PATEX: Exploring Pattern Variations

Paul Guerrero
University College London

Gilbert Bernstein
Stanford University

Wilmot Li
Adobe Research

Niloy J. Mitra
University College London

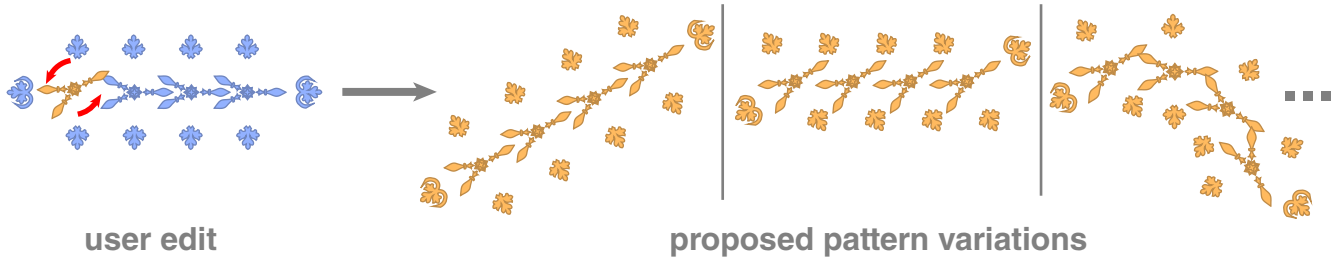


Figure 1: *We propose a method called* PATEX *to explore design variations for pattern editing. In response to a user edit,* PATEX *can generate several distinct and intuitive structure-preserving variations of the original pattern.*

Abstract

Patterns play a central role in 2D graphic design. A critical step in the design of patterns is evaluating multiple design alternatives. Exploring these alternatives with existing tools is challenging because most tools force users to work with a single fixed representation of the pattern that encodes a specific set of geometric relationships between pattern elements. However, for most patterns, there are many different interpretations of its regularity that correspond to different design variations. The exponential nature of this variation space makes the problem of finding all variations intractable. We present a method called PATEX to characterize and efficiently identify distinct and valid pattern variations, allowing users to directly navigate the variation space. Technically, we propose a novel linear approximation to handle the complexity of the problem and efficiently enumerate suitable pattern variations under proposed element movements. We also present two pattern editing interfaces that expose the detected pattern variations as suggested edits to the user. We show a diverse collection of pattern edits and variations created with PATEX. The results from our user study indicate that our suggested variations can be useful and inspirational for typical pattern editing tasks.

Keywords: patterns, arrangements, design tools, vector graphics, design space exploration, variations

Concepts: •**Computing methodologies** → **Shape analysis;**

1 Introduction

Patterns are an important component of many 2D graphic designs. In many cases, patterns play a supporting role, providing a backdrop to the primary design elements or adding texture and visual interest to filled regions. For some domains, like textile or packaging design, the pattern itself is the focus. In this work, we focus on the large class of patterns that consist of discrete elements organized into geometric arrangements (e.g., Figures 1 and 2). While such patterns can take many different forms, from highly symmetric compositions to more organic arrangements, the one common characteristic that defines all patterns is that they exhibit some amount of spatial regularity, which we can often express in terms of the geometric relationships between pattern elements. For example, in Figure 2, the diamond- and arrow-shaped elements are arranged radially around the clovers, and there are strong horizontal and vertical alignments between the four repeated motifs. Such relationships define the characteristic structure of a pattern.

A critical step in almost every design workflow is creating and evaluating multiple design alternatives. Pattern design is no exception. For example, consider the problem of decreasing the visual weight of the pattern in Figure 2a. There are many different ways to accomplish this goal, each of which may preserve, modify, or violate different sets of inter-element relationships. By creating variations of the pattern that modify the size, position, and orientation of elements in various ways, a designer can explore the space of possible solutions and only refine those that seem most promising.

However, exploring pattern variations with existing tools is challenging. On a practical level, most edits require users to manipulate many individual elements to ensure that the desired inter-element relationships are preserved. For example, depending on how the elements are grouped, the edit in Figure 2b requires 24 separate editing operations. Of course, one could argue that there is a parametric representation of the pattern that would be much more convenient for this particular edit, but this argument raises the main conceptual problem with existing tools. While any given representation of a

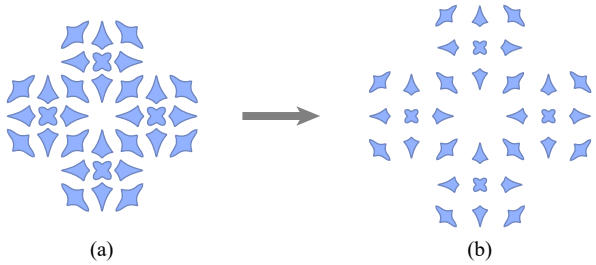


Figure 2: *Re-targeting a pattern is not well supported by current drawing tools. Even this simple re-targeting of the pattern to have less visual weight requires* 24 *separate editing operations. In Figure 9, we show that our method only requires four operations.*

SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925950

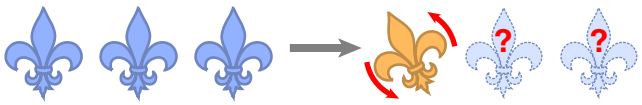


Figure 3: *A typical edit in a small pattern. The orange element is rotated by the user. Our goal is to find several variations for the arrangement of the remaining elements. See Figure 4 for variations proposed by* PATEX.

pattern may encode aspects of its regularity (e.g., repetitions of a symbol, reflective symmetry, etc.) in a way that facilitates the creation of *some* edits, different variations are often defined by *different* interpretations of the pattern regularity that are hard to encode in a single, fixed representation. For example, the four variations in Figure 6 correspond to two different interpretations of the relevant geometric relationships in the pattern.

We propose a different approach to exploring pattern variations that does not impose or assume any one given representation of the regularity that defines a pattern. Rather than explicitly encoding and modifying relationships to create variations, the user starts by directly manipulating individual elements. In response, the system automatically generates suggested variations of the pattern that are consistent with the user edits but also distinct from one another. The variations correspond to different interpretations of the pattern, each producing a different modification of the pattern elements in response to the user edits. At any point, the user can jump to a suggested variation and continue editing. Our approach does not force users to work with a fixed representation of the pattern structure that may constrain edits in undesirable (and frustrating!) ways. At the same time, our suggestions allow users to efficiently propagate individual edits to other elements when appropriate. Moreover, suggested variations may inspire users to consider different design directions that are still consistent with their manual edits.

There are two core challenges to realizing this approach. First, given all the geometric relationships that define different, potentially relevant types of regularity in a pattern, there are an exponential number of possible variations to consider. Thus, it is very difficult to efficiently compute a suitable subset of suggestions that are both intuitive and distinct based on the user edits. Take for example the pattern edit in Figure 3. Even for this very simple pattern, there are several distinct variations, some of which are shown in Figure 4. The second challenge is how to present suggested variations to users in a way that does not impede their editing interactions while also facilitating browsing and comparison of the suggestions.

Our work addresses both these challenges. We define a *pattern graph* representation that encodes the regular structure of a pattern. In contrast to traditional pattern representations, the pattern graph is overcomplete; that is, rather than relying on a single interpretation of which relationships define the characteristic structure of the pattern, the pattern graph encodes a large set of relationships that correspond to many different interpretations of the pattern. Given this pattern graph, we can identify different possible interpretations for any given edit and compute a corresponding variation via a non-linear optimization. We also introduce a method to efficiently find a subset of intuitive and distinct variations. Our technique combines random sampling of the variation space favoring well-determined variations with a fast, linear approximation of the variation optimization. Finally, we propose two user interfaces for presenting suggestions: one that shows variations on canvas as potential 'Autocomplete' targets for a given edit; and another that shows variations as small multiples in an explorer interface next to the canvas. We demonstrate how our approach facilitates editing of a wide range of patterns. The results from our exploratory user study indicate that our automated suggestions are generally seen as useful and inspirational, although the perceived value of the suggestions is somewhat task-dependent.

2 Related Work

Pattern creation and editing. Existing methods largely support element level manipulations, or different snapping strategies to align groups of elements into grid-like arrangements. Tools like Illustrator, InDesign, Inkscape, Visio, etc. are commonly used for creating vector drawings. While they implement a mix of modes to create individual elements, support for authoring layout arrangements remains limited. These tools are largely based on earlier efforts in element-level manipulation [Nelson 1985; Gleicher 1992; Ryall et al. 1997]. Users can either manually place individual elements, supported by various snapping modalities [Bier and Stone 1986; Baudisch et al. 2005], or order groups of elements as grid-based arrangements using 'distribute equally' or 'align to elements' options. More advanced methods use smarter local edits [Igarashi et al. 1997; Tsandilas et al. 2015], or more global variants [Pavlidis and Van Wyk 1985] that support snapping to regular arrangements (e.g., 1D or 2D grids). Nan et al. [2011] develop a computational model to encode and abstract shape arrangements in architectural drawings based on Gestalt rules. Recently, a group-aware layout arrangement has been proposed by Xu et al. [2015] by automatically decomposing a 2D layout into multiple 1D groups. Since automatically snapping to regular arrangements can be error prone, Xu et al. [2014] developed a global beautification interface that allows users to incrementally correct proposed snapping by deleting or accepting proposed alignments. In contrast, we focus on more general regular arrangements, and describe how to efficiently identify and present richer pattern variations.

In the context of document layout, grids have also been proposed for more flexible layouts [Jacobs et al. 2003]. Alternately, data-driven layouts have been utilized for graphics designs [O'Donovan et al. 2014] or to create artistic packings [Reinert et al. 2013]. These methods algorithmically propose a single final layout, rather than allowing the user to explore a space of possible layout variations.

Constrained modeling. There exists a large body of work on constraint-based modeling, particularly in the CAD community (cf., [Chen and Hoffmann 1995]) that treats edits in a constrained manipulation setting. These methods explore different alternatives to perform shape or pattern edits while conforming to specified (or inferred) relations. Gleicher and Witkin [1991] interpreted graphical elements as physical objects to propose a unified editing interface based on physics-inspired constraints. Physically-based manipulation was also proposed for discrete and continuous models [Harada et al. 1995]. Efforts in mechanical engineering [Daniel and Lucas 1997; Yvars 2008] resulted in declarative methods to specify geometric constraints for a given mechanical design. Huang et al. [2009] propose structure preserving optimization for resizing images and vector art. In the context of shape editing, several methods have been proposed to automatically determine relevant geometric relationships between parts of 3D models [Gal et al. 2009; Xu et al. 2009; Bokeloh et al. 2012; Zheng et al. 2012]. These relationships enable high-level editing and synthesis of 3D models. The Lillicon system [Bernstein and Li 2015] proposes transient widgets to support scale variations in icon designs.

PATEX takes a different avenue. Instead of conforming to a given set of constraints, it directly seeks out variations with *different* sets of constraints to expose a richer set of pattern variations. For example, in Figure 3, a fully constrained edit would lead to a 'trivial' rigid rotation of the original pattern. By allowing some of the original constraints to break, PATEX exposes a much larger set of non-trivial patterns as shown in Figure 4.

Design space exploration. In the context of novel synthesis, algorithms exist that support synthesis with topological variations. However, they either do not support direct user edits, or are domain-

Figure 4: *Variations of element arrangements for the edit shown in Figure 3. Even in this very simple pattern, several alternatives are possible.*

specific, are sampling based, or need to be based on a large dataset of pre-existing shapes. Talton et al. [2009] propose a method for estimating and sampling from high dimensional parametric design spaces to support exploratory design. There has been interesting work on sampling pattern designs from a distribution of options [Yeh and Mech 2009; Yeh et al. 2012; Yeh et al. 2013]. Alhashim et al. [2014] propose an interactive exploratory tool for part-based 3D modeling based on a spatio-structural graph composed of medial curves and sheets of model parts. They demonstrate how such a representation can be used to explore different split and merge options for shape synthesis with topology variations. This allows interpolating between shapes with different topologies by establishing part level shape correspondence. Fish et al. [2014] learn distributions of attribute values from shape collections that can be used to efficiently explore these collections and create novel designs. Instead, we directly characterize the set of valid pattern variations based on an initial pattern and a user edit to efficiently identify different yet valid patterns.

3 Overview

Each pattern is made up of a set of two-dimensional polygonal shapes, arranged in some way. We call the assignment of position and rotation to each of these *elements* an *arrangement*. Looking at an arrangement of the elements, we can observe different *relationships* (i.e., distance, rotation, scale) first between pairs of elements, and then recursively between the relationships themselves. This hierarchy of observed relationships forms a *pattern graph*, which encodes the regularity and structure of the pattern (§4). Since our focus lies on finding variations, rather than constructing pattern graphs, we assume this pattern graph to be available, see §8.

Given this setup, we want our system to respond to a user edit (modification of a small number of individual elements) by suggesting various possible modifications of the rest of the pattern, consistent both with the user edit and part of the structure of the pattern. That is, given an arrangement, its derived pattern graph, and a set of edited elements, our algorithm should output a set of new, candidate arrangements.

If we insist that all of the relationships in the pattern graph are precisely maintained by our algorithm, then the only possible response to a user edit will be rigid transformations of the whole pattern. This is because the relationships, when interpreted as constraints, are overcomplete. Therefore, we must relax some of these relationships. Every relationship is either preserved, updated (to the value suggested by the edit), or allowed to change freely; A complete assignment of each relationship to one of these 3 options defines a variation (§5). Given a such a variation, we can solve an optimization problem to find a new arrangement of the elements (§5.1). In this way, we reduce the problem of finding a diverse set of structure-preserving edits to the problem of finding a diverse set of variations.

The problem of finding a good candidate set of variations is challenging because (i) it requires searching an exponentially large space of possibilities, (ii) not all such variations are compatible with user edits (i.e., *valid*), and (iii) different variations do not necessarily produce different final arrangements. By linearly approximating the optimization problem, our algorithm is able to efficiently reason about the final arrangement, both its validity and diversity (§5.2).

Finally, we explore different interfaces for exposing the results of this algorithm to users (§6).

4 The Pattern Graph

Elements of a pattern are given as a set of simple polygons $\mathbf{E} = \{E_1, \ldots, E_n\}$ and the structure of the pattern is represented by a set of geometric relationships between elements $\mathbf{R}^E = \{R_1^E, \ldots, R_k^E\}$, where

$$R^E : \mathbf{E} \times \mathbf{E} \to \mathbb{R}.$$

These functions quantify the geometric relationship between two elements of the pattern. For example, the distance between their centroids, or the angular difference between their orientations. Note that multiple relationships use the same type of function. A complete list is given in Appendix A.

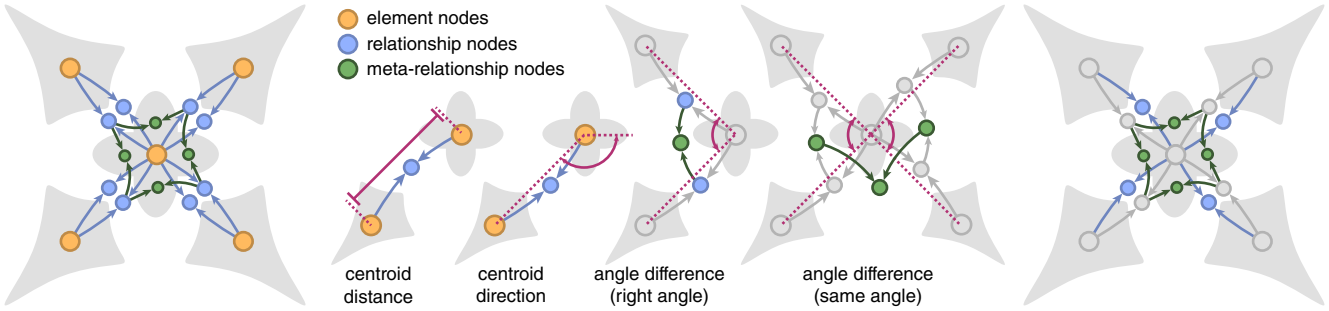
Relationships on element pairs can express *simple* geometric relationships. More complex relationships, like equidistance, or in-betweenness involve more than two elements. Rather than using n-ary relationships, we introduce relationships $\mathbf{R}^M = \{R_1^M, \ldots, R_l^M\}$ between relationship values, called *meta-relationships*, to handle these cases:

$$R^M : \mathbb{R} \times \mathbb{R} \to \mathbb{R},$$

where the inputs are two relationships values. Meta-relationships can also take values of other meta-relationships as input, allowing for a more complex pattern structure. This approach allows our graph to model any scalar-valued function that is decomposable into a tree of differentiable functions that take two inputs (like many common binary operators). This tree would have elements at the leafs, connected by relationships and meta-relationships. This type of functions includes many n-ary relationships like equidistant and collinear. We denote the full set of relationships of a pattern as $\mathbf{R} = \mathbf{R}^E \cup \mathbf{R}^M$, examples are shown in Figure 5b.

The structure of a pattern can then be represented as a directed graph $\mathcal{G} = (\mathbf{N}, \mathbf{A})$, where the nodes $\mathbf{N} = \mathbf{E} \cup \mathbf{R}$ are elements and relationships. Edges $\mathbf{A} = \{(N_{i1}, R_i), (N_{i2}, R_i)\}_{i=1\ldots k+l}$ connect relationships R_i to their two input nodes N_{i1} and N_{i2}, which might either be elements or relationships. We call this DAG the *pattern graph*, an example is shown in Figure 5a.

Relationship groups for repeated structures. Repeated structures in a pattern are exhibited in the pattern graph as groups of relationships with similar values (see Figure 5c). They may indicate symmetries or regularity in the pattern. To explicitly model these repetitions, we define (possibly overlapping) groups $\mathbf{G} = \{G_1, \ldots, G_h\}$ of relationships $G_i = \{R_{i1}, \ldots, R_{in}\}$. Relationships in a group always maintain the similarity of their values and breaking a group can mean breaking part of the pattern's symmetry or regularity. For convenience of notation, relationships that are not member of a group are assumed to form a group of size 1.



(a) pattern graph (b) geometric relationships (c) relationship groups

Figure 5: *The pattern graph describes the element arrangement of a pattern. (b) Pairs of elements are related by geometric relationships. Relationship values can in turn be related by meta-relationships. (a) These relationships form a directed graph. (c) Relationship groups describe repeating structures in the graph.*

5 Pattern Variations

The user starts a pattern edit by modifying one or more elements indicated by $\mathbf{E}_\Delta$. Our goal is to create several pattern variations based on this edit that each maintain part of the pattern structure, while allowing a different part of the structure to change. In the pattern graph, this means that some relationship values remain constant while others change.

In its initial state, the pattern fulfills all relationships of the pattern graph (see Figure 6b). A user edit introduces changes to the relationships, indicating which parts of the structure the user is potentially interested in changing, as illustrated in Figure 6c. Let $\mathbf{R}_\Delta \in \mathbf{R}$ be the changed relationships. We decide for each relationship in $\mathbf{R}_\Delta$, whether to constrain it to its new value, to its original value, or leave it unconstrained. For all unmodified relationships, $\mathbf{R} - \mathbf{R}_\Delta$ we can only decide between the latter two options. This gives us three sets: $\mathbf{R}_o$, $\mathbf{R}_c$, and $\mathbf{R}_u$, corresponding to relationships constrained to their original value, constrained to their changed value, and relationships that are unconstrained. We generate variations by assigning each relationship to one of these three sets. Each assignment will give us a pattern variation, a few examples are shown in Figure 6d. A specific assignment can be described by an indicator vector:

$$I_i = \begin{cases} 0 & \text{if } R_i \in \mathbf{R}_u \\ 1 & \text{if } R_i \in \mathbf{R}_o \\ 2 & \text{if } R_i \in \mathbf{R}_c \end{cases}$$

All possible combinations of assignments give us the exponential number of variations we can potentially generate.

5.1 Computing a Pattern Variation

Given a pattern defined by $\mathcal{G}$, a variation of this pattern is fully defined by the user edit $\mathbf{E}_\Delta$ and an assignment I. Computing the actual element arrangement corresponding to this variation can be framed as an optimization problem, where we modify the pattern elements and relationship values to satisfy the constrained relationships $\mathbf{R}_o$ and $\mathbf{R}_c$, while including the proposed modification of the elements $\mathbf{E}_\Delta$ as closely as possible.

The degrees of freedom of a pattern modification are given by the element *poses*: position, orientation, and uniform scale of each pattern element[1], while possible changes to the pattern structure are defined by the relationship values. The state of a pattern can then be encoded in a vector of dimension $4n+m$ where n is the number of elements and m the number of relationships:

$$\mu = (x_1, y_1, \theta_1, s_1, \ldots, x_n, y_n, \theta_n, s_n, r_1, \ldots, r_m),$$

(x, y) being the centroid position, θ the orientation and s the scale of an element, and r the value of a relationship. Note that the relationship values are fully determined by the state of the pattern elements, so they do not introduce new degrees of freedom. However, explicitly including them in the state reduces the direct interaction between variables and simplifies the optimization. For clarity, we will denote all entries of the state corresponding to the set of nodes $\mathbf{N}$ as $\mu_{\{\mathbf{N}\}}$, for example, $\mu_{\{N_i\}} = (x_j, y_j, \theta_j, s_j)$ if $N_i = E_j \in \mathbf{E}$ and $\mu_{\{N_i\}} = r_j$ if $N_i = R_j \in \mathbf{R}$.

Having both element poses and relationship values in the state allows configurations where the relationship values do not correspond to element positions. We say that a state is only *valid* if element poses fulfill the relationship values of the state. Let $\sigma(\mu)$ be the state obtained by evaluating all relationships from their input nodes in the state μ:

$$\sigma_i(\mu) = \begin{cases} N_i\big(T(N_{i1}, \mu_{\{N_{i1}\}}), T(N_{i2}, \mu_{\{N_{i2}\}})\big) & \text{if } N_i \in \mathbf{R}^E \\ N_i(\mu_{\{N_{i1}\}}, \mu_{\{N_{i2}\}}) & \text{if } N_i \in \mathbf{R}^M, \\ \mu_i & \text{if } N_i \in \mathbf{E} \end{cases}$$

where N_{i1} and N_{i2} are the input nodes of node N_i and $T(E, p)$ transforms the input element E to have pose p. A state is invalid if $\mu \neq \sigma(\mu)$. Some sets of relationship values might not have a valid state, if there is no set of element poses that fulfill the relationships. We relax this requirement by searching for states that minimize $\|\mu - \sigma(\mu)\|$.

Let τ be the initial unmodified state of the pattern and ν the modified state, including the edit $\mathbf{E}_\Delta$ proposed by the user, and the resulting change in value of the relationships $\mathbf{R}_\Delta$:

$$\nu = \tau + \sum_{\{i \mid E_i \in \mathbf{E}_\Delta\}} (0, \ldots, \Delta x_i, \Delta y_i, \Delta\theta_i, \Delta s_i, \ldots, 0) + \sum_{\{j \mid R_j \in \mathbf{R}_\Delta\}} (0, \ldots, \Delta r_j, \ldots, 0),$$

where $\mathbf{R}_\Delta$ are ancestors of $\mathbf{E}_\Delta$ in the pattern graph. The user edit is preserved if elements $\mathbf{E}_\Delta$ take on values $\nu_{\{\mathbf{E}_\Delta\}}$, therefore, we search for a state that minimizes $\|\mu_{\{\mathbf{E}_\Delta\}} - \nu_{\{\mathbf{E}_\Delta\}}\|$.

The constrained relationships $\mathbf{R}_o$ and $\mathbf{R}_c$ are met if they take on the values $\tau_{\{\mathbf{R}_o\}}$ and $\nu_{\{\mathbf{R}_c\}}$, respectively. This effectively splits the state into a free sub-state $\mu_{\text{free}} = \mu_{\{\mathbf{E}, \mathbf{R_u}\}}$ and a fixed substate $\mu_{\text{fixed}} = \mu_{\{\mathbf{R_o}, \mathbf{R_c}\}}$. Putting all the terms together gives us the

[1] Note that relationships may also use other properties of elements such as the element boundary as input, but these properties can only be modified indirectly through the element poses.

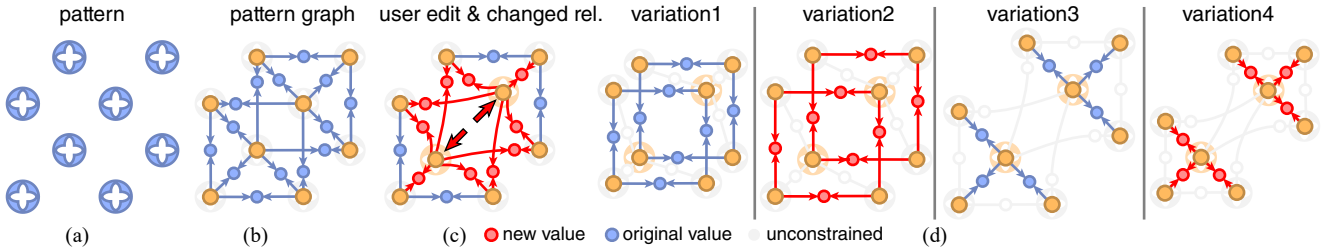


Figure 6: *Generating pattern variations. The structure of the pattern in (a) is captured by the pattern graph in (b). For clarity, each blue relationship represents one centroid distance and one centroid direction relationship. A user edit of the two central elements (c) causes some relationships change (red). Variations can be generated by fixing relationships to their new or original values (red and blue, respectively), or leaving them unconstrained (grey).*

following optimization problem to compute a pattern variation:

$$\begin{aligned} \min_{\mu} \quad & \|\alpha F_{\Delta}(\mu,\nu)\|^2 + \|\beta F_{\text{valid}}(\mu)\|^2 \\ \text{s.t.} \quad & \mu_{\{\mathbf{R}_o\}} = \tau_{\{\mathbf{R}_o\}} \text{ and } \mu_{\{\mathbf{R}_c\}} = \nu_{\{\mathbf{R}_c\}}, \end{aligned} \tag{1}$$

with

$$\begin{aligned} F_{\Delta}(\mu,\nu) &= \mu_{\{\mathbf{E}_{\Delta}\}} - \nu_{\{\mathbf{E}_{\Delta}\}} \\ F_{\text{valid}}(\mu) &= \mu - \sigma(\mu), \end{aligned}$$

where α and β are weights for the two terms, which we set to 1 and 10, respectively. In practice, we can fix the constrained relationship values and only optimize over the remaining state, resulting in an unconstrained non-linear optimization problem, which we solve using a 2D-subspace trust region method [Branch et al. 1999].

Regularization. While relationship values are fully determined by the element poses, the reverse is generally not true. For example, a rigid transformation of the pattern might not affect relationship values, if the relationships are invariant to these transformations. Additionally, variations that assign a large set of relationships to $\mathbf{R}_u$ are usually under-determined. For these cases, we formulate a global regularization based on two main principles:

First, *repeated structures should be preserved* as much as possible. These structures may describe symmetries or regularities in the pattern. Since repeated structures are described by relationship groups, we add a penalty for breaking groups. Second, *edits should remain local*, and modify the state as little as possible. Many moving parts make it hard to understand and predict the result of an edit. Based on these two principles, the regularization term is defined as:

$$\left(\|\gamma F_{\text{reg}}(\mu)\|^2 + \|\gamma F_{\text{local}}(\mu,\tau)\|^2\right)$$

$$\text{with } F_{\text{reg}}(\mu) = \begin{bmatrix} \mu_{\{G_1\}} - \widetilde{\mu}_{\{G_1\}} \\ \vdots \\ \mu_{\{G_h\}} - \widetilde{\mu}_{\{G_h\}} \end{bmatrix} \text{ and } F_{\text{local}}(\mu,\tau) = \mu - \tau,$$

where $\widetilde{\mu}_{\{G_i\}}$ denotes the average over all nodes in group G_i. We add this term to the objective in Eq. 1 with a weight of $\gamma = 0.01$.

5.2 Choosing Pattern Variations

For each edit to the pattern, there are 3^n possible variations, given n initial relationships. Of this large set, only a small percentage will yield useful element arrangements. This has three main causes: several variations result in unintuitive element arrangements; many variations do not have a valid element arrangement and large subsets of variations result in the same element arrangement. In this section, we present our approach to choosing a subset of variations that is valid, distinct, and intuitive.

Our approach combines several strategies in a filtering pipeline. The first step is to randomly sample the space of all possible variations with a sample set of fixed size (10000 samples in our experiments). This is followed by a heuristic *intuitiveness filter* that can be evaluated efficiently to filter out variations likely to result in unintuitive pattern variations. Next, we evaluate a random subset of the remaining variations (150 variations in our experiments – this number may be tuned to achieve optimal responsiveness) in a *linear approximation* of our optimization problem described in §5.1. This is the core enabling part of choosing among variations. The linear approximation allows us to approximate properties of a variations efficiently, without requiring an expensive non-linear solve. Based on the results of this approximation, we remove invalid variations and variations that are too similar using the *valid* and *distinct* filters. Finally, we order variations by distinctness, i.e., we make sure that the first few suggestions are as visually distinct as possible.

Intuitiveness. There is no clear definition of what an *intuitive* response to an edit is. However, through experimentation, we have identified one property that consistently causes unexpected variations. Variations that modify multiple relationship groups usually introduce too much change at once for the user to understand or predict the changes. We attempt to only retain variations where few groups change. Since the user can modify elements only, changes in meta-relationships are only possible by also changing their child relationships. We therefore consider changes of relationships or groups along a path in the pattern graph starting at the element nodes as a single change and filter out variations where relationship groups along more than one path have been changed.

This filtering is an optional step based on the observations we made during our experiments and could be replaced by any other filter. Our core strategy of linear approximation is independent of this step, as described below.

Linear approximation. First, we split the objective into simpler terms that are more amenable to linear approximation. Arranging these terms into a vector-valued objective gives us the following expression:

$$\mathbf{F}(\mu|\tau,\nu) = \left[\alpha F_{\Delta}(\mu,\nu), \beta F_{\text{valid}}(\mu), \gamma F_{\text{reg}}(\mu), \gamma F_{\text{local}}(\mu,\tau)\right]^{\mathrm{T}}$$

Note that taking the squared L^2-norm of this expression gives us the non-linear objective function. We linearize this objective with a two-term Taylor expansion at the initial state ν. As in the non-linear case, we split the state into μ_{free} and μ_{fixed} and only optimize over μ_{free} to avoid using constraints:

$$\begin{aligned} \mathbf{F}(\mu) = \mathbf{F}(\nu) &+ \mathbf{J}_{\text{free}}\left(\mu_{\text{free}} - \nu_{\text{free}}\right) + \mathbf{J}_{\text{fixed}}\left(\mu_{\text{fixed}} - \nu_{\text{fixed}}\right) \\ \text{with } \mathbf{J}_{\text{free}} &= \frac{\partial}{\partial \mu_{\text{free}}}\mathbf{F}(\nu) \text{ and } \mathbf{J}_{\text{fixed}} = \frac{\partial}{\partial \mu_{\text{fixed}}}\mathbf{F}(\nu). \end{aligned}$$

For clarity, we have omitted the two parameters τ and ν from $\mathbf{F}$ that remain constant during the optimization. Setting $\mathbf{F}(\mu)$ to zero

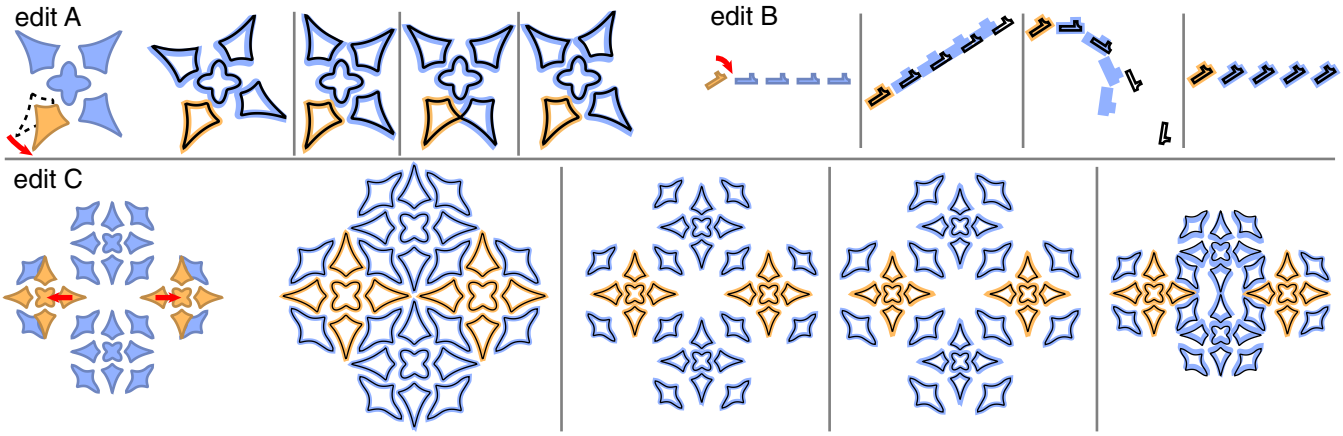


Figure 7: *Qualitative comparison of linear versus non-linear solutions of pattern variations. We show a few variations for three edits on different patterns. The linear solution is overlayed with black lines on top of the non-linear solution in blue and yellow. In most cases, the linear solution is close to the non-linear solution (edit A). The largest errors occur due to strong rotations around a distant pivot (edit B). The examples shown here are the largest errors we have observed. Usually the linear approximation is accurate even for more complex patterns (edit C).*

and re-arranging gives us the following linear system:

$$\mathbf{J}_{\text{free}}\,(\mu_{\text{free}} - \nu_{\text{free}}) = -\mathbf{F}(\nu) - \mathbf{J}_{\text{fixed}}\,(\mu_{\text{fixed}} - \nu_{\text{fixed}}).$$

The objective value at the initial state, $\mathbf{F}(\nu)$, is usually dominated by the distance to the user edit. Intuitively, we want to find a change of the *free* sub-state that cancels out the increase in the objective function introduced by the change of the *fixed* sub-state while also reducing the distance to the user edit. Since $\mathbf{J}$ only depends on the original state, we only need to compute it once. We find a minimum-norm solution of this system by computing the pseudo-inverse of $\mathbf{J}_{\text{free}}$ using Singular Value Decomposition. For reasonably small changes, the linear approximation is accurate enough to give good predictions for the properties of pattern variations. A qualitative comparison of linear vs. non-linear solutions for a few examples is shown in Figure 7. The largest errors we have observed occur due to the non-linear nature of strong rotations around a distant pivot, especially if these errors accumulate, as edit B of Figure 7, shows the largest errors we have observed when doing typical edits. Large errors usually occur due to the non-linear nature of strong rotations around a distant pivot. Usually the linear solution is accurate even for complex patterns, and even in the worst case we have observed, the user can get an impression of the element arrangement from the linear solution.

Validity and distinctness. For both of these filters we make use of the linear approximation. Solutions with high approximated objective value are likely to be invalid (i.e., no valid configuration of elements for the given fixed relationships exists) and can be removed. Distinct solutions are found by applying agglomerative hierarchical clustering to the solutions based on the L^2 distance of their relationship values. We use conservative threshold values and also rely on the final sorting to move elements that may be too similar to the end of the variations list. We preferred this option over having the threshold as a free parameter that may need to be tuned.

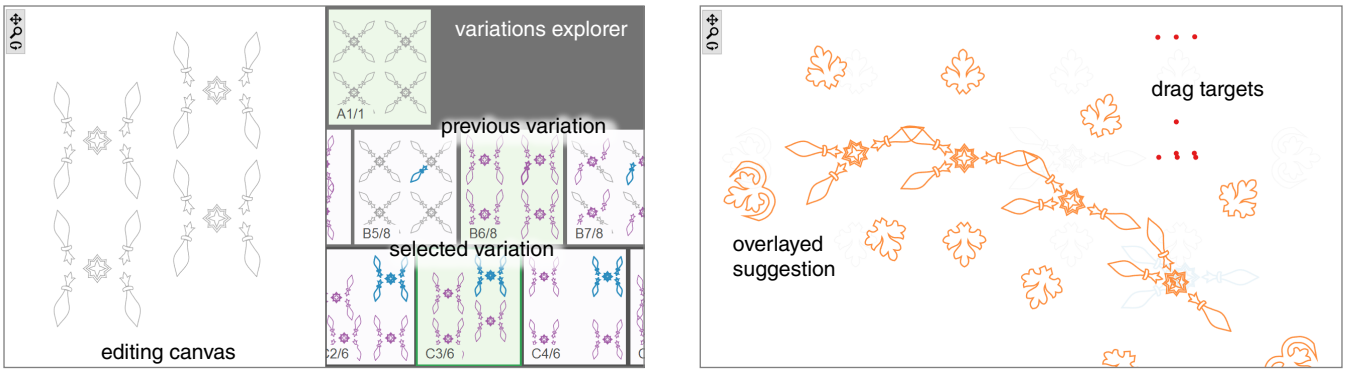
Assigning groups. As stated above, one of our goals is to preserve repeated structures in the pattern. When generating variations, we can assign relationship *groups* to the three sets $\mathbf{R}_u$, $\mathbf{R}_o$, and $\mathbf{R}_c$, instead of individual relationships. The indicator vector I then has one entry per group, not per relationship. This naturally preserves repeated structures and significantly reduces the number of possible variations. When two or more members of the same group have been changed to different values, we generate variations where the entire group is set to each of these distinct changed values. The indicator vector I for the group may then take on values up to $1 + N_c$, where N_c is the number of different values that members of the group have been changed to. Breaking repetitions might be desirable in some cases and variations that break repetitions can still be generated through groups that are assigned to $\mathbf{R}_u$.

6 Exposing Variations to the User

We present two pattern editing interfaces that expose suggested variations to the user. Please refer to the supplementary video to see each interface in action.

Exploration. To encourage creating and evaluating multiple alternatives, we developed an *Exploration* interface that allows users to quickly generate and examine many suggested variations as they directly edit a pattern. A screenshot is shown in Figure 8a. Our interface has two main components. On the left is an editing canvas that provides standard direct manipulation tools. On the right is the variations explorer, which shows a dynamically updated set of suggested variations. Every time the user edits an element in the canvas, the explorer generates a new set of variations as described in §5 and arranges the variations in a row, sorted from most to least intuitive. The user can choose a variation and continue editing in the canvas to receive new rows of suggestions. By going back to previous rows, users can revisit previous decisions and choose a different suggestion to explore different design directions. To keep the interface responsive, the explorer displays the linear approximations of variations, as described in §5.2. In most cases, we found these approximate variations provided a reaonable preview of the suggested edit. Once the user accepts a variation, we solve the corresponding full non-linear optimization to update the canvas.

Auto-complete. While the Explorer interface facilitates browsing and navigation of different variations, one disadvantage is that the editing canvas gets a limited amount of screen real estate, which can hinder tasks that require detailed element manipulation. To address such scenarios, we developed an *Auto-complete* interface that allows users to preview and accept suggested variations directly on the canvas as part of their direct manipulation interactions. After the user edits an element, the system automatically generates pattern variations based on this 'reference' edit. When the user starts manipulating the next element, we compute the position of the element in each of the variations and show them as dots on the canvas. We call these dots Auto-complete *targets*. Dragging the element close to the targets previews the variations as overlays on the canvas without interrupting the workflow. Dropping the element on a



(a) Exploration (b) Auto-complete

Figure 8: *Two editing interfaces expose pattern variations to the user. In the exploration interface (a), the user can preview several suggestions for each edit in the* variations explorer *(grey side-panel). In the Auto-complete interface (b), the user can choose suggestions by snapping a dragged element to one of several drag targets shown as red points. Each such target reveals a different variation.*

target accepts the suggestion and the elements are updated accordingly. Subsequent edits generate new sets of variations. This interface is similar in spirit to existing snap dragging functionality like the Smart Guides in Adobe Illustrator. An important aspect of such tools is that they allow the user to ignore or override the suggested edits if they do not match the desired result. In such scenarios, temporarily turning off all Auto-complete dots avoids accidentally accepting a suggestion.

7 Extensions

Our core method can generate a set of valid, distinct, and intuitive variations of a pattern given a user edit. The extensions described below enable application of our method to specific user interaction approaches and provide techniques to get large performance improvements on some common types of patterns, allowing application of our method to larger patterns.

Pattern hierarchy. Complex patterns often exhibit a hierarchical structure, where sets of simple elements are arranged to form composite elements. We say that elements are children of the composite element they are forming. In these patterns only relationships between children that are in the same composite element are relevant, or between elements that are not part of any composite element, significantly reducing the number of possible relationships. We exploit this structure by modeling this hierarchy in our pattern graph and only allowing relationships between children of the same parent. The hierarchy acts similar to a scene graph for our pattern: moving a parent element also moves all child elements.

Instance groups. Many of the more regular pattern make heavy use of instancing: the same composite element is cloned multiple times across the pattern. We model instancing as groups of composite elements. After the optimization, modifications to the children of an instanced element are propagated to all other members of the instance group. This allows us to update large patterns containing many instances without having to optimize over all elements.

8 Results

To evaluate our method, we performed several quantitative and qualitative experiments, and performed a user study. We start by showing results of the qualitative evaluations, followed by quantitative evaluations, and the user study.

Recall the proposed re-targeting of a pattern shown in Figure 2, where the goal is to decrease the visual weight of a pattern. Manually editing the pattern (making good use of selections and element movement) requires 24 edit operations. In Figure 9, we show the same re-targeting performed with our method. After each edit, variations can be selected using one of our interfaces. Our method only requires two edits and two variation selections to reach the desired re-target. For the operations performed to reach some of the other results shown in this section, we refer to the accompanying video.

One key feature of our method is the ability to generate useful pattern variations for a wide range of edits. In Figure 10, we show the first 5–6 variations generated by our method for different edits, which involve rotating, scaling, or moving elements in the pattern. One of our goals is to provide inspiration to the designer working with our method. The variations found by our method are distinct and intuitive, but additionally, they may not be immediately obvious from looking at the original pattern, which may bring variations to the attention of the designer she did not think of before.

Figure 11 shows the results of several editing sessions we performed on seven different patterns. The original patterns are shown in blue and the results after editing in orange. For some of the patterns, we show the results of several different editing sessions. Note how we can create several distinct patterns using few operations compared to traditional manual editing, where elements would need to be translated, rotated, and scaled individually. These types of edits may be useful in several real-world scenarios; for example, to efficiently re-arrange larger structures (Pattern 1), make a pattern less or more visually dense (Pattern 2), reduce a pattern's regularity (Pattern 3), giving the pattern a more interesting global shape (Pattern 4), re-targeting a pattern to remove an axis-symmetry (Pattern 5), re-arranging elements to fill in holes that may have resulted from removing other elements (Pattern 6), or re-size some elements while retaining the pattern's structure (Pattern 7).

Pattern graph construction. All pattern graphs used in this paper are created manually by selecting pairs of elements or relationships and choosing a set of connecting relationships from the list in Appendix A. Similarly, the pattern hierarchy and repeated structures are defined manually by selecting elements that form composite elements and by marking groups of repeated composite elements. The center, orientation and size of elements are computed automatically using the polygon centroid, PCA, and bounding sphere, respectively, and are adjusted manually only if needed (e.g., the orientation of circles). Correspondences between elements of repeated structures are found automatically by first aligning their parent composite elements using the known centroid, orientation and size, and then assigning elements in this aligned coordinate frame based on their centroid distances. For very complex patterns with

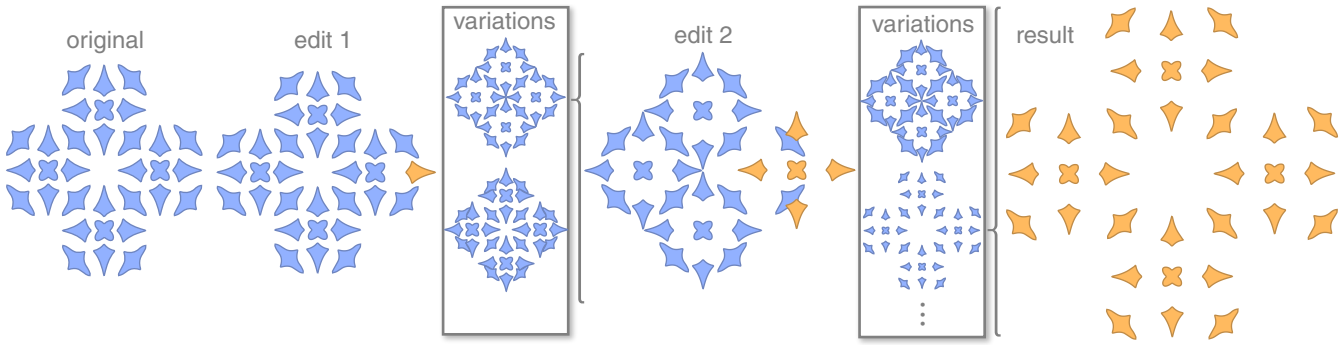


Figure 9: *Operations required to perform the re-targeting described in Figure 2. To decrease the visual weight of the pattern shown on the left, we perform two edits, each followed by the selection of a suggested variation. Manually re-targeting the patten would require 24 operations.*

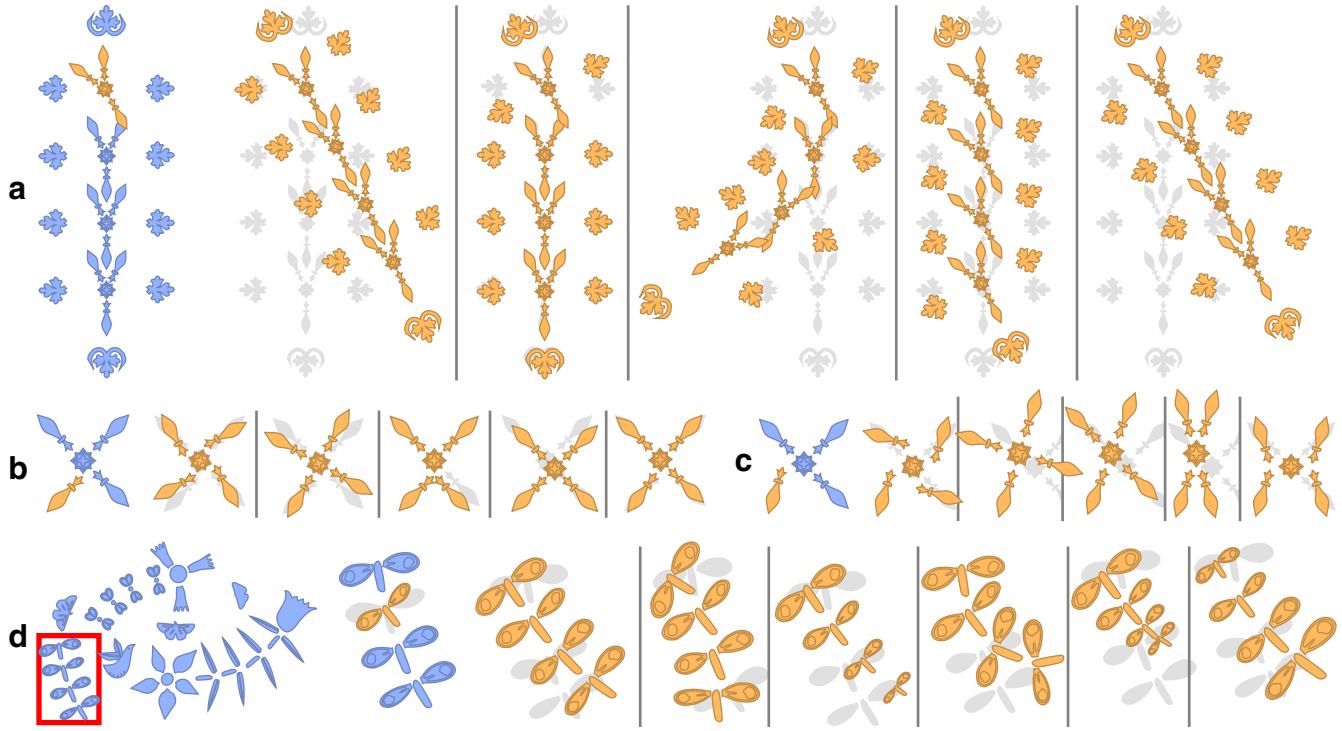


Figure 10: *The first 5–6 variations generated for each of the four pattern edits shown on the left. Note how the variations are distinct and intuitive for the given edit, but could also give inspiration to the designer, since coming up with all these variations and previewing them without assistance may not be trivial.*

repeated structures, we can save time by defining relationships in one of the structures only and propagating to the repetitions. As discussed above, the number of relationships that need to be defined typically depends on the branching factor of the hierarchy, as opposed to the number of element pairs. Creating the full pattern graph typically takes under 5 minutes of manual work for most of the pattern graphs, slightly longer for Pattern 6 shown in Figure 11. Pattern graphs used in the results, including the pattern hierarchy, are shown in the supplementary material. For real editing workflows, automatic or assisted pattern graph extraction would be more suitable; a few ideas are discussed in §9. One consideration when constructing pattern graphs is that additional relationships often result in additional valid and distinct variations. If a designer wants to focus on a specific subset of variations, e.g., variations that do not alter element scale, adding size-based relationships would unnecessarily clutter the list of suggestions. In this case, improving variation ranking, for example, by adding a term that weighs relationship types, would improve exploration.

8.1 Quantitative Experiments

In this section we provide quantitative evaluation of our method. Our evaluation is based on 8 typical edits, shown in Figure 12. The table in the same figure gives several statistics for each edit. The first two rows show the number of pattern graph nodes in each of the scenes, as well as the number of nodes in the composite element that was changed by the edit (see §7 for details). The average time for a linear and a non-linear solution are shown below. Rows 5–8 of the table show the number of generated variations, and how many variations remain after each filtering step. The random selection discussed in §5.2 is split into two parts, the first random selection is performed before the filter that removes unintuitive variations, since this filter does not require the linear approximation, the second random selection is done before computing the linear approximation, which is needed to determine if variations are valid and distinct. On average. The 'intuitive'-filter retains 56% of the variations, the 'valid'-filter retains 66%, and the 'distinct'-filter retains 13%.

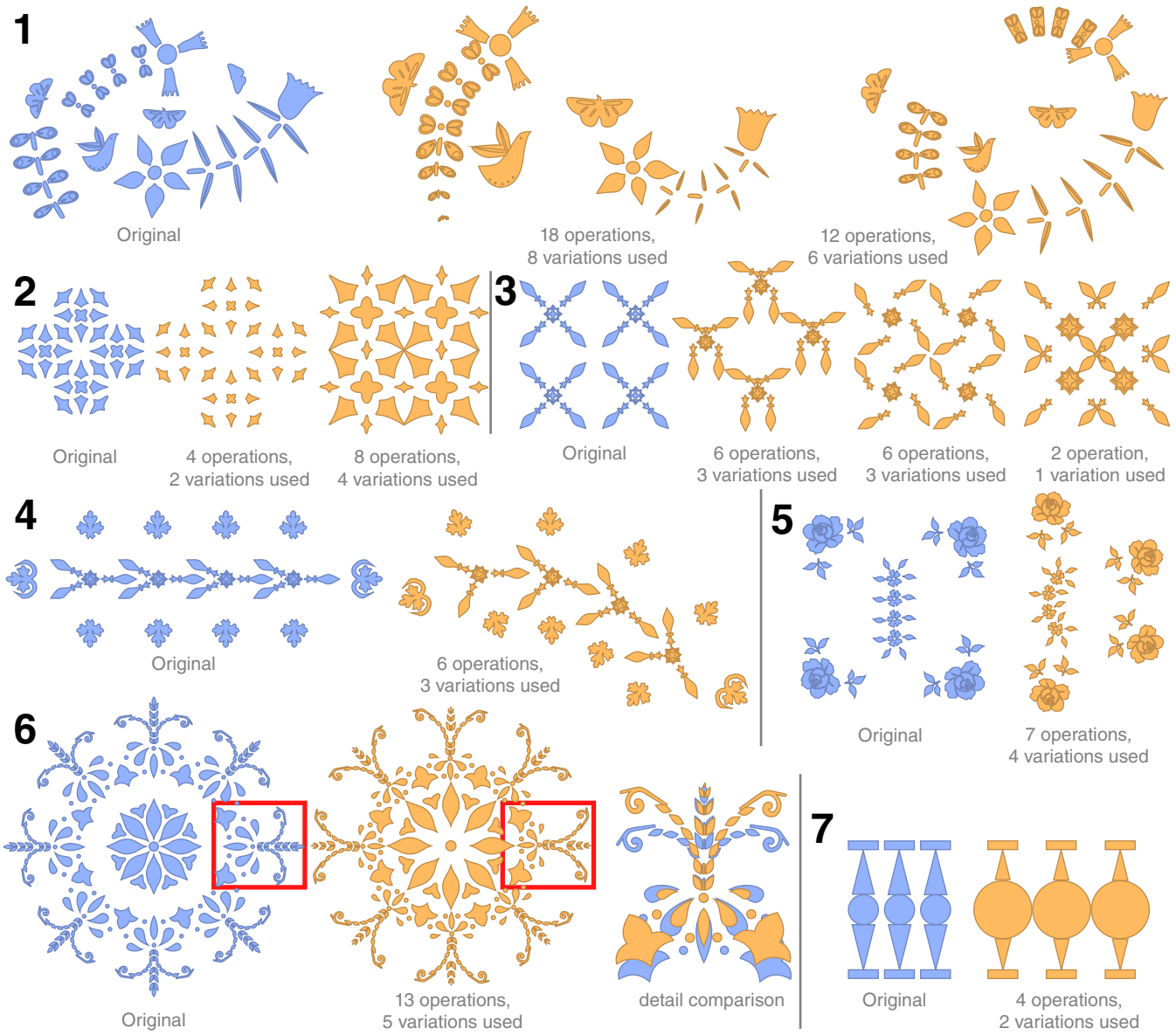


Figure 11: *Results of several pattern editing sessions performed with* PATEX. *The number of operations (including choosing suggestions) and the number of chosen suggestions are shown below each result pattern. Note how new, often visually quite distinct patterns can be created from the original in just a few operations. Manual editing would require a much larger number of operations.*

The sensitivity of our method to the number of linearly approximated variations is shown in Figure 13. The topmost plot relates the total number of distinct, valid and intuitive variations found by our method to the number of linear approximations computed. Numbers show a clear tendency for convergence to a stable number of variations at around 2000 linear approximations. On average, we get diminished returns after approximately 150 linear evaluations, which is the value we used for our user study and our experiments.

Scalability. Two main factors contribute to the scalability of our approach. First, we sample the space of possible variations before filtering. Thus, even for patterns with many possible variations, the computation time is bounded by how many samples we generate. The relation between sampling density and the number of missed valid and distinct variations is shown in Figure 13. Note that the set of variations need not be complete; in our experience, showing 5-10 different suggestions is often enough to facilitate exploration. Additionally, using groups versus relationships in the assignments (§5) means the variations grow in the number of groups, not relationships. Second, the pattern hierarchy bounds the complexity of each optimization, as only the relationships and elements that are children of the same composite element need to be considered in practice. Most patterns can be grouped hierarchically and in such patterns, the optimization can be split up into several smaller independent optimizations, one per composite element that was directly modified by the user. For each of these optimizations, the number of relevant relationships and elements only depends on the branching factor of the hierarchy (which remains relatively constant as the size of the pattern increases) rather than all possible relationships between individual elements.

8.2 User Evaluation

To evaluate the utility of our approach and gain insight about how designers might use automated suggestions in the context of con-

	edit							
	A	B	C	D	E	F	G	H
# nodes	111	160	204	69	160	595	111	131
# comp. nodes	59	34	108	69	34	65	59	16
linear time (ms)	34	20	39	32	20	38	34	12
non-lin. time (s)	4.28	0.92	1.17	2.60	1.60	6.56	1.26	0.497
# variations	~74k	~83k	~37k	~9k	~249k	~9k	216	36
	pick 10000 random variations							
intuitive filter	6712	5524	666	6144	2287	8192	112	32
	pick 150 random variations							
valid filter	145	86	97	93	89	150	104	28
distinct filter	17	8	9	6	6	6	7	13

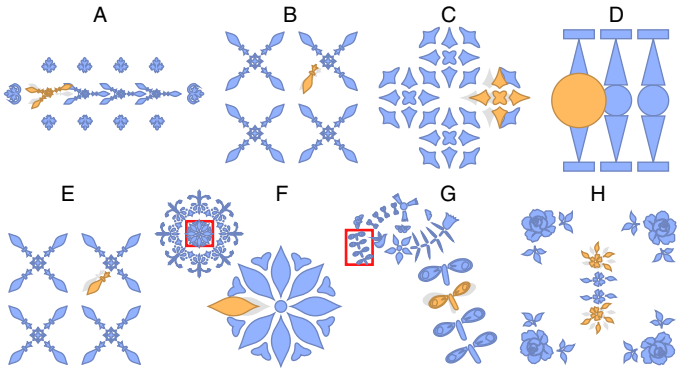


Figure 12: *Our quantitative experiments are based on the eight typical edits shown on the right. Modified elements are shown in orange. In the table on the left, we show several statistics for each edit that exemplify the time required for generating variations, as well as the effect of filters on the final set of variations.*

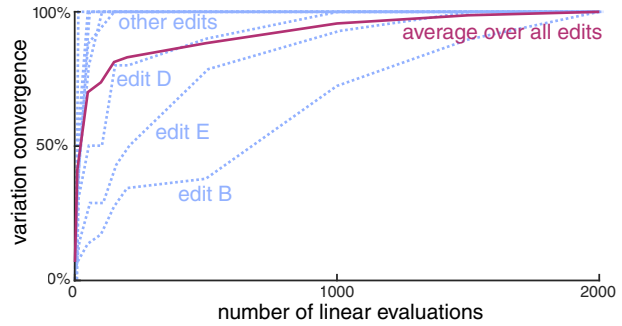


Figure 13: *Sensitivity of our method to the number of linear approximations. We show the normalized number of distinct variations found for each edit versus the the number of linear evaluations. Note that there is a clear tendency for convergence at around* 2000 *evaluations. On average, there are diminished returns above* 150 *variations.*

crete design tasks, we conducted an exploratory user study with our two proposed editing interfaces. We recruited six participants with moderate to extensive experience using existing drawing software who create vector graphics (in the form of graphic designs, infographics, layouts) at least once a month as part of their profession/studies. One participant designed textile patterns in a previous job. We asked each participant to perform two design tasks with one interface and then two different tasks with the other interface. We kept the sequence of tasks constant but randomized the order in which the interfaces were presented. Figure 14 shows the input patterns and prompts for all tasks. We encouraged participants to think aloud as they worked, and after the tasks, we asked them additional questions about their experience. Our questions focused on the usefulness of the suggestions and things that worked well or not in the two interfaces. A demo of our method with both interfaces is available on the project website [pat 2016].

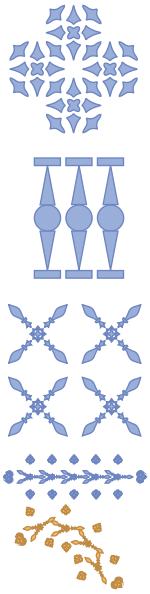
Findings

Usefulness of suggestions. In general, participants described the suggestions as useful, and there was consensus that automatically generated variations would be helpful in the context of their current tools and workflows. That said, the perceived level of usefulness seemed somewhat task-dependent. For tasks 1 and 2, participants tried to preserve much of the regular structure in the input pattern, and the suggested variations seemed to retain many of the expected relationships between elements. For example, a common strategy for task 1 was to either scale down an element or move it farther away from nearby elements. Such edits tended to produce variations that propagated the smaller scale and increased inter-element distance to other parts of the pattern. However, for task 3 (reducing the regularity of the input pattern), participants often found the suggested variations to be a bit too regular and, as one participant remarked, 'computer-like'. In task 4, linearization artifacts (discussed in more detail below) sometimes produced unexpected results that lowered the perceived usefulness of the suggestions for this task.

Behavior. The way in which participants used the two interfaces was consistent with their comments about the usefulness of the suggestions. With the Auto-complete interface, participants almost always previewed suggestions when they were available, and for most tasks, they accepted one or more variations during the editing session. With the Exploration interface, participants generally spent more time browsing (and accepting) suggestions in the explorer view than they did editing elements on the canvas. With both interfaces, the first few sets of suggested variations seemed to have a considerable impact on how participants interacted with suggestions for the rest of the task. When the initial suggestions were deemed expected or useful, participants expressed their pleasure and seemed more willing to generate and browse through subsequent suggestions. Otherwise, participants often focused more on manual edits. One participant explicitly expressed apprehension about using the Auto-complete targets after she previewed some surprising suggestions earlier in the task.

Preferences. Across participants, there was no clear preference between the two interfaces. Some liked that the Auto-complete suggestions were displayed on-canvas and accessible during direct manipulation of the elements. In addition, one participant pointed out that mousing back and forth over Auto-complete targets was a convenient way to see the differences between the current pattern and the suggested variation. Others preferred the Explorer interface because it facilitated browsing and previewing many alternatives. Three participants noted that they liked the ability to go back through their history of accepted suggestions to see how their edits evolved. Overall, participants felt that one key strength of both interfaces was the ability to quickly preview and evaluate many alternative variations. For example, the former textile designer described her experience with the Auto-complete interface as similar to playing with different arrangements of physical paper cutouts, which was one of her methods for seeking inspiration early in the desing process. Another participant felt that the variations explorer was more inspring because it gave her time to contemplate many alternative designs while the Auto-complete interface forced her to make choices immediately.

Usability. A few usability issues arose during the study. In some cases, the difference between the variation previews and the actual refined variation used to update the canvas was noticeable. As



Task 1: This pattern is to be placed on a coloured background for a poster design. Increase the spacing so that more of the background shows through while retaining the structure of the pattern.

Task 2: This is a company's trademark pattern. In an advert, the company wants to put content into the three central circles. Make the circles 1.5x bigger (relative to the rest of the pattern) while keeping the pattern recognizable.

Task 3: This is a popular gift-wrapping pattern, but market research suggests that customers prefer less regular patterns. Make the design slightly less regular while preserving its basic structure.

Task 4: Take the input pattern (top) and make it look like the target pattern (bottom). You do not have to reproduce the target exactly, but try to match the overall characteristics of the pattern.

Figure 14: *Input patterns and prompts for user study tasks.*

mentioned in §5.2, this is mainly due to large rotations around a distant pivot point. In such cases, participants expressed surprise and typically displeasure at the discrepancy. In addition, a few participants commented on the responsiveness of the interfaces. The latency of our current implementation makes multiple edits to the same element a bit cumbersome, since there is a brief pause after each edit as the system generates suggested variations. There were also situations where participants had trouble distinguishing between similar variations, especially in the Exploration interface. Finally, some participants mentioned that the lack of standard features like multi-select and grouping operations made some of the tasks less convenient than they might otherwise have been.

Design Implications

Our findings suggest several potential ways to improve the design of our system.

Unobtrusive suggestions. The fact that our suggestions are more useful for some tasks and less so for others highlights the importance of presenting automated variations in an unobtrusive way. In cases where the suggestions are less helpful (e.g., tasks that mainly involve breaking rather than preserving relationships), the interface should not encumber standard editing operations like freeform manipulation of elements. To this end, one obvious opportunity for improvement is to prevent the interface from freezing the editing canvas while generating variations by executing the computations in a separate thread.

Explaining variations. The impact of the initial suggestions on subsequent editing behavior implies that participants quickly formed some mental model for how variations were being generated. In some cases, this model ("It's only giving me suggestions where elements are in a straight line") prevented the user from considering potentially useful variations generated later on in the session. In part, this problem could diminish as users gain familiarity with system and see a larger collection of automatic variations. However, it may also be worth considering visualization techniques that help explain what relationships are being preserved, changed, or ignored in each variation.

Better filtering. The confusion around suggestions that are very similar to each other indicates that our filtering of variations could be better. In particular, the threshold that we use for identifying 'identical' variations could be tuned. Moreover, it may be worth giving users interactive control over how many variations are displayed. With this strategy, the default set of variations could be restricted to very distinct alternatives, and if the user asks for more, the subsequent suggestions could vary in more subtle ways.

Quality of linear approximation. There are a few ways to prevent large, misleading discrepancies between variation previews generated from poor linear approxiations and the final optimized pattern. Putting variation computations on a separate thread, as discussed above, would allow us to incrementally update linear approximation previews shown in the explorer view so that they gradually become more accurate. We could also modify our pattern variation computation to produce better (but more expensive) approximations in cases where the linear approximation is unlikely to be sufficient.

8.3 Limitations

The main focus of our paper is exploring variations, therefore we opted to generate our pattern graphs manually. Nevertheless, as we will discuss in §9, our method would be well-suited for automatic pattern generation approaches.

Currently, we do not support generating variations that change the number of elements. Adding or removing elements might be useful in some scenarios; for example, if a variation would result in overlapping elements. However, as we have shown in §8, our current set of operations is already sufficient to create interesting pattern variations and we would like to leave this feature to future work.

As shown in Figure 7, the linear approximation may be inaccurate in some cases, mainly due to strong rotations around a distant pivot. Although we encountered this problem only rarely, one possible option we would like to explore in future work is moving the computations to a separate thread that updates the currently shown linear approximations with the non-linear solver in the background.

9 Conclusions and Future Work

In this paper we try to answer an interesting question: *how can we explore variations of the complex structures that can be found in patterns?* Current options are to either to employ a fixed parameterization, thereby limiting the exploration to a narrow space, or to re-arrange elements manually, which is infeasible for more complex patterns and puts all the creative effort on the shoulders of the designer. We introduce PATEX, a new technique for representing and generating pattern variations that preserve relationships between elements while respecting user-specified edits. We incorporate this functionality in two pattern editing interfaces that expose automated variations to users in the form of suggested edits.

The feedback from our user study along with our own experiences using the editing interfaces suggests that automatic variations are useful for a variety of pattern editing scenarios. In particular, our variations help users explore the design space by showing how different interpretations of an edit can yield different results. In addition, for many tasks, automatic variations can reduce friction in the editing process by enabling users to execute global edits without manually editing many individual elements. Our study also reinforces the fundamental problem with a single, fixed representation of a pattern. The fact that different participants wanted to preserve or break different sets of relationships, even for the same pattern, indicates that there is almost never a single 'correct' set of relationships that characterize a pattern. This in turn highlights the impor-

tance of presenting multiple variations that correspond to different interpretations.

In addition to the specific design improvements discussed at the end of §8.2, we see several interesting directions for future work:

Higher-order relationships. Currently, all our relations take exactly two inputs. While this is allows us to describe a wide range of structures, some structures like axial symmetries could be modeled more efficiently using relationships with more than two inputs. In future work we plan to explore explicit representations for ternary and higher-order relationships.

Automatic extraction of pattern graphs. As mentioned earlier, our pattern graphs are specified manually. However, our method seems to be well-suited to support automatically extracted patterns. Since we assume overcomplete input graphs, we can accumulate automatically detected relationships without worrying about conflicting or redundant constraints. Extraction would require finding groups of repeated or symmetric structures to determine which elements to connect with relationships. We could adapt several existing techniques, including symmetry detection [Yeh and Mech 2009], inverse parametric modeling [Šťava et al. 2010; Wang et al. 2011], and edit history analysis in the spirit of Denning [2011] or Doboš et al. [2014].

Data-driven evaluation of variations. In our work, we propose simple heuristics for filtering and sorting variations in an effort to identify intuitive and distinct sets of suggestions to present to users. Future work could explore data-driven techniques for evaluating variations based on human ratings of candidate variations.

Acknowledgements

We thank the anonymous reviewers for their comments and constructive suggestions, and the participants of our user study. This work was supported in part by the ERC Starting Grant SmartGeometry (StG-2013-335373), Marie Curie CIG 303541, and the Open3D Project (EPSRC Grant EP/M013685/1).

References

ALHASHIM, I., LI, H., XU, K., CAO, J., MA, R., AND ZHANG, H. 2014. Topology-varying 3d shape creation via structural blending. *ACM TOG 33*, 4 (July), 158:1–158:10.

BAUDISCH, P., CUTRELL, E., HINCKLEY, K., AND EVERSOLE, A. 2005. Snap-and-go: Helping users align objects without the modality of traditional snapping. In *CHI*, 301–310.

BERNSTEIN, G. L., AND LI, W. 2015. Lillicon: Using transient widgets to create scale variations of icons. *ACM TOG 34*, 4 (July), 144:1–144:11.

BIER, E. A., AND STONE, M. C. 1986. Snap-dragging. In *ACM SIGGRAPH*, 233–240.

BOKELOH, M., WAND, M., SEIDEL, H.-P., AND KOLTUN, V. 2012. An algebraic model for parameterized shape editing. *ACM SIGGRAPH 31*, 4, 78:1–78:10.

BRANCH, M. A., COLEMAN, T. F., AND LI, Y. 1999. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J. Sci. Comput. 21*, 1 (Aug.), 1–23.

CHEN, X., AND HOFFMANN, C. M. 1995. Design compilation of feature-based and constraint-based cad. In *Solid Modeling and Applications*, 13–19.

DANIEL, M., AND LUCAS, M. 1997. Towards declarative geometric modelling in mechanics. In *Integrated Design and Manufacturing in Mechanical Engineering*, P. Chedmail, J.-C. Bocquet, and D. Dornfeld, Eds. Springer Netherlands, 427–436.

DENNING, J. D., KERR, W. B., AND PELLACINI, F. 2011. Meshflow: Interactive visualization of mesh construction sequences. *ACM TOG 30*, 4 (July), 66:1–66:8.

DOBOŠ, J., MITRA, N. J., AND STEED, A. 2014. 3d timeline: Reverse engineering of a part-based provenance from consecutive 3d models. *CGF 33*, 2 (May), 135–144.

FISH, N., AVERKIOU, M., VAN KAICK, O., SORKINE-HORNUNG, O., COHEN-OR, D., AND MITRA, N. J. 2014. Meta-representation of shape families. *ACM TOG 33*, 4 (July), 34:1–34:11.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iwires: An analyze-and-edit approach to shape manipulation. *ACM SIGGRAPH 28*, 3, #33, 1–10.

GLEICHER, M., AND WITKIN, A. 1991. Differential manipulation. In *Graphics Interface*, 61–67.

GLEICHER, M. 1992. Briar: A constraint-based drawing program. In *CHI*, 661–662.

GUERRERO, P., JESCHKE, S., WIMMER, M., AND WONKA, P. 2014. Edit propagation using geometric relationship functions. *ACM TOG 33*, 2 (Apr.), 15:1–15:15.

HARADA, M., WITKIN, A., AND BARAFF, D. 1995. Interactive physically-based manipulation of discrete/continuous models. In *ACM SIGGRAPH*, ACM, New York, NY, USA, 199–208.

HUANG, Q.-X., MECH, R., AND CARR, N. 2009. Optimizing structure preserving embedded deformation for resizing images and vector art. *CGF 28*, 7, 1887–1896.

IGARASHI, T., MATSUOKA, S., KAWACHIYA, S., AND TANAKA, H. 1997. Interactive beautification: A technique for rapid geometric design. In *UIST*, 105–114.

JACOBS, C., LI, W., SCHRIER, E., BARGERON, D., AND SALESIN, D. 2003. Adaptive grid-based document layout. *ACM TOG 22*, 3 (July), 838–847.

NAN, L., SHARF, A., XIE, K., WONG, T.-T., DEUSSEN, O., COHEN-OR, D., AND CHEN, B. 2011. Conjoining gestalt rules for abstraction of architectural drawings. *ACM SIGGRAPH 30*, 6. 185:1-185:10.

NELSON, G. 1985. Juno, a constraint-based graphics system. In *ACM SIGGRAPH*, 235–243.

O'DONOVAN, P., AGARWALA, A., AND HERTZMANN, A. 2014. Learning Layouts for Single-Page Graphic Designs. *IEEE TVCG 20*, 8, 1200–1213.

2016. PATEX project website. http://geometry.cs.ucl.ac.uk/projects/2016/pattern-variations/. Accessed: 2016-19-04.

PAVLIDIS, T., AND VAN WYK, C. J. 1985. An automatic beautifier for drawings and illustrations. *ACM SIGGRAPH 19*, 3, 225–234.

REINERT, B., RITSCHEL, T., AND SEIDEL, H.-P. 2013. Interactive by-example design of artistic packing layouts. *ACM SIGGRAPH Asia 31*, 6.

RYALL, K., MARKS, J., AND SHIEBER, S. 1997. An interactive constraint-based system for drawing graphs. In *UIST*, 97–104.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory modeling with collaborative design spaces. *ACM SIGGRAPH Asia 28*, 5, 167:1–167:10.

TSANDILAS, T., GRAMMATIKOU, M., AND HUOT, S. 2015. Bricosketch: Mixing paper and computer drawing tools in professional illustration. In *Proc. International Conference on Interactive Tabletops & Surfaces*, 127–136.

ŠT'AVA, O., BENE, B., MCH, R., ALIAGA, D. G., AND KRITOF, P. 2010. Inverse procedural modeling by automatic generation of l-systems. *CGF 29*, 2, 665–674.

WANG, Y., XU, K., LI, J., ZHANG, H., SHAMIR, A., LIU, L., CHENG, Z., AND XIONG, Y. 2011. Symmetry hierarchy of man-made objects. *CGF 30*, 2, 287–296.

XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. 2009. Joint-aware manipulation of deformable models. In *ACM SIGGRAPH*, 35:1–35:9.

XU, P., FU, H., IGARASHI, T., AND TAI, C.-L. 2014. Global beautification of layouts with interactive ambiguity resolution. In *UIST*.

XU, P., FU, H., TAI, C.-L., AND IGARASHI, T. 2015. Gaca: Group-aware command-based arrangement of graphic elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '15, 2787–2795.

YEH, Y.-T., AND MECH, R. 2009. Detecting symmetries and curvilinear arrangements in vector art. *CGF 28*, 2, 707–716.

YEH, Y.-T., YANG, L., WATSON, M., GOODMAN, N. D., AND HANRAHAN, P. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM SIGGRAPH 31*, 4, 56:1–56:11.

YEH, Y.-T., BREEDEN, K., YANG, L., FISHER, M., AND HANRAHAN, P. 2013. Synthesis of tiled patterns using factor graphs. *ACM TOG 32*, 1, 3:1–3:13.

YVARS, P.-A. 2008. Using constraint satisfaction for designing mechanical systems. *International Journal on Interactive Design and Manufacturing (IJIDeM) 2*, 3, 161–167.

ZHENG, Y., CHEN, X., CHENG, M.-M., ZHOU, K., HU, S.-M., AND MITRA, N. J. 2012. Interactive images: Cuboid proxies for smart image manipulation. *ACM SIGGRAPH 31*, 4, 99:1–99:11.

A Geometric Relationship Types

In our experiments, we use seven relationship types based on element poses that are defined similar to Guerrero et al. [2014], and three meta-relationship types that relate relationship values. This list can be extended as needed and we plan to include relationships based on additional properties of elements, like a distance between element boundaries, in future work.

We use the following element relationships:

- The *distance* relationships is defined as the ℓ^2 distance between element centroids.
- The *orientation difference* relationship is the angular difference between element orientations in $[-\pi, \pi)$.
- The *size difference* relationship is the difference between element bounding sphere radii.
- The *absolute direction* relationship is the angle the direction from the centroid of element E_1 to the centroid of element E_2 makes with the global x-axis. Angles are in $[-\pi, \pi)$, clockwise rotations are negative angles.
- The *relative direction* relationship is the angle the direction from the centroid of element E_1 to the centroid of element E_2 makes with the orientation of E_1. Angles are in $[-\pi, \pi)$, clockwise rotations are negative angles.
- The *symmetric direction* relationship is the angle the direction from the centroid of element E_1 to the centroid of element E_2 makes with the orientation of E_1. Angles are in $[0, \pi)$, both clockwise and counter-clockwise rotations are positive angles.
- The *relative distance* relationship is the distance between the centroids of elements E_1 and E_2, relative to the size of E_1.

Our three meta-relationships are defined as follows:

- The *relationship difference* is the difference between two relationship values $R_1 - R_2$.
- The *relationship angular difference* is the difference between two angular values constrained to $[0, \pi)$.
- The *relationship ratio* is R_1 divided by R_2.