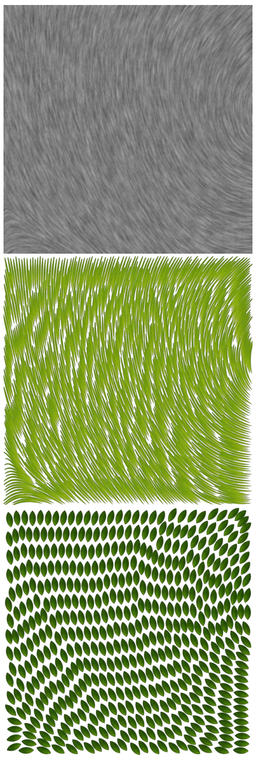

(h) *frame 24*

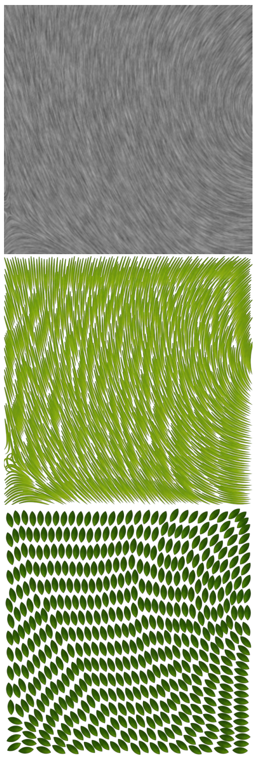
Figure B.3: Dynamic input fields from 17 to 24 frames.

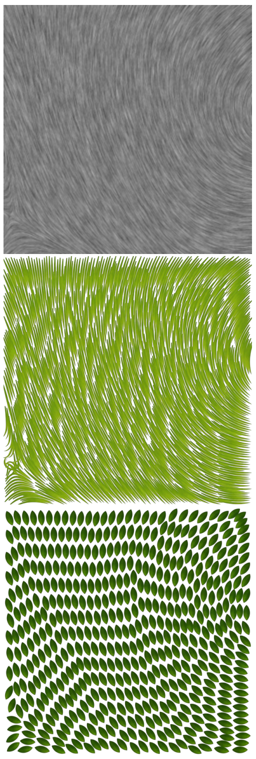

(a) *frame 25*

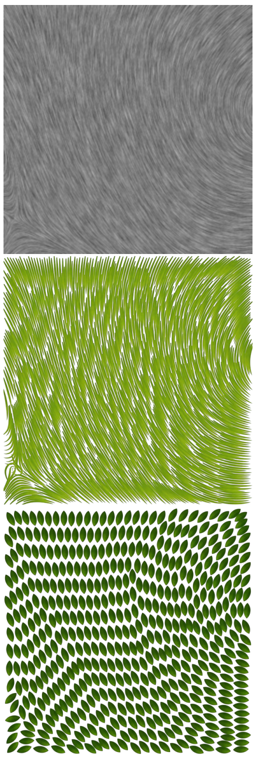

(b) *frame 26*


(c) *frame 27*


(d) *frame 28*


(e) *frame 29*


(f) *frame 30*


(g) *frame 31*

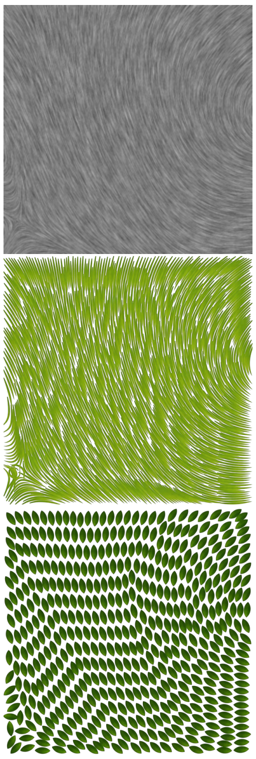
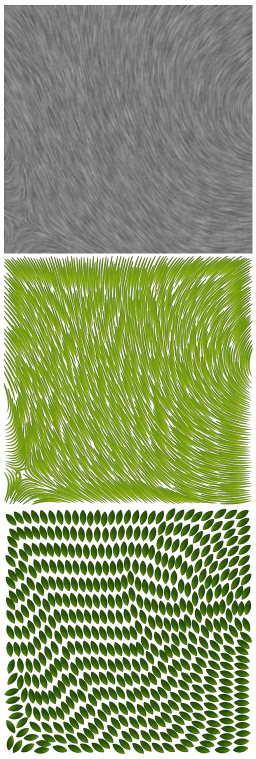
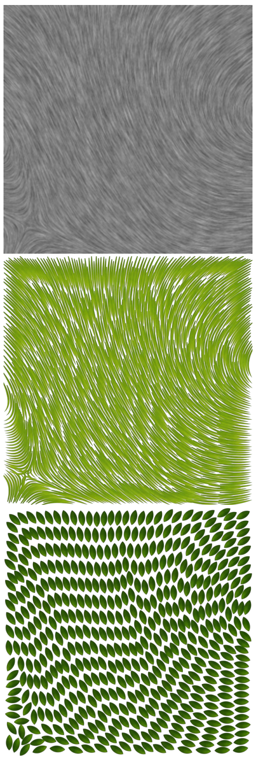
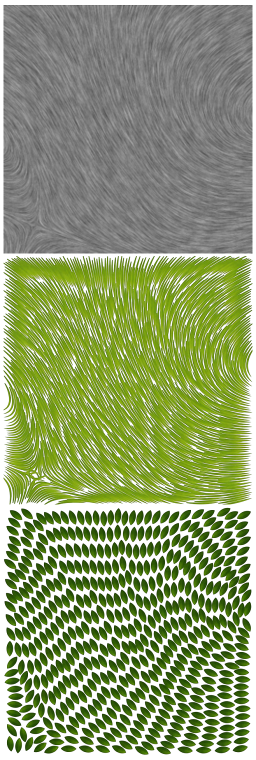
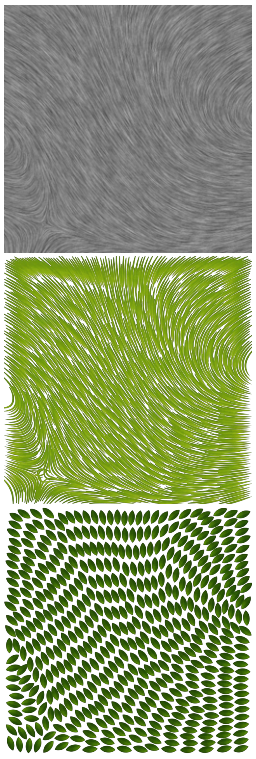

(h) *frame 32*

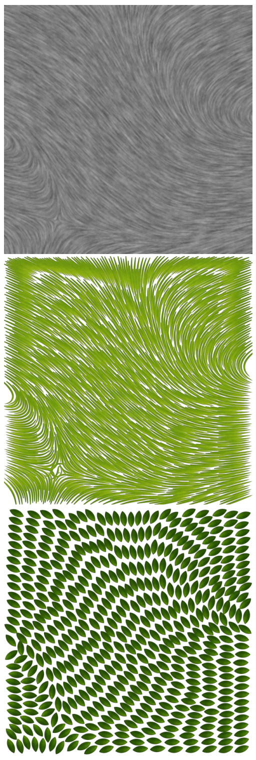
Figure B.4: Dynamic input fields from 25 to 32 frames.

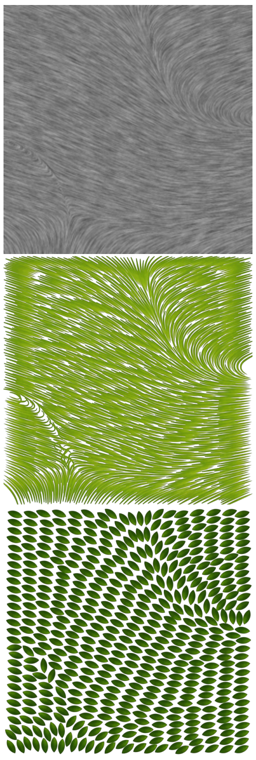

(a) *frame 33*

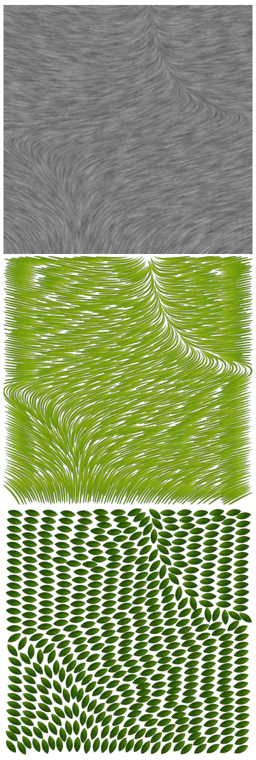

(b) *frame 34*


(c) *frame 35*


(d) *frame 36*


(e) *frame 37*


(f) *frame 38*


(g) *frame 39*

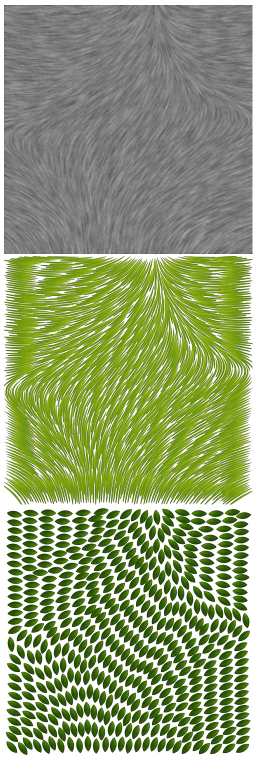
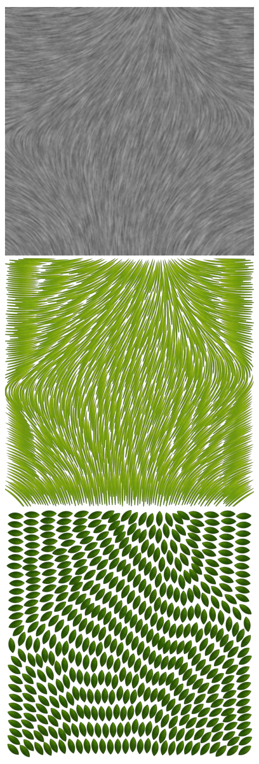
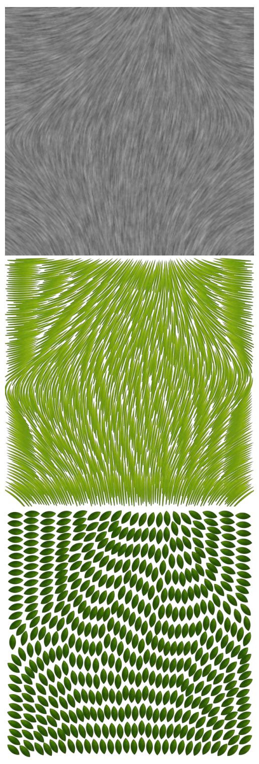
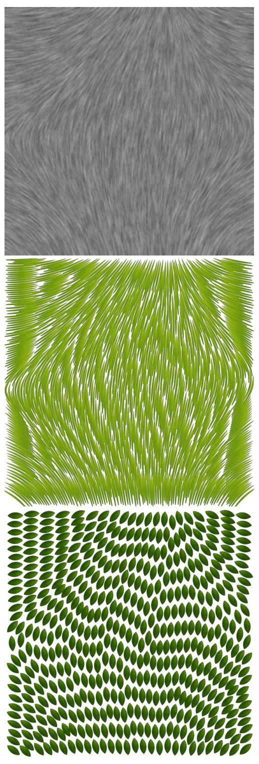
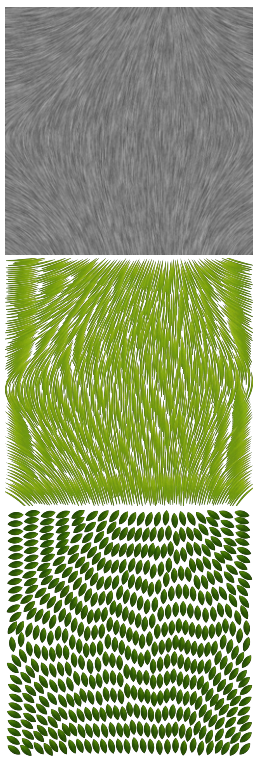

(h) *frame 40*

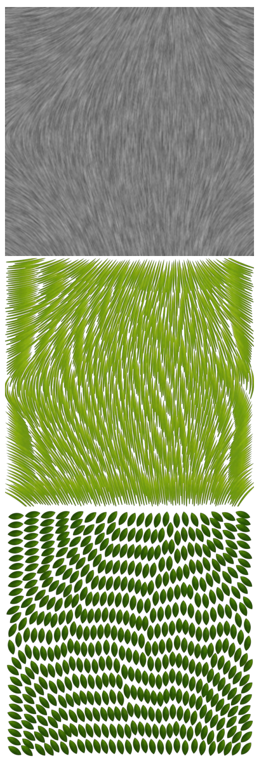
Figure B.5: Dynamic input fields from 33 to 40 frames.

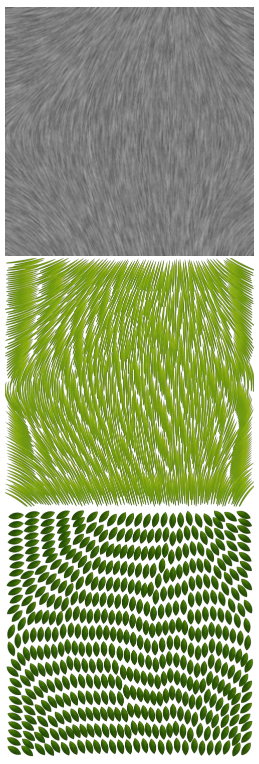

(a) *frame 41*

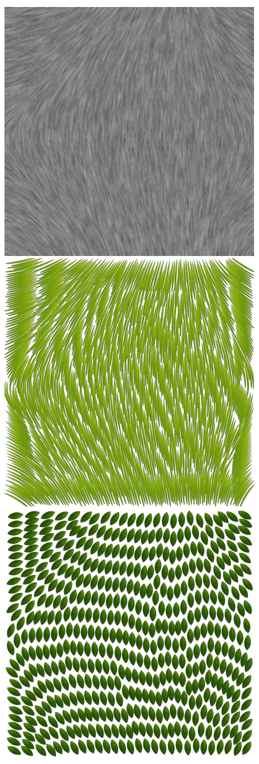

(b) *frame 42*


(c) *frame 43*


(d) *frame 44*


(e) *frame 45*


(f) *frame 46*


(g) *frame 47*

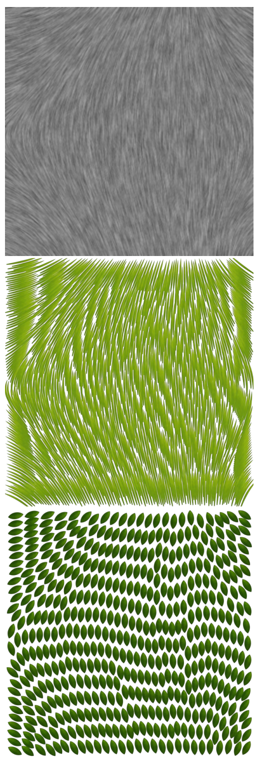
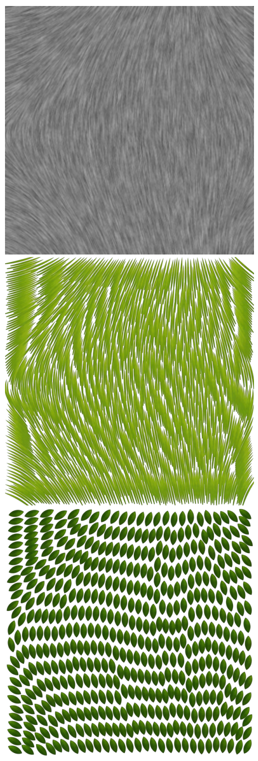
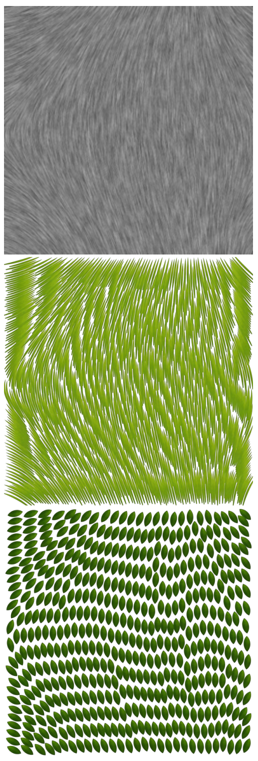
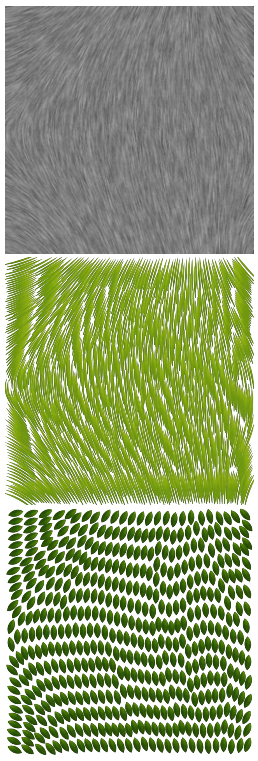
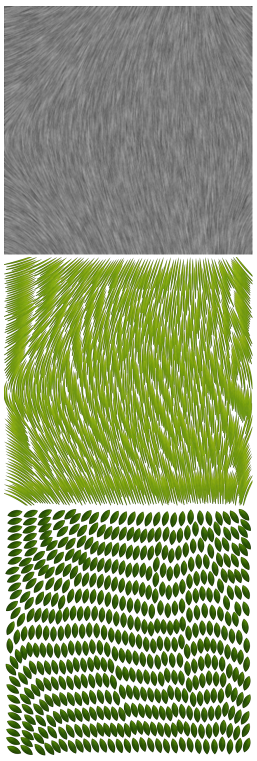

(h) *frame 48*

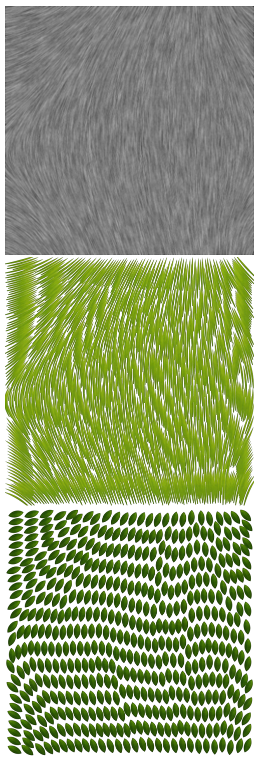
Figure B.6: Dynamic input fields from 41 to 48 frames.

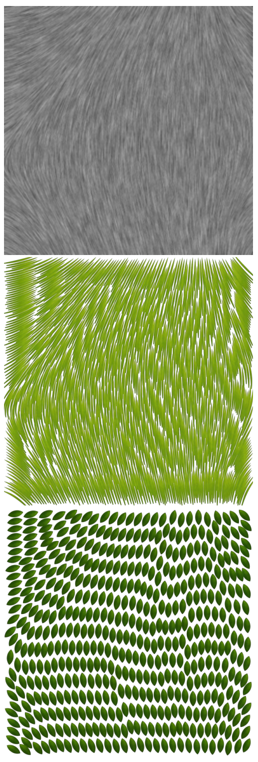

(a) *frame 49*

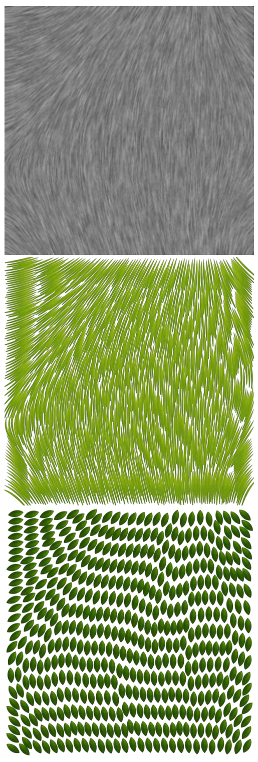

(b) *frame 50*


(c) *frame 51*


(d) *frame 52*


(e) *frame 53*


(f) *frame 54*


(g) *frame 55*

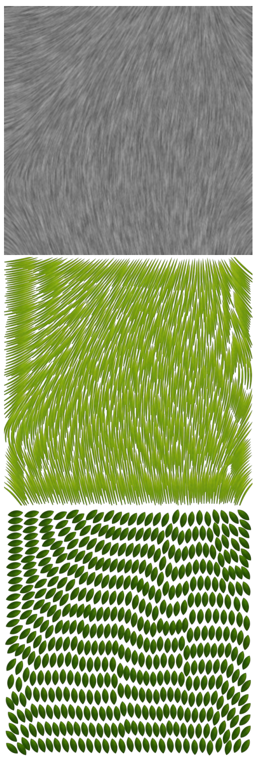
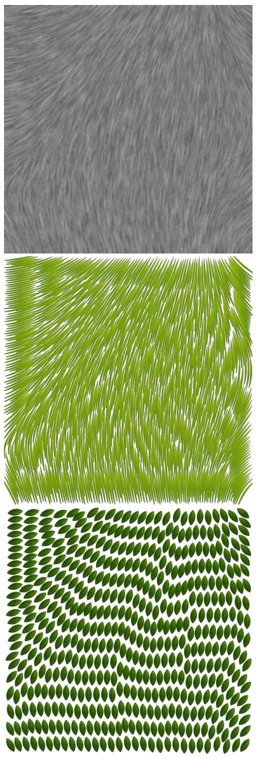
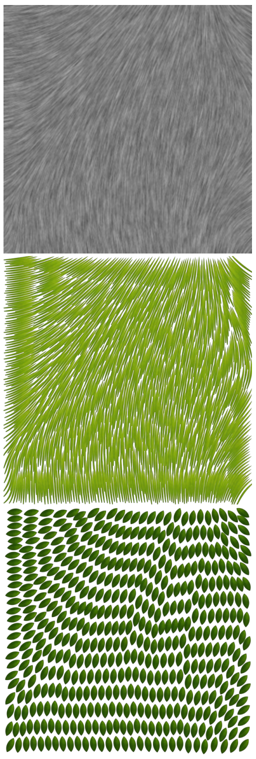
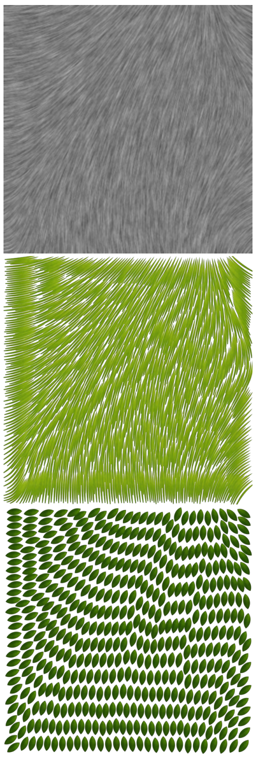
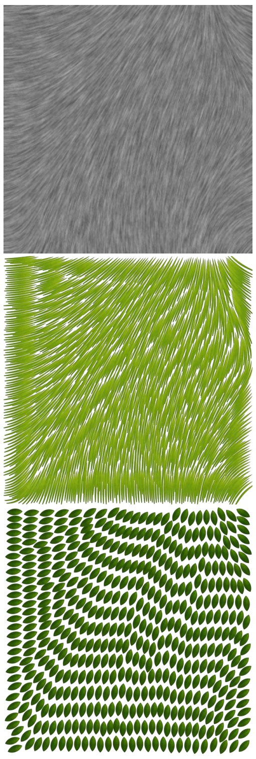

(h) *frame 56*

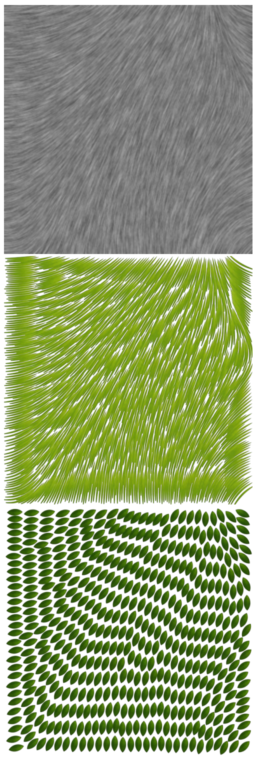
Figure B.7: Dynamic input fields from 49 to 56 frames.

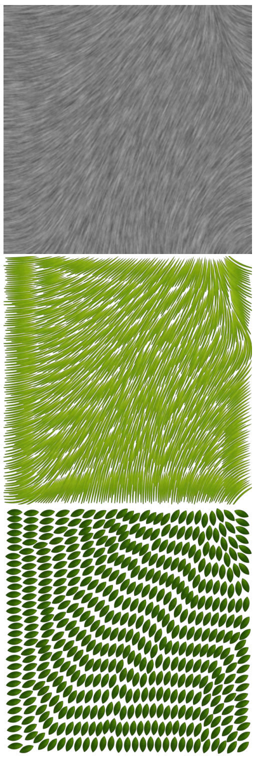

(a) *frame 57*

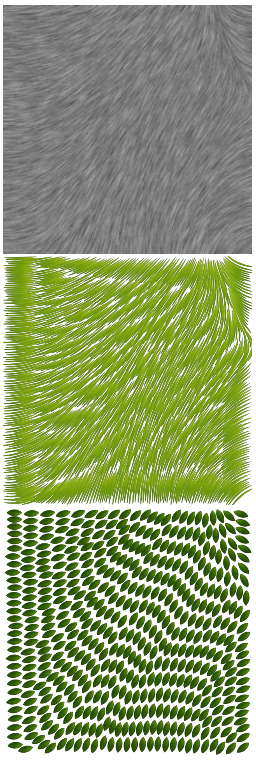

(b) *frame 58*


(c) *frame 59*


(d) *frame 60*


(e) *frame 61*


(f) *frame 62*


(g) *frame 63*

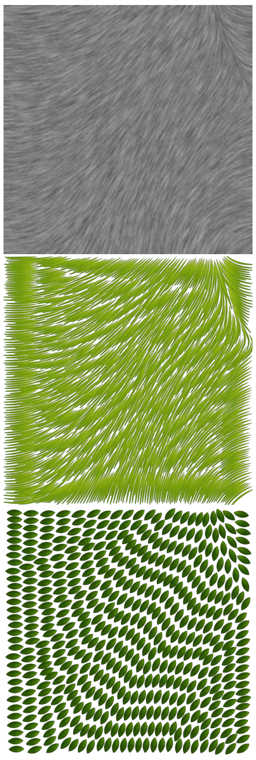
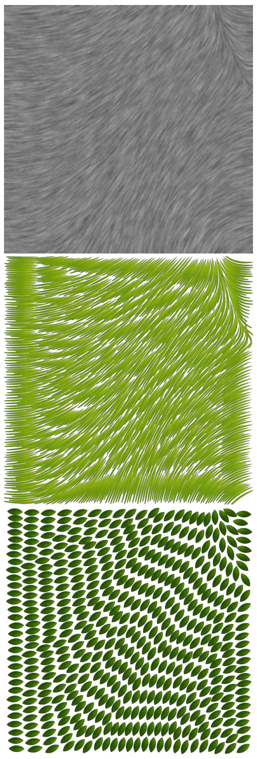
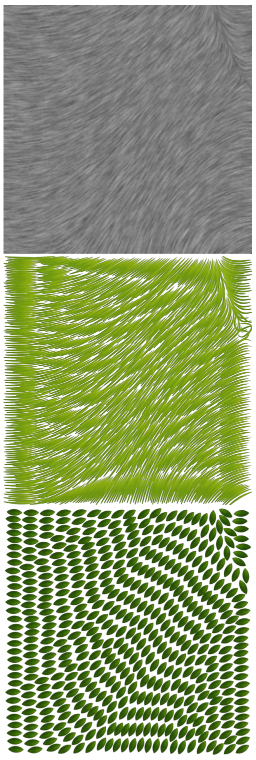
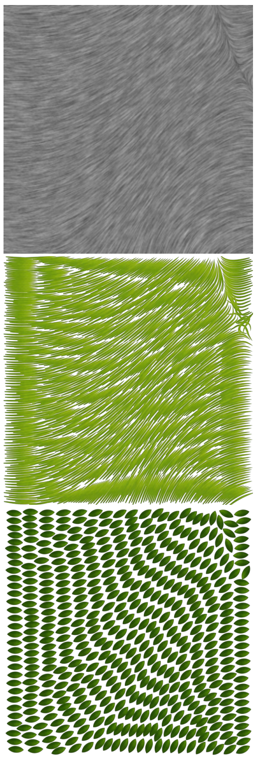
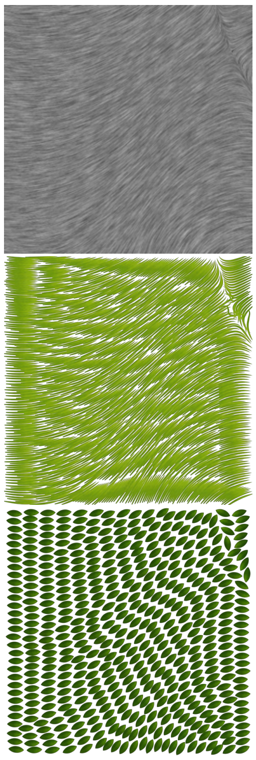

(h) *frame 64*

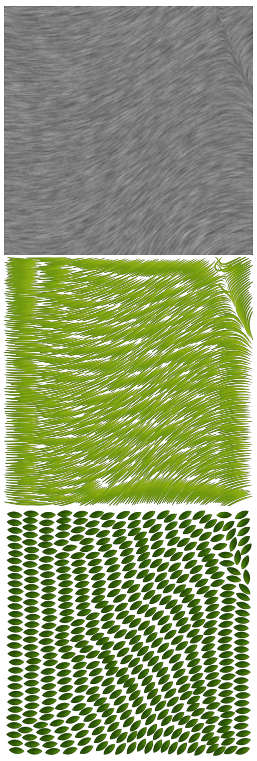
Figure B.8: Dynamic input fields from 57 to 64 frames.

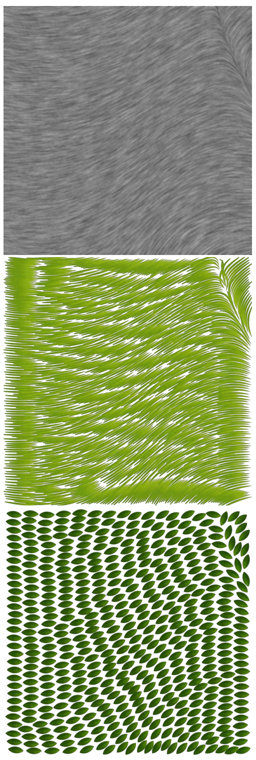

(a) *frame 65*

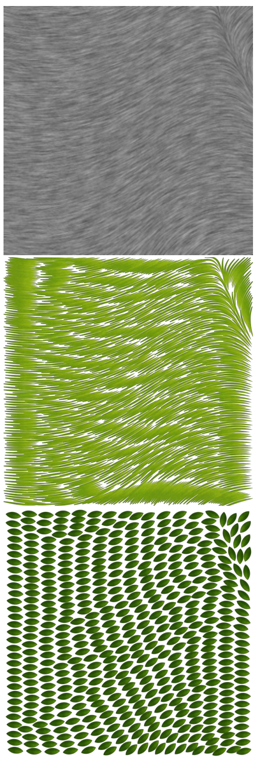

(b) *frame 66*


(c) *frame 67*


(d) *frame 68*


(e) *frame 69*


(f) *frame 70*


(g) *frame 71*

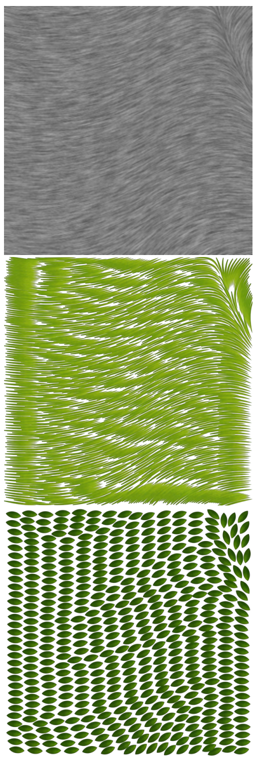
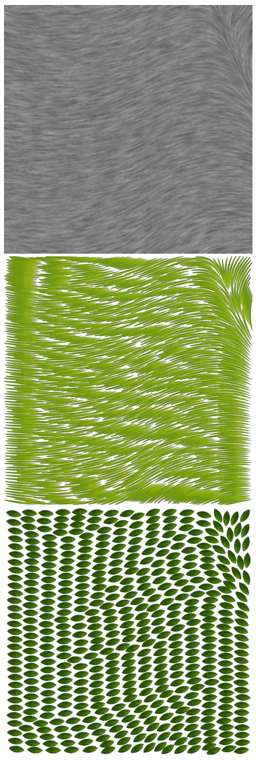
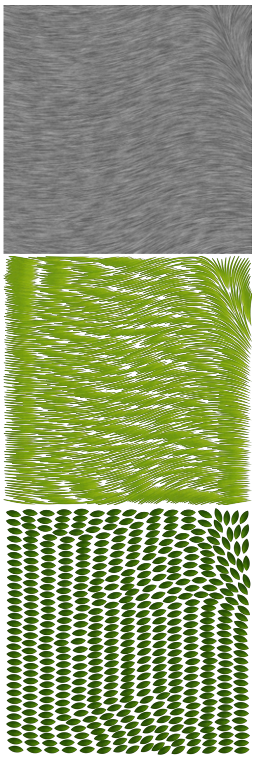
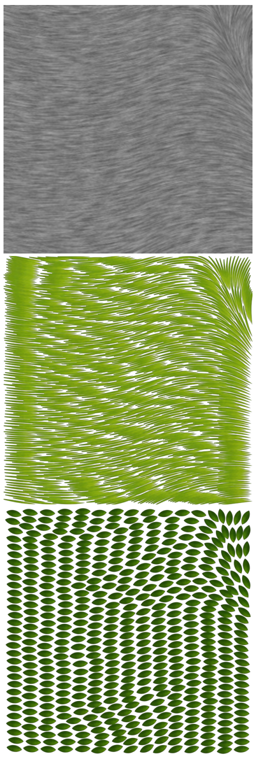
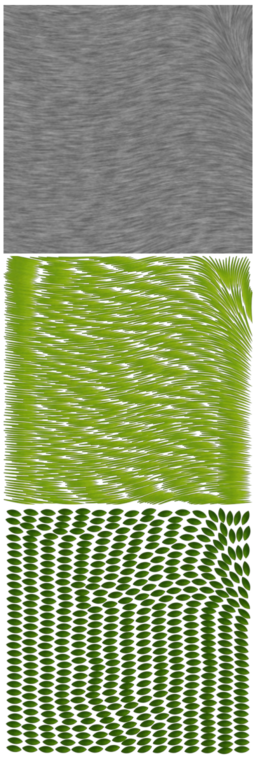

(h) *frame 72*

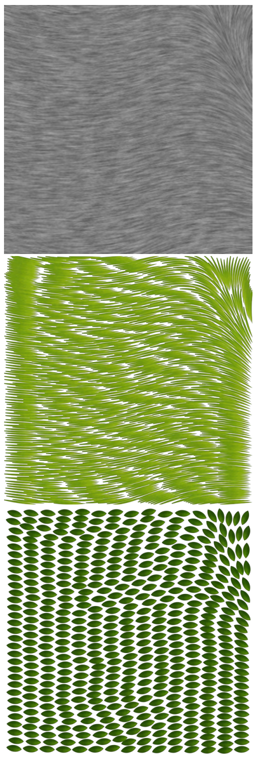
Figure B.9: Dynamic input fields from 65 to 72 frames.

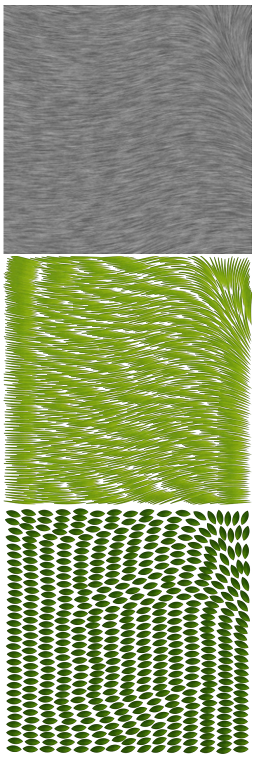

(a) *frame 73*

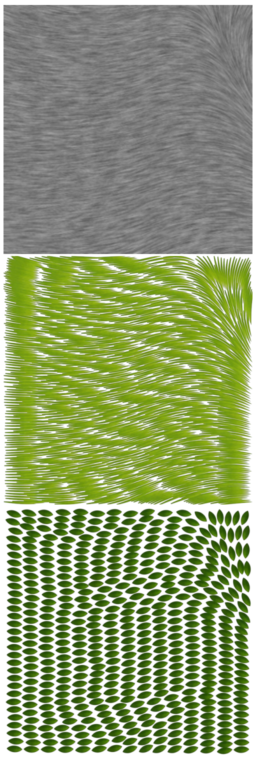

(b) *frame 74*


(c) *frame 75*


(d) *frame 76*


(e) *frame 77*


(f) *frame 78*


(g) *frame 79*

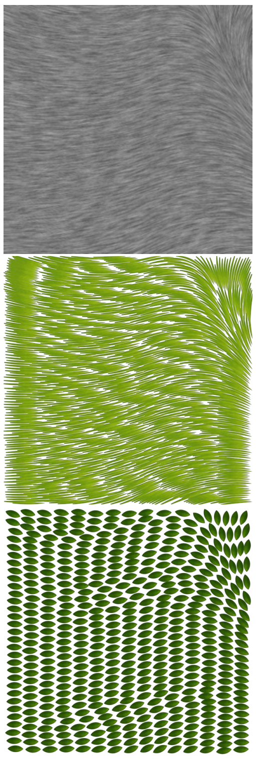
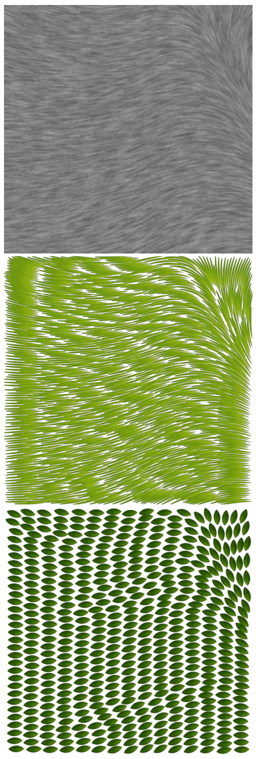
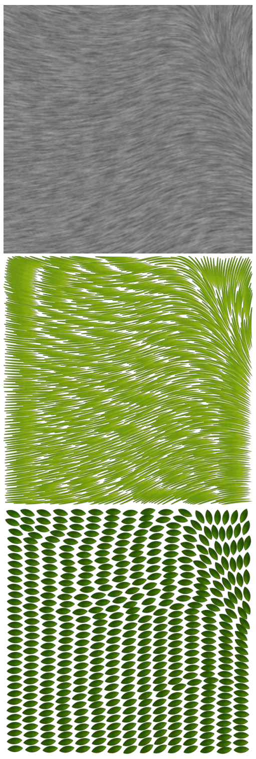
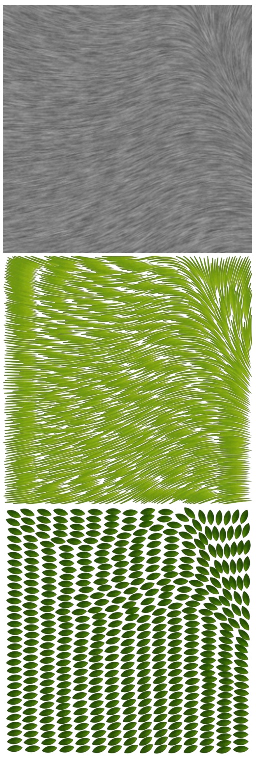
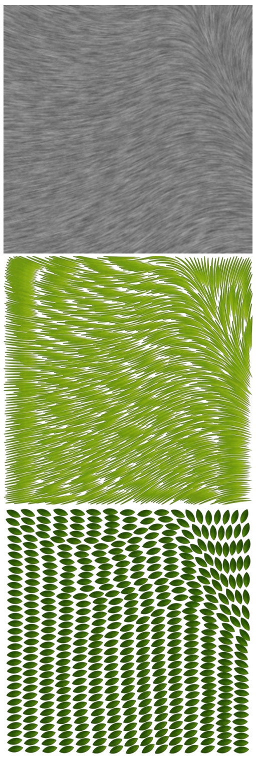

(h) *frame 80*

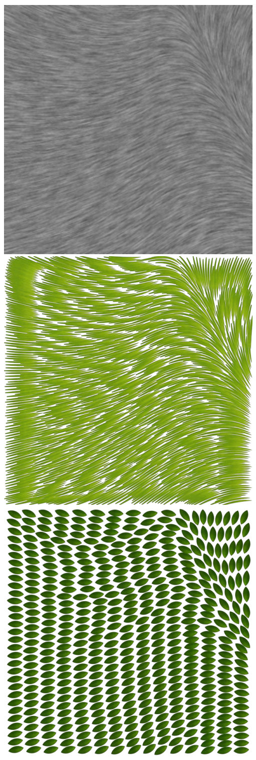
Figure B.10: Dynamic input fields from 73 to 80 frames.

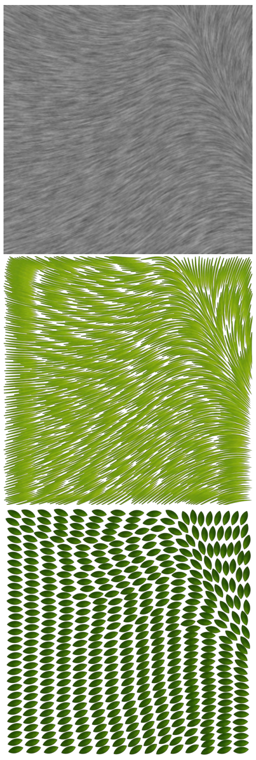

(a) *frame 81*

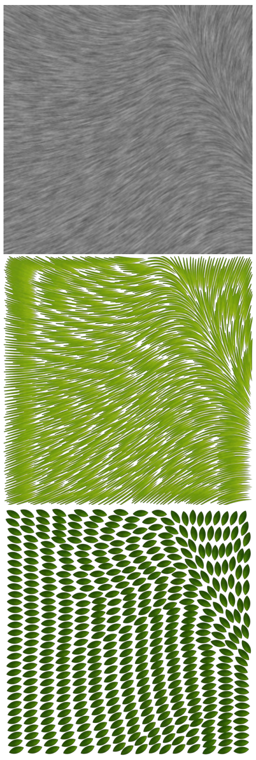

(b) *frame 82*


(c) *frame 83*


(d) *frame 84*


(e) *frame 85*


(f) *frame 86*


(g) *frame 87*

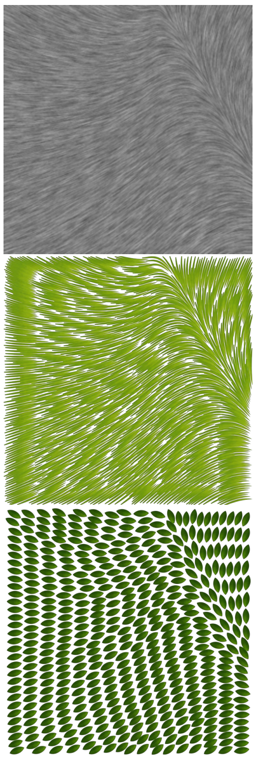
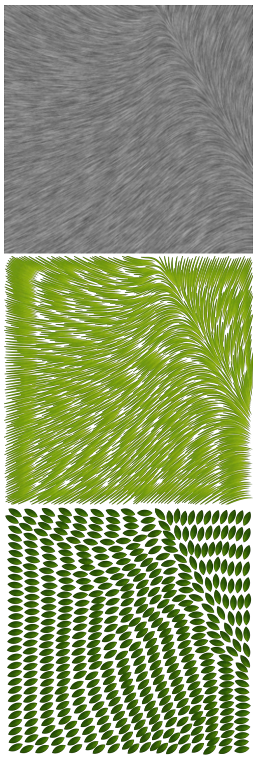
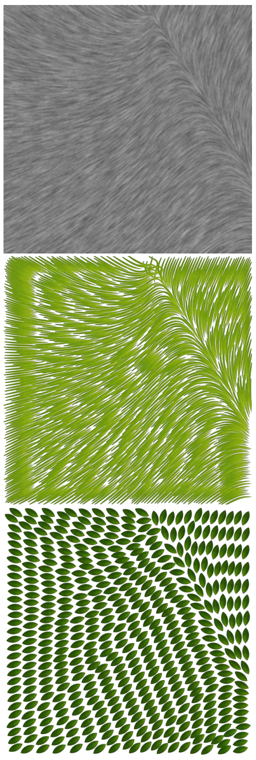
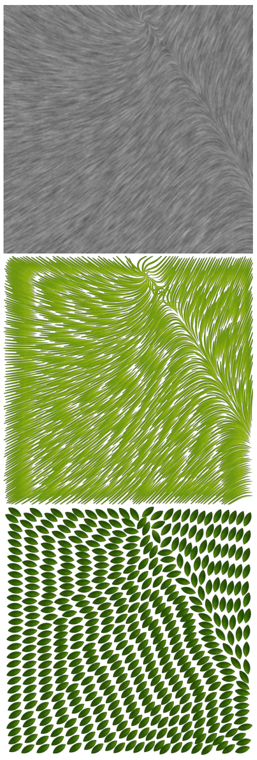
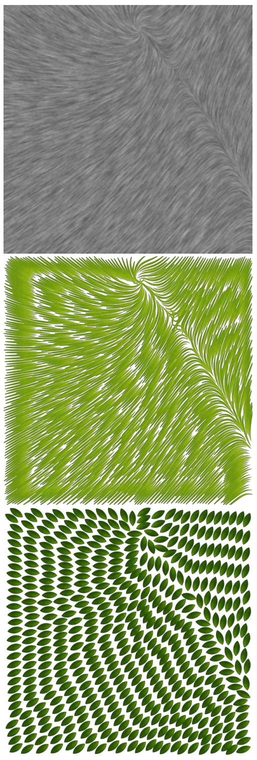

(h) *frame 88*

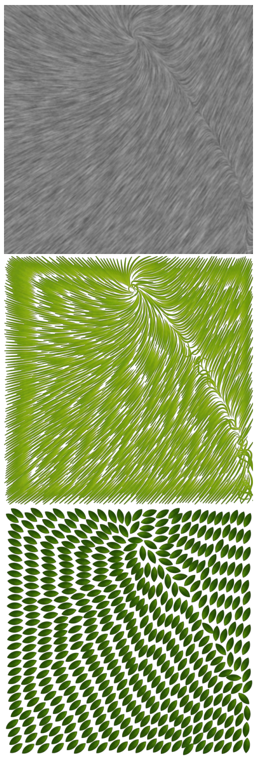
Figure B.11: Dynamic input fields from 81 to 88 frames.

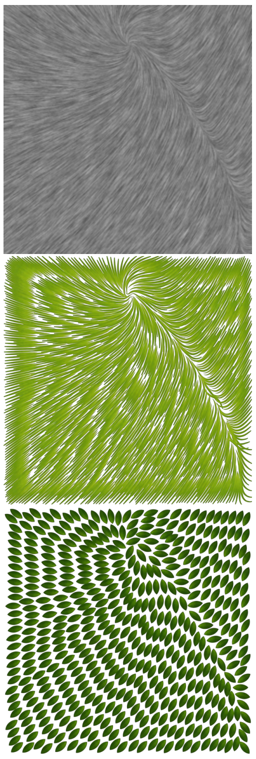

(a) *frame 89*

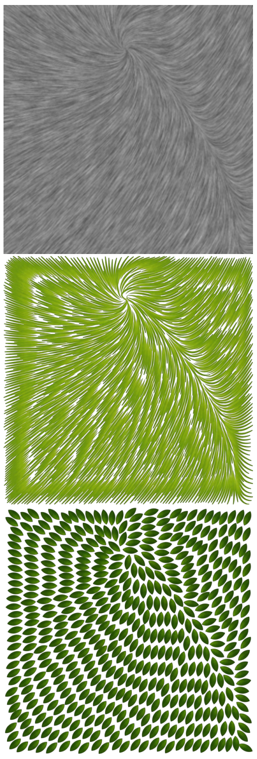

(b) *frame 90*


(c) *frame 91*


(d) *frame 92*


(e) *frame 93*


(f) *frame 94*


(g) *frame 95*

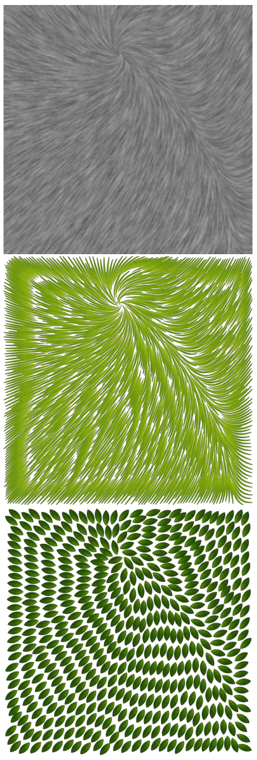
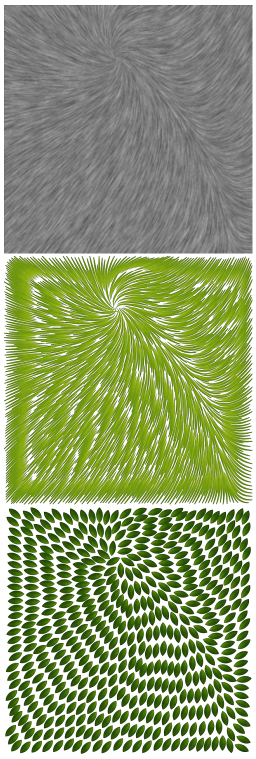
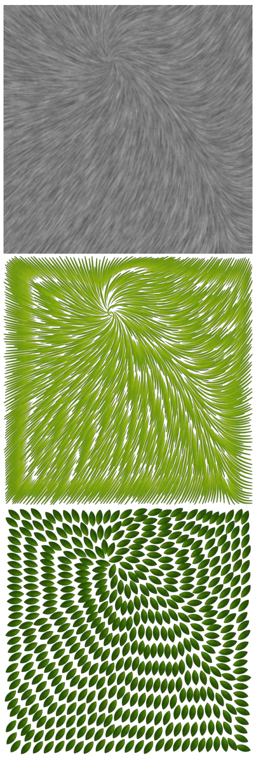
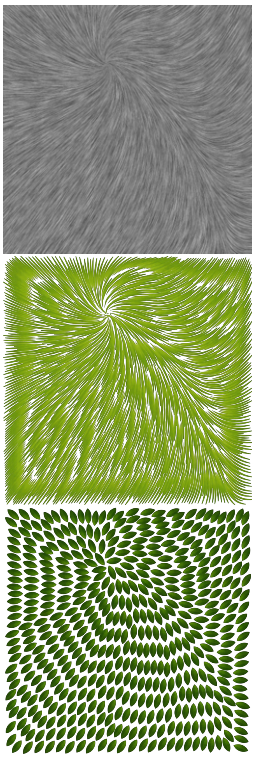
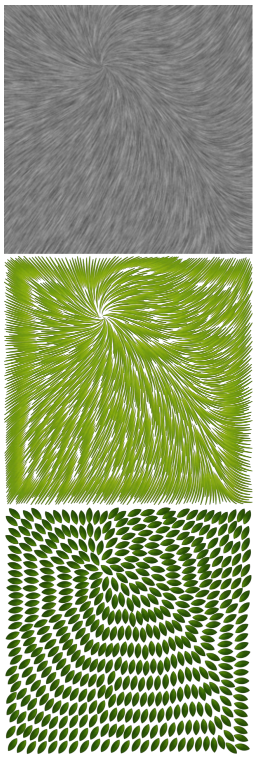

(h) *frame 96*

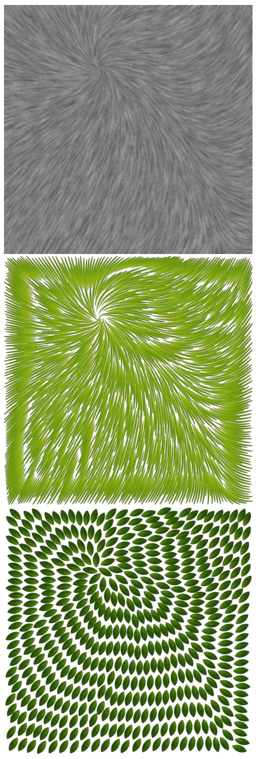
Figure B.12: Dynamic input fields from 89 to 96 frames.

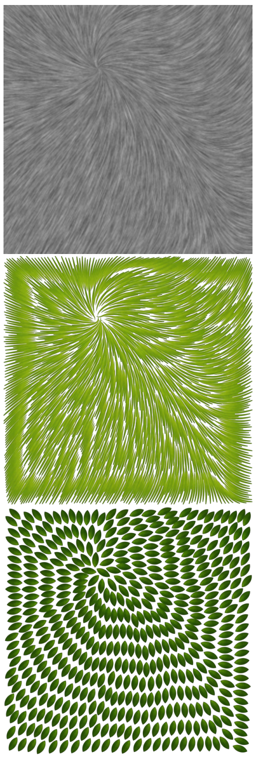

(a) *frame 97*

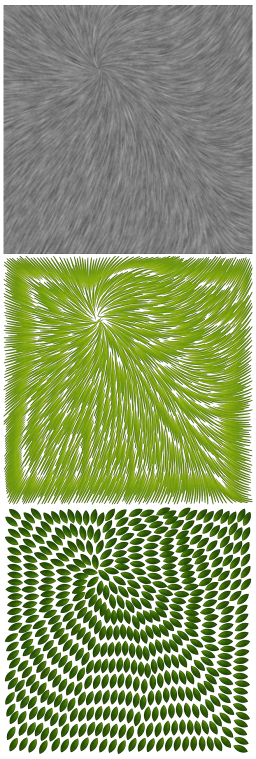

(b) *frame 98*


(c) *frame 99*


(d) *frame 100*


(e) *frame 101*


(f) *frame 102*


(g) *frame 103*

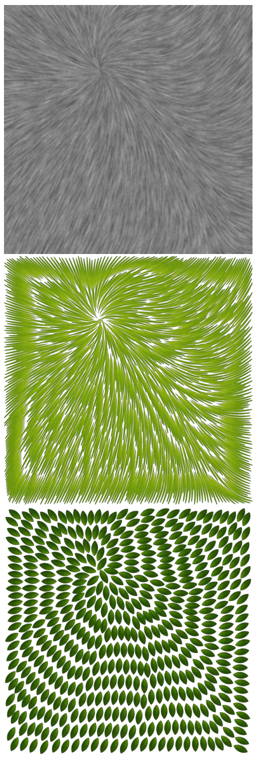
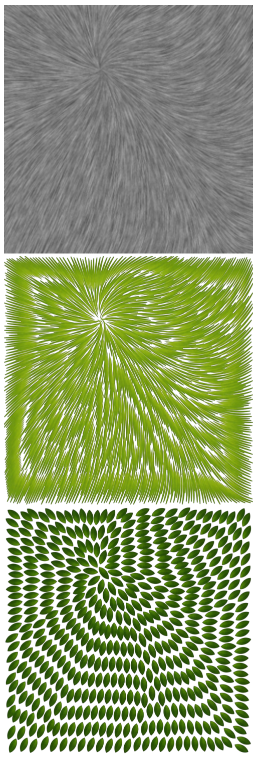
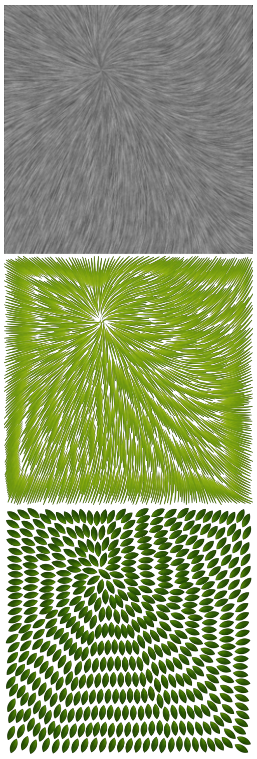
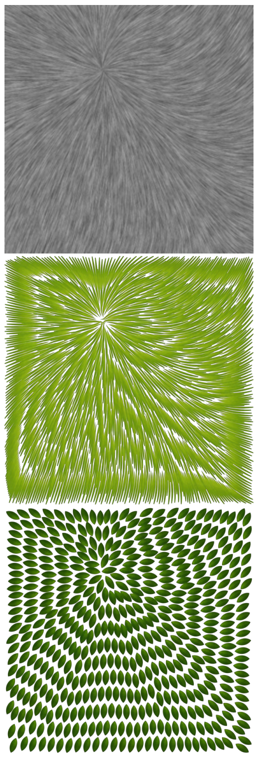
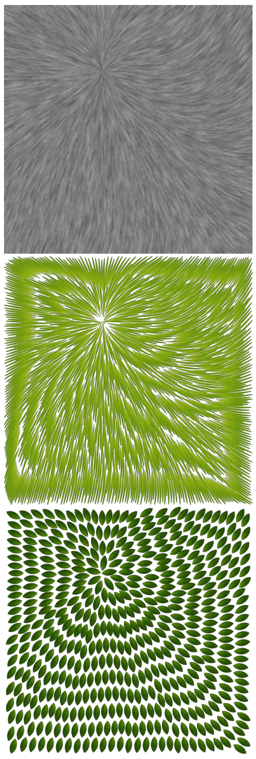

(h) *frame 104*

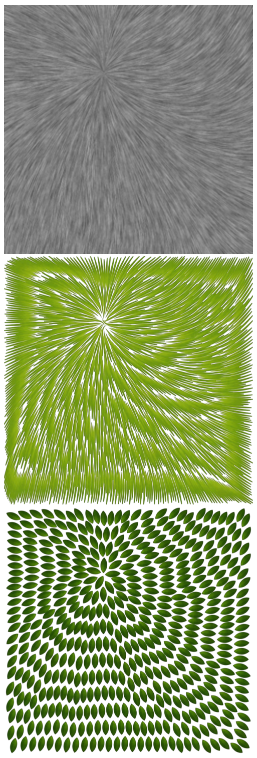
Figure B.13: Dynamic input fields from 97 to 104 frames.

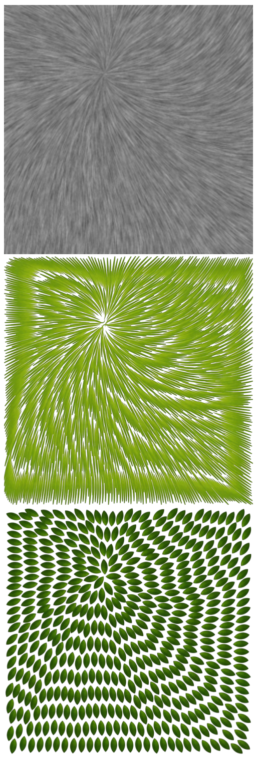

(a) *frame 105*

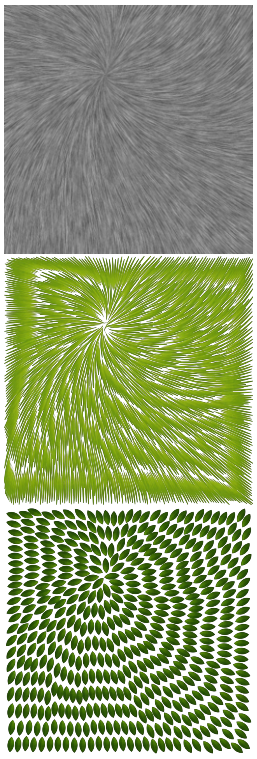

(b) *frame 106*


(c) *frame 107*


(d) *frame 108*


(e) *frame 109*


(f) *frame 110*


(g) *frame 111*

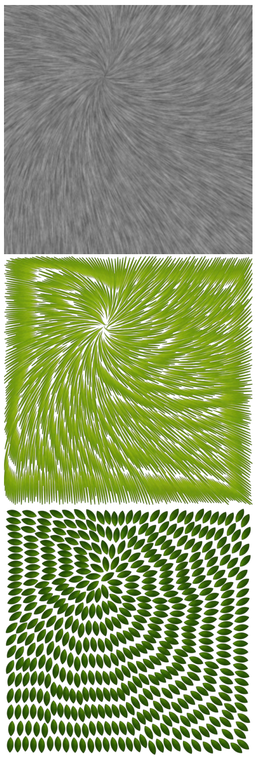
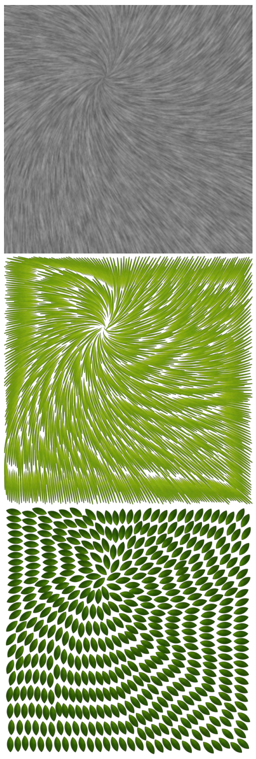
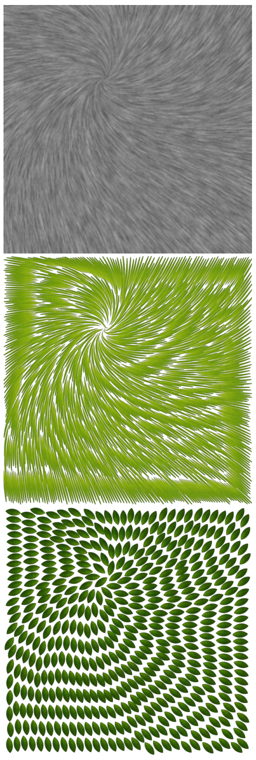
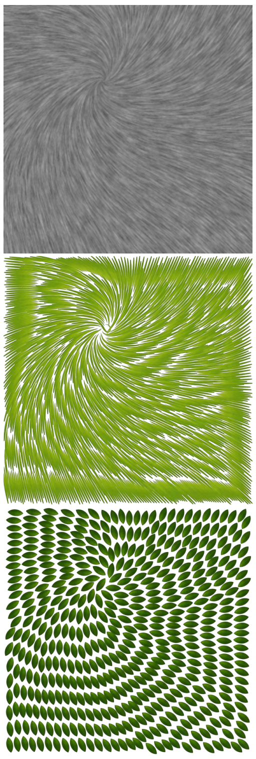
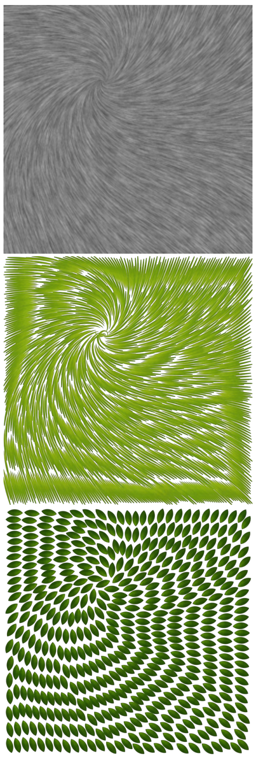

(h) *frame 112*

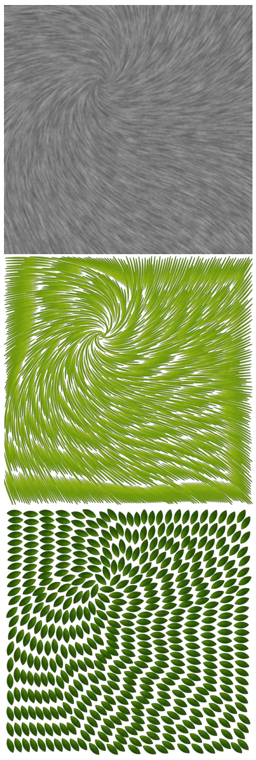
Figure B.14: Dynamic input fields from 105 to 112 frames.

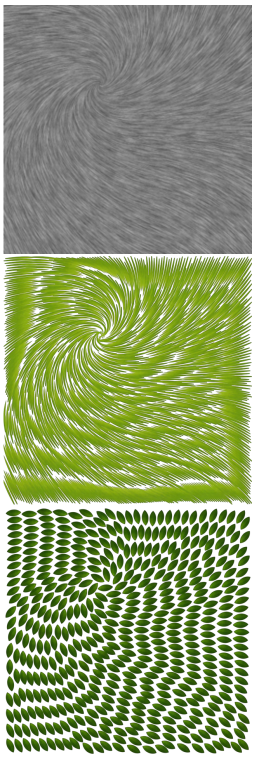

(a) *frame 113*

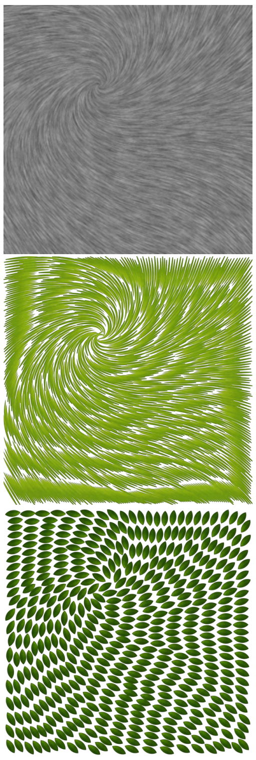

(b) *frame 114*


(c) *frame 115*


(d) *frame 116*


(e) *frame 117*


(f) *frame 118*


(g) *frame 119*

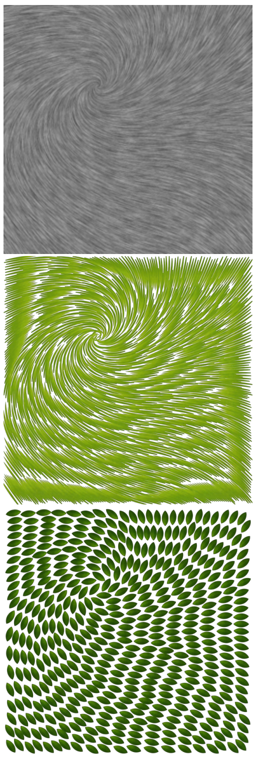

(h) *frame 120*

Figure B.15: Dynamic input fields from 113 to 120 frames.