Control Mechanisms for the Procedural Generation of Visual Pattern Designs

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Lena Edith Elfriede Gieseke

aus Georgsmarienhütte

Hauptberichter: Prof. Dr.-Ing. Martin Fuchs
Mitberichter: Prof. Dr. Jens-Uwe Hahn
Prof. Dr. Thomas Ertl

Tag der mündlichen Prüfung: 05. November 2018

Institut für Visualisierung und Interaktive Systeme
der Universität Stuttgart

2018
# Contents

Abstract ix  
German Abstract — Zusammenfassung xi  

1 Introduction 1  
1.1 Problem Statement 2  
1.2 Research Questions 2  
1.3 Contributions 3  

2 Analysis Framework for Creative Control 6  
2.1 Terminology 7  
2.2 Taxonomy of Control Mechanisms 8  
  2.2.1 Control Paradigms 9  
  2.2.2 Control Mechanisms 12  
2.3 Creative Means 14  
  2.3.1 Navigation 18  
  2.3.2 Transparency 19  
  2.3.3 Variation 19  
  2.3.4 Stimulation 20  
2.4 Conclusion 21  

3 State of the Art of Creative Control for Procedural Ornamentation 22  
3.1 Terminology 23  
3.2 Design Goals 24  
3.3 Models 28  
  3.3.1 Procedural 28  
  3.3.2 Data-Driven 31  
  3.3.3 Ornamental 32  
3.4 Analysis of the State of the Art 33  
  3.4.1 Texturing Methods 34  
  3.4.2 Shapes and Masks 41  
  3.4.3 Vector Fields 44  
  3.4.4 Curves, Sketches and Painting 46  
  3.4.5 Element Placement 50  
3.5 Discussion 51  
3.6 Conclusion 54  

4 Goal-Oriented Control for Interactive Parameter Retrieval 56  
4.1 Related Work 58  
4.2 Texture 60  
4.3 Texture Descriptors 60  
  4.3.1 Fourier Descriptor 61  
  4.3.2 Laplace Pyramid Descriptor 62
# Contents

6 Explorative Control for Artistic Expression 124
   6.1 Related Work .......................................... 127
   6.2 Pipeline ................................................... 128
      6.2.1 Segmentation ........................................ 129
      6.2.2 Reconfiguration of Image Constituents ............... 129
   6.3 Parameter Space Exploration ............................. 132
   6.4 User Study ............................................... 135
      6.4.1 Results and Analysis ................................ 136
   6.5 Discussion and Outlook ................................ 137
   6.6 Conclusion ............................................... 138

7 Conclusion and Outlook 140
   7.1 Summary of Contributions ................................. 140
   7.2 Generalizability .......................................... 142
   7.3 Outlook ..................................................... 143
   7.4 Closing Remarks .......................................... 144

Image References 145

Image Sources 149

Bibliography 159
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Historic ornamentation examples</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Ornamentation principles</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Islamic and Celtic ornamental designs</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Gabor filter bank</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Survey results on a natural images database</td>
<td>69</td>
</tr>
<tr>
<td>4.3</td>
<td>Preliminary test textures</td>
<td>74</td>
</tr>
<tr>
<td>4.4</td>
<td>Optimization performance comparison</td>
<td>75</td>
</tr>
<tr>
<td>4.5</td>
<td>The best matching procedural texture images</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>Convergence behavior of the optimizers</td>
<td>78</td>
</tr>
<tr>
<td>4.7</td>
<td>Start parameter for optimization</td>
<td>79</td>
</tr>
<tr>
<td>4.8</td>
<td>Parameter retrieval pipeline</td>
<td>82</td>
</tr>
<tr>
<td>4.9</td>
<td>Configuration chart</td>
<td>84</td>
</tr>
<tr>
<td>4.10</td>
<td>Weighting tradeoff for global structures</td>
<td>85</td>
</tr>
<tr>
<td>4.11</td>
<td>Noisy texture results</td>
<td>87</td>
</tr>
<tr>
<td>4.12</td>
<td>Grid texture results</td>
<td>88</td>
</tr>
<tr>
<td>4.13</td>
<td>Tile texture results</td>
<td>89</td>
</tr>
<tr>
<td>4.14</td>
<td>Wooden texture results</td>
<td>90</td>
</tr>
<tr>
<td>4.15</td>
<td>Ring texture results</td>
<td>91</td>
</tr>
<tr>
<td>4.16</td>
<td>Texture models brick, cellulargrid, grid and perlin</td>
<td>92</td>
</tr>
<tr>
<td>4.17</td>
<td>Texture models star, symbol, turbulence and turbulenceridge</td>
<td>93</td>
</tr>
<tr>
<td>4.18</td>
<td>Texture models woodlines, woodplanks, woodstreaks and rings</td>
<td>94</td>
</tr>
<tr>
<td>4.19</td>
<td>Results for mismatches</td>
<td>95</td>
</tr>
<tr>
<td>4.20</td>
<td>Rendered Maya 3D scene</td>
<td>96</td>
</tr>
<tr>
<td>4.21</td>
<td>Results with varying lighting</td>
<td>98</td>
</tr>
<tr>
<td>4.22</td>
<td>Example of matched lighting</td>
<td>98</td>
</tr>
<tr>
<td>5.1</td>
<td>Commercial ornament examples</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>Pipeline for procedural ornamentation</td>
<td>106</td>
</tr>
<tr>
<td>5.3</td>
<td>Evolution of the placement function</td>
<td>108</td>
</tr>
<tr>
<td>5.4</td>
<td>Greedy element placement</td>
<td>109</td>
</tr>
<tr>
<td>5.5</td>
<td>Maintaining symmetry</td>
<td>109</td>
</tr>
<tr>
<td>5.6</td>
<td>Gradient images as placement functions</td>
<td>110</td>
</tr>
<tr>
<td>5.7</td>
<td>Ornamentation results</td>
<td>111</td>
</tr>
<tr>
<td>5.8</td>
<td>Vector field employment</td>
<td>113</td>
</tr>
<tr>
<td>5.9</td>
<td>Path planning around collisions</td>
<td>114</td>
</tr>
<tr>
<td>5.10</td>
<td>Manual local editing</td>
<td>116</td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison of results</td>
<td>119</td>
</tr>
<tr>
<td>5.12</td>
<td>Quantitative evaluation</td>
<td>120</td>
</tr>
<tr>
<td>6.1</td>
<td>Image reconfiguration examples</td>
<td>126</td>
</tr>
<tr>
<td>6.2</td>
<td>Reconfiguration results</td>
<td>127</td>
</tr>
<tr>
<td>6.3</td>
<td>The decomposition and rearrangement pipeline</td>
<td>129</td>
</tr>
<tr>
<td>6.4</td>
<td>Arrangers and layouts</td>
<td>130</td>
</tr>
<tr>
<td>6.5</td>
<td>Collision detection and correction forces</td>
<td>131</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison of segmentation setups</td>
<td>133</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparison of all visual features</td>
<td>134</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.8</td>
<td>Cluster layout rankings</td>
<td>135</td>
</tr>
<tr>
<td>6.9</td>
<td>Exemplary reconfiguration images</td>
<td>137</td>
</tr>
<tr>
<td>6.10</td>
<td>Future work</td>
<td>139</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Prevalence of control mechanisms ........................................ 15
3.1 Interaction methods for example-based techniques ..................... 40
3.2 Creative means for example-based techniques .......................... 41
3.3 Interaction and creative means for shape-filling methods ............. 44
3.4 Interaction and creative means for techniques with vector fields ..... 45
3.5 Interaction and creative means for techniques with curves and sketches 49
3.6 Interaction and creative means for placement techniques ............ 51
3.7 Relative distributions of the creative means ............................. 51
4.1 Database dimension, size and retrieval times ............................ 95
5.1 Interaction and creative means for organized ornamentation ........ 122
Abstract

Supporting artists with meaningful digital creation tools is an ongoing research challenge and spans over various disciplines. The objective of this thesis is to provide a better understanding of the needs for, and specific methods toward an artist-centered and cohesive creation pipeline.

To address this challenge, this thesis puts it into the context of procedural generation. The algorithmic nature of procedural generation has unique capabilities and potential to offer truly novel benefits to traditional creation processes. However, with the power of procedural generation comes difficult controllability. For the development of meaningful control mechanisms, we chose two-dimensional procedural patterns as representative design goal as their design space ranges from realistic, over abstracted to artistic.

As groundwork, a well defined general understanding of what is needed from an artist perspective is crucial. This thesis therefore starts by dissecting a creation process into the methodologies of how, what, where and when. Specific control mechanisms are classified into exemplars, parameterization, handling, filling, guiding and placing interactions and related to the stages of a creation process. For making the domain of creative creation more manageable, the creative means of navigation, transparency, variation and stimulation are defined and linked to the control mechanisms. These theoretical findings are applied to analyzing the specific goal of procedural ornamentation, which is challenging due to its complex design principles. It is shown that no single approach of the state of the art can fulfil the intricate task. The analysis bridges to various related techniques such as data-driven ones and the unification of different building blocks to a coherent pipeline, is discussed.

One central building block for all artist-centered procedural creation pipelines is the efficient navigation of the parameter space and design variation. The choice of parameters to achieve a desired appearance poses a demanding problem even for experienced artists. This thesis proposes a method to automatically determine such parameters to reproduce the appearance of input images. The technique is based on a preliminary numerical analysis as well as a user study in regard to the visual features of a pattern, which are relevant to a human observer. Addressing two-tone textures, the estimation of color and structure information is separated and the problem interpreted as image retrieval task from the space of procedural outputs. Applying a perceptually motivated image metric based on a texture descriptor enables the pre-computation of a comprehensive collection of possible parameter sets. This achieves interactive retrieval performance while supporting a large variety of designs.

After focusing on the building block of an efficient and targeted control, the addition of means for creative work is conquered. The presented technique offers the control and quality of manual creation, and the efficiency and accuracy of computation. By automating tedious tasks and including familiar input mechanisms like drawing, artists are able to focus on their creative intent. For this, the context of ornamentation is picked up again and a comprehensive creation pipeline is presented. At the core of
our system, customizable and modularly combinable element placement functions fill a space automatically under global design constraints. A provided set of example placement functions implements order based on design principles for ornamentation, such as balanced element distribution and symmetry. To create structural hierarchies and to guide an ornament to the space it fills, artists can direct the connectivity of elements with drawn strokes. Artists can also draw guides to create vector fields that organize the ornament along streamlines. Path planning automatically routes around obstacles while aligning the ornament to the boundaries of the obstacles. Hence, the method combines high-level control mechanisms, such as taking guidance from example images, to low-level control, such as placing single elements as visual accents and making local edits within a computed ornament. The feedback of designers confirms the usability of the technique.

In order to add more artistic means like explorative control, this thesis contributes by offering an artistically motivated parameterized model. The output of the model aspires to surprise an observer and to offer a different perspective to what was given. The algorithm automatically deconstructs an image into visually coherent constituents and rearranges those pieces in a surprising and aesthetically pleasing fashion. The technique is flexible and artists can create individual artistic expressions. A user study investigated the aesthetic appeal of the different results and identified preferred layouts.

In summary, this thesis analyses how artists can be supported with meaningful control mechanisms, and it offers techniques for a purposeful creation pipeline. Instead of merely adding singular features, this thesis focuses on novel methods that can complement each other as part of a cohesive pipeline. All in all, this thesis hopes to inspire innovation for artist-centered creation processes on a grander scheme.
Die Entwicklung von funktionalen digitalen Werkzeugen, die eine kreative Gestaltung unterstützen, stellt eine fortwährende Forschungsfrage und Teil verschiedener Forschungsdisziplinen dar. Ziel dieser Dissertation ist es, sowohl ein besseres Verständnis für die Anforderungen an derartige digitale Werkzeuge zu entwickeln als auch spezifische Methoden bereitzustellen, die sich zweckmäßig zu einem einheitlichen Gestaltungsprozess zusammenfügen.


Diese theoretischen Erkenntnisse werden in der Analyse des bisherigen Forschungsstandes zu der kreativen Gestaltung von Ornamenten angewandt. Ornamente basieren auf komplexen Gestaltungsprinzipien und stellen für die Automatisierung eine Herausforderung dar, die keine einzelne Methode erfüllen kann, wie es die Analyse aufzeigt. Daher berücksichtigt die Analyse verschiedene Techniken, zum Beispiel datengesteuerte Methoden, und diskutiert die verschiedenen Bausteine einer kohärenten und funktionalen Pipeline.


In ihrer Gesamtheit analysiert diese Arbeit zum einen, wie kreative Gestaltung mit funktionalen Kontrollmechanismen unterstützt werden kann, zum anderen bietet sie konkrete Techniken für unterschiedlich motivierte Schaffensprozesse. Hierfür werden nicht nur separate Lösungen präsentiert, sondern die entwickelten Methoden können sich in einem einheitlichen Gestaltungsprozess sinnvoll ergänzen. Alles in allem
hat diese Dissertation den Anspruch, einen ganzheitlichen Blick auf künstlerisch-kreative Gestaltungsprozesse zu unterstützen und basierend auf diesem zu wahren Innovationen zu inspirieren.
Introduction

The universe is made of repeating structures that can be found on all scales, from the nature of galaxies down to molecular micro-patterns. Equally ubiquitous are repetitive structures visible to the human eye. Such patterns range from natural appearances, like stone or wood textures, to highly stylized and abstracted designs, such as ornaments. Artists throughout all cultures and times have used patterning and ornaments to embellish the world around them.

The defining quality of pattern is to repeat elements and structures and it usually follows general principles. These principles can be formally described and algorithmically recreated.

Procedural models for digital pattern manufacturing make explicit use of underlying design rules and translate them into algorithms. These models can be understood as a design blueprint for a pattern class from which the execution of the algorithm creates a specific output. Procedural representations have a long history in the computer graphics community. Already in the late 1980s, a few lines of code could reproduce many natural phenomena [Ebert et al. 2002].

These algorithmic designs are traditionally controllable with a set of parameters that can be modified to produce certain visual features of that pattern class. For example, a wood texture could have the parameters frequency of lines, regularity of lines and amount of grain. Parameter ranges, and hence the variability of the design, are predefined by the model.

However, a purely parameter-based design process is difficult for artists to work with [Bourque and Dudek 2004; Lagae et al. 2010b; Gilet and Dischler 2010; Beneš et al. 2011; Lasram et al. 2012a,b]. The exploration of the design space is time consuming and the translation of a certain design goal to a specific parameter set is too abstract. The traditional derivation of parameters or control mechanisms from the underlying model – that is, a model-centered perspective – needs to be changed to an artist-centered approach. This includes the consideration of a creation process as a whole because non-creative configuration requirements and long computation times, for example, also hinder a creative workflow.
To adopt the artist perspective is especially challenging in the context of creation tools for procedural pattern generation due to the formal nature of the visuals and their at times complex underlying algorithms. Novel control mechanisms that are efficient but also intuitive for artist have to be found and integrated into a creative creation process in a unified manner.

### 1.1 Problem Statement

Digital tools for creative processes with various forms of output are indispensable for most artists. Even designs that have a distinct hand-made quality to them, such as Disney’s animation Paperman for example, were computer-generated [Walt Disney Animation Studios 2012], employing novel software solutions that are fully controllable by artists. Furthermore, more than three decades of research in academia have produced various control mechanisms for creation. These are often said to be artist-controllable but are less often proven to be so.

Most research has been executed without direct and continuous collaboration with artists. Moreover, large-scale user studies with suitable participants are usually impractical. Due to these obstacles, there is little common understanding in the research community regarding what artist needs actually are and how mechanisms are validated. Furthermore, research has typically focused on solving one specific aspect while accepting significant trade-offs for other steps in a creation process. For example, the automatic targeted control of a large design space results in long computation times and non-intuitive configuration requirements (e.g., [Bourque and Dudek 2004; Wong et al. 1998]). Equally, flexible creation pipelines that enable both automation and manual controllability often suffer from a restricted design space (e.g., [Santoni and Pellacini 2016]).

It is an important challenge to investigate creation processes from a more artist-centered perspective and to consider all tasks and efforts as a whole, from initial configuration requirements to local edits. As a further challenge, control mechanisms have to be linked to an expressive design space, which is the basis for all meaningful creative creation. In order to analyze and validate novel creation algorithms, artist feedback also has to be evaluated. However, such interdisciplinary studies still suffer at times from undefined language and vague discussions. It is an open challenge to complement user studies with a more precise and determinative terminology that is also commonly applicable and understandable by artists.

### 1.2 Research Questions

The previous statement naturally leads to general questions about artist-centered controllability in combination with distinctive output spaces and a meaningful and applicable evaluation of digital creation tools.

Tools should cover the full design spectrum, from realistic pattern, over abstracted ones to artistic experiments. Hence, also the motivation for the controllability ranges from being goal-oriented, over creative, to explorative. Investigations within
such an extensive design and control scope are based on decades of research in multiple disciplines. To bring further insights, this thesis investigates those problems in the context of two-dimensional procedural visual pattern generation, tackling a challenging and representative creation and design space. Overall, this thesis integrates into and contributes in parts to the following common major research questions.

First of all, there is no common ground on what is needed as theoretical basis for an evaluation of creative control mechanisms. More precisely, what are relevant characteristics of a creation process with digital tools? What are specific control mechanisms, ranging from global to local and from automatic to manual, with varying levels of abstraction for their handling? How do these mechanisms relate to the stages of a creation process? What are the requirements for creative creation, and how can these be customized to the context of digital creation tools?

Based on a well-defined theoretical basis the current research state needs to be evaluated. Which control mechanisms are available in the state of the art for procedural, two-dimensional ornamentation? How do current research results support, and what potential do they have for creative creation?

How to offer automatic control for realistic designs one the one hand while retaining a sizable design space and editing options on the other?

In order to extend computation with individual controllability, how can interactive creative control and specific design goals extend automatization in a unified manner for abstracted procedural patterns?

Lastly, the full range of possible artist motivations should be handled, adding to automated and realistic and interactive and abstracted techniques also artistic considerations. How can an aesthetically motivated procedural model enable artistic exploration?

### 1.3 Contributions

This work focuses on the investigation and development of control mechanisms for two-dimensional pattern generation. Based on a theoretical analysis in regard to controllability and creative creation, this thesis investigates controllability along an axis of decreasing automation, from fully automated goal-oriented control to interactive creative control to manual experimental control. Potential design spaces are similarly structured – from realistic to abstracted to artistic two-dimensional pattern designs.

Each task along this design and control scope is evaluated with a user study. For the goal-oriented and automatic control for realistic designs, a study refers to perceptually relevant features. For the creative creation of abstracted patterns, a second study investigates the usability for an artist. For the artistic exploration a third study covers the aesthetic appeal of the pattern.

In the context of the more general research questions posed in Section 1.2, this thesis makes the following contributions:
Contribution 1 –
A Framework for the Analysis of Creative Control

Chapter 2 presents an interlinking analysis of creation methodologies and a taxonomy for control mechanisms, ranging from automatic to manual control. The connection of a specific mechanism to its capabilities in a creative process are assessed.

The presented taxonomy allows for an evaluation of creativity support with a well-defined framework, which complements the common approach of user surveys.

Contribution 2 –
Analysis of the State of the Art in regard to Creative Control for Ornamentation

As exemplary design goal for a meaningful application of the analysis framework to the state of the art, ornamentation is established in Chapter 3. Due to the complexity of this pattern type, the analysis contributes, next to discussing procedural representations, with bridging between procedural and data-driven solutions and a joint classification.

Contribution 3 –
Comparison of Texture Descriptors and Optimization Strategies

In order to enable goal-oriented control for realistic pattern designs, abstracted texture descriptors and optimization strategies are compared in the first part of Chapter 4. This analysis presents numerical results as well as a user study. The user study contributes insights into the visual alignment of the texture descriptors with human perception, a problem that is theoretically highly complex to address.

The achieved results indicate that the choices made for the following retrieval pipeline are reasonable and balance a representation of relevant visual features with performance and storage requirements.

Contribution 4 –
Interactive Parameter Retrieval

Based on the insights of the comparison of texture descriptors and optimization strategies, the second part of Chapter 4 presents a perceptually driven distance metric and a strategy to contain the large parameter spaces of textures. This makes it possible to interpret the parameter fitting of a procedural texture to
a given exemplar as retrieval task. The solution contributes to a larger design
space and interactive performance.

**Contribution 5 –**

**Creative Ornamentation**

Chapter 5 presents a hybrid technique for abstracted pattern, offering both
control and quality of manual creation combined with the efficiency and accu-
rapacity of computation. Artists have control through familiar tools, while ordered
structures are automatically computed. The pipeline contributes with an op-
timization strategy that incorporates customizable and modularly combinable
design functions. Functions for ornamentation are identified and a ready-made
set is provided. Further control mechanisms, like sketching, are incorporated in
a unified manner, enabling control at all scales, from global to local.

A study in which designers used the technique gives insights into its usability
and the designers confirm the appeal, efficiency and further potential of the
methods.

**Contribution 6 –**

**Artistic Model Creation**

As a first step toward artistic exploration as control mechanisms for a procedural
model, Chapter 6 introduces an artistically motivated image reconfiguration
model. The model is carefully evaluated in regard to its parameter space and
the interlinked aesthetic quality.

A user study contributes preferences regarding certain aesthetics and gives a
well grounded direction for further developments.
Many content generation contexts in computer graphics involve creative considerations. To consider creativity is especially relevant for investigating control mechanisms because the controls provide the means for artist-centered creation. However, creativity is also a notoriously difficult topic to address because it is an ill-defined domain and involves insights from various disciplines.

One research direction solves creative generation requirements with computational creativity. This investigation of algorithms that perform creatively is again a multidisciplinary field. It merges research in artificial intelligence, cognitive science, art, design, philosophy and psychology. The interested reader is referred to the Association for Computational Creativity [Association for Computational Creativity 2018] and its various resources.

This chapter, rooted in the field of computer graphics, examines the capabilities of algorithms in enabling artists to be creative. The focus is on underlying algorithms and their control mechanisms. The computer graphics community usually addresses questions that solve specific and singular tasks, such as example-based texturing, brush-based modeling or optimizing the structural integrity of a pattern. Such tasks concentrate on efficiency, consist of exact descriptions and are well qualified for algorithmic solutions. Creativity, however, as described in more detail in the following chapter, feeds on flexibility, diversity, exploration and engagement. In order to support these characteristics, solutions have to be multifaceted, and this survey investigates the state of the art of solutions toward this goal.

The development of suitable interface designs is a well established field in the human-computer interaction community, based largely on the pioneering work of Shneiderman [2007] about Creativity Support Tools. Interface design aspects are included but are not focus of this survey. However, the need for bridging between developing the core algorithms and the interface in order to support creative-artistic intent has
been voiced by past research [Deterding et al. 2017; Isenberg 2016; Salesin 2002], and this work contributes to this effort.

The overall goal of this chapter is twofold. On the one hand, it fosters a better understanding of the nature of digital tools in the realm of creative control. On the other hand, it offers an analysis framework to classify the state of the art. For a better understanding, some terminology is assessed first. The framework then presents creation stages and their methodologies. Thereafter, control mechanisms are discussed and categorized. Creative means are summarized, and their defining properties are analyzed.

The framework is independent from any specific design goal and can be applied in various contexts.

**Contribution 1** – The introduction of control paradigms for creation processes with digital tools.

**Contribution 2** – The identification and classification of common interaction mechanisms according to control paradigms.

**Contribution 3** – The analysis of methods for creative control and the discussion of determining properties for it. The translation of interaction mechanisms to these creative means.

### 2.1 Terminology

The following clarifies the usage of terms that are relevant for a taxonomy of control mechanisms. Some aspects are discussed in detail in different sections of this thesis but are included here for an overview of terms.

**Artistic**: Refers to a task with an outcome that potentially has meaning and value beyond aesthetics and practicality. In addition to formal skills that depend on a given domain, an artistic task usually requires creative thinking as well as intuition, emotion and sensual considerations, for example.

**Canvas**: Constitutes the area in which the output is generated, similar to a canvas in a painting context.

**Shape**: Refers to the external boundary or outline on the canvas or of an object without any restrictions on the form.
Curves: Refers in this work as general term for arbitrarily shaped curves or lines without any implications for the formal representation they are based on. Curves can be computed or be derived from drawn strokes, for example. The specifics of these inherently different formal representations are not relevant for the following discussion about control mechanisms.

User Interface (UI): Refers in this work to a space that is separate from the canvas where an artist controls the system through abstracted representations, such as buttons and sliders or custom-made visual controls. In terms of controllability it makes a difference for an artist to be able to work directly on a canvas or being required to do so in a separated and often abstracted UI and hence this explicit distinction is needed in the context of this work.

Interactive: Refers in this work to systems with which an artist can interact (e.g., through a UI with reasonable response performance). In terms of control mechanisms, we evaluate interactive systems as a whole and also qualitatively. An one-time investment for an initial computation of 10 seconds, for example, is still acceptable, while a 5-second delay at each click is not.

Design space: Refers loosely to all visual results a technique can create. For example, a simple Perlin noise has a rather restricted design space of noise images, only differing, for example, in their frequency. Drawing with a pen can result in many different designs, thus resulting in a larger design space.

Expressiveness: Refers in this work to the size, the variability and the openness of a design space (these terms are in detail discussed in Section 2.3). It is important to note that expressiveness is commonly used in the context of creative controls – however, usually without a clear understanding of its meaning.

Goal-oriented control: Refers to an clearly targeted design task and stands in contrast to exploration. For example, reference images might be given, or an artist may have a clear mindset about how the output of the creation process should look.

2.2 Taxonomy of Control Mechanisms

To offer creative means is a layered process. At its core, the generating algorithm should ease the creation process for an artist and fully use the benefits of computational control. However, the algorithm also has to be flexible enough to allow for various interaction methods and individual design goals. There is a delicate balance between giving artists as much control as is needed without burdening them with unwanted details and tedious manufacturing. Similarly, this analysis framework is multi-layered to distill and summarize the different aspects of creative control.

A classification of creative means cannot be derived directly from the related work. Past authors have followed various motivations and have emphasized different aspects
When describing their work and results. In order to classify the work in an objective and unified manner, we first analyze the actual presented control mechanisms and relate them to general control paradigms. From this analysis, we then illustrate the creative means. While the classification of the control mechanisms is directly taken from the authors’ descriptions, the following classification into the creative means has an interpretative nature to it. Despite our best efforts to give a clear reasoning for each classification, we also agree that this is not always unambiguously manageable. Thus, this taxonomy should be viewed as a step toward an objective discussion about terms such as artist-usuable and creatively controllable as well as a more realistic usage of these terms.

2.2.1 Control Paradigms

A creation process can be described by answering the questions of *how, what, where, when* and *who*. These paradigms can be discussed in various creation contexts and could even be translated to traditional media such as aquarell on paper.

*How*

How is a control executed or an input given by an artist? How far is it from the visual result on the canvas?

- **File**: The control is externally given, such as with code or a configuration file.
- **UI**: A separate UI is given through which an artist gives input and activates states. User interfaces are often in close proximity to the canvas, carefully designed and easily usable. However, because they detach the work from the actual output, UIs still have an abstract nature. An artist must actively translate his or her interaction with the UI to the resulting output on the canvas.
- **On canvas**: Controls are executed directly on the output canvas. Most of these controls require an activation or selection of a tool in a separate UI, such as selecting a pen for drawing on a canvas. In this case we consider the pen primarily as a control mechanism. There are cases where controls cannot clearly be classified as either UI or on canvas. A pen, for example, can have different characteristics that an artist needs to set in the UI. Ideally, the adjustment of settings should be as seamlessly integrated into an on-canvas tool as possible (e.g., with selection choices appearing as tool tips).
What
What does an artist give as input? What is the level of abstraction of the content that an artist works with?

- **Code**: Input is a syntactically structured and formal language.
- **Value**: The input is a single value, chosen from a range – for example, with a slider.
- **Intermediate**: The input is visual but still of an abstract nature, such as controlling sketches for a mask or arrows for directionality. Again, artists have to interpret how these inputs affect the result.
- **Element**: The input constitutes a component of the resulting pattern.

Where
Where does the input have an effect spatially and what is its area of influence?

- **Global**: The input has global influence (e.g., by filling the whole space or by adjusting all elements on the canvas).
- **Region**: The input has an effect in a region of the canvas (e.g., on a drawn curve).
- **Local**: The input has an effect on one specific element.
When

When can input be given and at what time in the creation process is the control executed?

- **Before**: Input is given before the actual creation process.
- **During**: Input is given during the creation process, when parts of the results are already visible. This is typically a painting mechanism. However, some processes can also be paused and adjusted.
- **After**: Input is given after the creation process. The result is visible to the artist and can be adjusted retrospectively.

Who

Who can give the input in regard to the type of skill set needed? This category can be in part derived from the above characteristics of how and what. In most general terms this category can be classified as the following.

- **Programmer**: To give input with dissecting analytical-formal and logical thinking and the ability to abstract.
- **Artist**: To give input with comprehensive intuitive-visual and spatial thinking and the ability to create (e.g., by drawing).

The who category is listed here for completeness. However, to fully answer the questions of needed competencies, skill- and mindsets, including the accompanying psychological and artistic aspects, is out of the scope of this thesis and requires knowledge in fields other than computer science. The following discussions are rooted in computer graphics research and aim for an assessment of algorithmic controllability. Hence, the classification of who specifically is most suitable to use a tool, is put aside.
Summary

<table>
<thead>
<tr>
<th>Control Paradigms</th>
</tr>
</thead>
<tbody>
<tr>
<td>• How is input given by the artist?</td>
</tr>
<tr>
<td>• What is given as input?</td>
</tr>
<tr>
<td>• Where does the input have an effect?</td>
</tr>
<tr>
<td>• When does the input have an effect?</td>
</tr>
</tbody>
</table>

2.2.2 Control Mechanisms

For a meaningful analysis, the above classification must be further broken down into the specific control mechanisms.

The following low-level characteristics categorize input modes and their primary effect. Because this survey focuses on interfacing algorithms, UI specifics, such as the layout of buttons, are not considered. Once the specific control mechanisms are analyzed, they constitute a method in combination with the above control paradigms.

Initialization

• **System Configuration**: Required overall setup of the system, such as computing caches or training a model. This is usually a one-time investment.

• **Task Initialization**: A non-creative task that has to be executed each time in order to produce an output, such as selecting the specific optimization algorithm.

Exemplars

• **Image**: An example image that should be matched in its entirety. Examples are usually pixel data.

• **Element Arrangement**: An example element arrangement that should be matched in its entirety. Elements are usually separate shapes and might carry additional data.

• **Element**: One specific asset that becomes in the result part of a whole. Elements can be shapes or pixel data.

Parameterization

• **Visual Output**: Parameters that can adjust visual features directly in the output.
• **System/Generation**: Parameters that influence the output indirectly, such as parameters for an optimization algorithm or constraints.

### Handling

• **Visualization**: Any type of visual interface that goes beyond the standard UI elements, such as sliders and buttons.

• **Image-Based**: Images as indirect control input, such as pixel data masks.

• **Sketch-Based**: Sketches and curves directly put on the canvas – for example, the drawing of a mask with a pen tool.

### Filling

• **Shapes**: A space to fill (e.g., a specific shape).

• **Masking**: Areas within the shape to fill that should remain unaffected.

• **Curves to Fill**: A one-dimensional curve or path to be filled. The curve is given as a whole before the filling starts.

### Guiding

• **Painting/Strokes to Follow**: A curve, usually created by mouse movements or with a stylus pen, that is filled with output elements while the curve is generated – often understood as brushing.

• **Directions**: Visual elements such as intermediate curves, arrows or output components that define directions for the design to follow (e.g., with an underlying vector field).

### Placing

• **Element Placement**: The direct placement of components on the canvas as part of the final result.

• **Element Drag & Drop**: Drag and drop of components on the canvas within the existing result.

For interrelating the control mechanisms to the control paradigms, we considered the publications that are investigated in Chapter 3 State of the Art of Creative Control for Procedural Ornamentation. Due to the diversity of the underlying methods and the different design goals of the considered body of work, we believe this to be a representative summarization.

Table 2.1 shows that global, hence automatic, control is usually enabled through intermediate representations, such as an example image, while on the other end of the spectrum, the placement of elements as part of the actual output is local, and automation is lost.
Parameterization and the different types of handling also require abstracted input from an artist, such as the use of a slider. Sketch-based controls, such as an eraser, move the interaction onto the canvas and can make small-scale adjustments. The definition of a space or a curve to fill and masking areas is also usually done directly on the canvas but only influence the output indirectly.

A painting mechanism simultaneously creates the output directly on the canvas but can only do so in a limited region depending on the brush size. All other inputs are typically given before or after the generation of the output.

This classification underlines that a focus on one control type, as is usual in computer graphics research, leads to the common trade-off between global automation and local manual manufacturing. In order to support creative work, control mechanisms need to be combined in a novel and unified manner.

Summary

<table>
<thead>
<tr>
<th>Control Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
</tr>
<tr>
<td>Exemplars</td>
</tr>
<tr>
<td>Parameterization</td>
</tr>
<tr>
<td>Handling</td>
</tr>
<tr>
<td>Filling</td>
</tr>
<tr>
<td>Guiding</td>
</tr>
<tr>
<td>Placing</td>
</tr>
</tbody>
</table>

2.3 Creative Means

In order to solve complex tasks for which no obvious standardized solution is apparent, thinking creatively enables a change in perspective and the development of different solutions.

Creativity is also an established field of research in computer science. On the one hand, there are efforts to develop algorithms that perform creatively. On the other hand, there is the common goal of supporting human creativity with digital tools, which is the focus of this survey.

In the context of this goal of enabling human creativity, Cherry and Latulipe [2014] presented the quantifiable Creativity Support Index (CSI), which has found its way into the graphics community [Shugrina et al. 2017]. The index measures how well a tool enables creativity based on a psychometric survey. The development and
2.3 • Creative Means

Table 2.1: Prevalence of control mechanisms in the literature: In total, 40 publications are included (the discussed state of the art work of Chapter 3). Please note, that the totals of each step (how, what, where, when) can exceed the total of that category as it can be implemented within multiple usage scenarios.

<table>
<thead>
<tr>
<th>HOW (out of 40)</th>
<th>WHAT</th>
<th>WHERE</th>
<th>WHEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>File</td>
<td>UI</td>
<td>Canvas</td>
</tr>
<tr>
<td>Setup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Initialization</td>
<td>14</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Exemplars</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Arrangement</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Element</td>
<td>9</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Parameterization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual Output</td>
<td>31</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>System</td>
<td>16</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Handling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual UI</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Image</td>
<td>9</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Sketch</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Filling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shapes</td>
<td>32</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>Masking</td>
<td>9</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Curve</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Guiding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Painting</td>
<td>6</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Directions</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Placing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Drag&amp;Drop</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

validation of the measurement dimensions – namely, exploration, expressiveness, immersion, enjoyment, results worth effort, and collaboration – are mainly based on user tests. Cherry and Latulipe [2014] quantified the specific phrases participants used to describe a creative process. However, a clear definition of terms like exploration and expressiveness is missing or the meaning of a statement such as “I was able to be very creative, [...]” is left open.

In order to assess the related work, which is discussed in the next chapter, a quantified user study for all presented techniques is not feasible. Doing so would also not be meaningful because the support of creativity is not a goal for most methods. However, most methods do offer carefully developed control mechanisms. We propose it to build a discussion of the means for creativity on the presented control mechanisms in a publication and on the specifics given from the authors. Based on the given information, we reflect on the potential for creative means in a meaningful way, even if creative control was not necessarily the authors’ intention. This survey is meant
as a step toward understanding the creative control options within the current state of the art. In terms of measurement dimensions, this survey can be seen as a subset of the more general and user-study-based classification with the *Creativity Support Index*.

In order to assess the means of a technique to enable human creativity, a common understanding of human creativity is needed. Academic discussions about what constitutes creativity have a long history in the field of psychology [Weisberg 2006], cognitive science [Boden 2004] and philosophy [Gaut 2010] and is ongoing.

Recently, Robert W. Weisberg, a cognitive psychologist, made a valuable advancement in defining creative processes [Weisberg 2006]. Weisberg’s main argument is that a creative person “intentionally produces a novel product” (p.70). Weisberg explicitly decouples a possible generally accepted value of a product from being the result of a creative process (please refer to [Weisberg 2006], p.63, for a detailed argumentation). We follow Weisberg’s exclusion of the potential and often diffuse value of the result of a creative process and also focus on the creative intent.

For a better understanding of *novelty*, Boden [2010] described it as a surprising product, one that the creator did not directly anticipated (p.30). Magret Boden also differentiated between a product being surprising or novel to oneself in contrast to something being universally novel ([Boden 2010] p.30). In this survey of creativity regarding control mechanisms, we only include novelty in reference to the expectations of a single artist.

The integration of *intention* in describing a creative process is crucial for the development of meaningful algorithms. Weisberg explains that a painter who accidentally stains a painting – a stain which is later applauded by the art world as an innovative technique – cannot be considered a creative result. Hence, algorithms need to enable artists to follow their intentions with transparent and controllable mechanisms. The idea, for example, of an algorithm producing a large number of random design choices for an artist to choose from contradicts the principle of intention.

Further arguing against randomness in creative processes, Weisberg states that a creative process entails “staying within the box.” A creator needs domain-specific knowledge and expertise in order to come up with something novel or surprising. Weisberg bases his argument on an exemplary in-depth analysis and on the empirical evidence of unique case studies: Watson and Crick’s formulation of their DNA model, representing a scientific creative process; the Wright brother’s invention of the airplane as creative engineering task; and Picasso’s creation of Guernica as an artistic creative work ([Weisberg 2006] p.6, [Markman and Wood 2009] pp.28-38). Weisberg rejects the common perception of creativity as being an “unfathomable leap of insight” and advocates its systematic accessibility. He argues that this perception of creativity results from missing context and domain knowledge and from neglecting to include the whole process that leads to a novel product.

We applied Weisberg’s argument in the context of control mechanisms by requiring techniques to enable the artist to fully understand the domain they work with. Cause and effect of interactions as well as the overall options to control the output must be transparent and navigable.
Weisenberg concludes his considerations about creative processes and innovation by stating that “you must also work to broaden and deepen your database” [Markman and Wood 2009]. Hence, control mechanisms not only need to be transparent and fully steerable for an artist, but they also must offer a large space for an creator to explore. Boden [2010] describes this as a landscape to navigate through. This increase of possible options is a core aspect of many common creativity techniques, such as brainstorming, and must also be used for the development of digital tools [Terry et al. 2004].

It is interesting to note that a substantial body of work [Onarheim and Wiltschnig 2010; Shih et al. 2011; Biskjaer et al. 2014; Stokes 2005] suggests that constraints also stimulate creative processes. This seemingly contradicts the argument of offering a large design space to explore and again emphasizes the need for a design space to be meaningful and well framed for the domain it represents. It has to provide space to delve into without the danger of getting lost. Classical brainstorming, for example, is on the one hand based on the idea of coming up with as many answers to a question as possible – with no restrictions. On the other hand, a brainstorming session starts with a carefully crafted problem statement, which is supposed to be as precise and descriptive as possible. Hence, human brainstormers intuitively remain in the domain of the problem statement and exclusively offer solutions related to the problem. Therefore in a system that computes options for a design space, all options need to make sense, while “enabling someone to see possibilities they hadn’t glimpsed before” [Boden 2010].

Well designed constraints can act as stimuli for discovering unpredicted results. Common creativity techniques often include such stimulating constraints or motivations to guide the exploration in a specific direction. For example, with the Six Thinking Hats technique, each hat represents a specific mindset, such as “critical” or “emotional”, with which a participant should operate. This technique enables a large variety of possible stimuli and cues such as associations, analogies, abstractions, visualizations and reversals, including purely random inputs. Stimuli are a field of active research and as mentioned above, the usefulness of random cues has been doubted. Christensen and Schunn summarize their insights ([Markman and Wood 2009], pp.48-69) about cognitive support for creative processes, saying that the pool of random stimuli needs to be restricted to increase the opportunity for novelty and to decrease the probability of misleading failures (p.68).

In the process of enabling creativity, the target audience is also an influencing factor. Each skill level requires its own unique type of support. Cherry and Latulipe [2014] discuss the handling of different user competencies as another example for the importance of balancing simplicity and expressiveness. The authors mention that more expressive but also more complex tools score higher on the CSI. The framework presented in this chapter, does not explicitly discuss the appropriateness of a technique for different skill levels (unless it specifically distinguishes a related work, as, for example, in [Benedetti et al. 2014]) but instead focuses on the general suitability of a technique to create the design goal with a reasonable training curve for an average artist.
To summarize, control mechanisms that support creative methods should offer variation, the chance of steerable exploration and meaningful stimuli, according to the domain a given mechanism serves. For these characteristics, there is no clear translation into quantifiable metrics, such as timings or error rates, which are standardized measurements for productivity [Cherry and Latulipe 2014; Shneiderman 2007].

For our framework, we understand variation as the size of the design space within the context of the technique. For the exploration of different designs we distinguish between the general controllability necessary for navigating a design space (“there are many different roads in the landscape”), and the transparency of that navigation and the understanding of cause and effect when using the tool (“I have the map to the landscape and know how to get from one point to another”). Lastly we investigate the stimuli of a method and its suggestive capabilities. All categories can be seen as somewhat loose and experimental and aiming toward a better understanding of requirements for creative controls.

Each of the classification categories – namely navigation, transparency, variation and stimulation – is summarized in a discussion-based rating of non-existent, weak (◦) and strong (●). The definition of weak and strong for each category is clarified in the following. The judgment of one classification category might also be closely connected to another one. For example, a technique with little variability is much easier to navigate. Equally, limitations in navigation or transparency can result in stimulating surprises.

The specifications for the stated quantities for strong and weak are intended to make the techniques comparable and to give an overall impression of their capabilities. However, when applying the analysis framework, the specific numbers for each category might need to be adjusted for a creation context by experts for that specific domain.

### 2.3.1 Navigation

The means of navigation describe whether a creation processes is efficiently manageable as well the extent of the controllability.

- **Interactive:** Refers to a system with ideally no noticeable delays when executing controls and computing results. Lengthy, non-creative configuration requirements are also potentially distracting. Hence, a thorough analysis should consider the whole process an artist has to go through to produce a result.

  We liberally accept a manageable performance as strong when the reported performance overall is under five seconds and as weak for a performance under 30 seconds.

- **Quantity of Controls:** This category indicates how flexible and controllable a technique is by counting the number of different controls that can be adjusted for one output.
Strong refers to at least eight (approximately half of all discussed control mechanisms) different control mechanism types (listed in Section 2.2.2) and weak, to at least four.

Ideally, this category would refer to the ratio of visual features of the possible output that are relevant to humans to controllable features. This would ensure that the controls cover all necessary features and that they complement each other. However, the identification of generally describable, perceptually relevant visual features is out of the scope of this chapter and left to future work.

- **Navigation History:** Describes the ability to go back and forth in one’s own creation process, such as using an eraser.
  
  Strong refers to a navigable editing history of at least three steps, while weak refers to a clearly defined undo functionality.

### 2.3.2 Transparency

The means of transparency describe how clear the understanding of cause and effect within the system are.

- **Control Domain:** Refers to how well controls are mapped to visual features and how well they cover the possible design range of each feature. A high-quality control should not have any overlapping effects with other controls.

  We consider the control domain setup to be strong if there is sufficient controllability, meaning a match of visual features with controls, and if the controllability is meaningful, hence the controls all affect different visual features. The control domain is weak if only one of the aspects is well developed. This category highly depends on the creation context and needs to be individually defined for a design space.

- **Control Communication:** This category describes how well controls (e.g., with a visualization and/or little abstraction) represent their effects on the result. For artist-centered tools this could mean that controls should be visual and directly on the canvas.

  Hence, strong refers to at least five of the control mechanisms that are less abstract, namely from the categories of exemplars, handling, filling, guiding or placing, while weak refers to at least three.

### 2.3.3 Variation

The means of variation indicate how visually different the results can be.

- **Size of the Design Space:** A design space is limited if all results look rather similar to each other and are part of a specific design class. A large design space of one technique allows, for example, for different texture classes such as combining stochastic and structural creation.

  We consider this category to be strongly pronounced if the technique includes at least five design types, while it is weak with at least three.
• **Openness of the Design Space:** Refers to the limitlessness of possible designs and that there is no attachment of the technique to a specific design class. An open design space enables an artist to come up with a distinctive individual style, for example. Different artists can create inherently different and unique results with the same tool if it has a open design space.

We rank a technique as *strong* with at least five of the following characteristics and as *weak* with three. We total the number of the least determining controls, namely sketching, painting and placement mechanisms. As the possibility to add different creation models gives undetermined design options to a technique, both in regard to design logic and specific elements, for example with different procedural texture models, we count this option twice. If only the option to provide any element, for example graphical assets, is given with no influence on the design logic, this is counted once.

We do understand that a clear definition of the available different design classes is needed. However, these dependent on the design context. The analysis in Chapter 3 presents possible classes for the context of procedural generation.

### 2.3.4 Stimulation

The means of stimulation indicate how well an artist can enter a pleasurable and stimulating workflow.

• **Immersion:** How natural and enjoyable the usage of a system feels.

An immersive technique needs to be fluent to navigate, controls have to be intuitive and the design space large enough to not to hit its boundaries while using the tool.

Hence for a ranking as *strong*, the categories control quantity, domain, communication and the design space size have to be ranked as *strong*. If at least one of those are marked as *weak*, the immersion experience is also marked as *weak*.

• **Stimuli:** The support to find surprising results – for example, with design suggestions or variations of the input.

Options to support stimulation are still underrepresented but on the rise with machine learning techniques. A clear definition of this category is not feasible at this point.

For stimulation being more broadly accepted as relevant research question and required element for tools that enable creative creation, first more research has to go into understanding what constitutes effective stimulation before it can be implemented. A psychologically well-grounded theoretical assessment of stimuli is out of scope of this thesis. For the following discussion stimuli are included if a technique is explicitly evaluated in regard to its stimuli, for example with a survey. However, this category can not be included overall and in a unified manner and its generalizable assessments is left to future work.
Summary

*Creativity*

- Intentionally producing – for oneself – a novel and surprising product.

*Creative Means*

- Navigation
- Transparency
- Variation
- Stimulation

2.4 Conclusion

Towards the goal of supporting artists in their creative work with innovative and meaningful tools, first a well defined, generalized and interdisciplinary understanding of creation processes and creativity is needed.

We dissect a creation process into overall characteristics and classify specific control mechanisms by their interaction types. The taxonomy shows the capabilities of the different control mechanisms and potential trade-offs between approaches.

For handling the ill-defined topic of creativity, we follow the definition of creativity as intentionally producing a novel and surprising product. We establish means for creative control and relate specific control mechanisms to creative processes. By this we further a more objective judging of the ability of a technique to support creativity and a detailed comparison of methods.

However, some aspects of the analysis framework still leave room for interpretation. Knowledge from other disciplines, for example in regard to the perception of visual features, can contribute with valuable insights. We hope that our results inspire such research towards a quantifiable analysis of creative control.
With a well defined analysis framework for creative control in place, we chose procedural ornamentation as a case study to apply the framework.

Procedural representations are notoriously difficult to control [Bourque and Dudek 2004; Lagae et al. 2010b; Gilet and Dischler 2010; Beneš et al. 2011; Lasram et al. 2012b,a], and much effort has gone into investigating control mechanisms within specific contexts. Botanical and architectural procedural modeling, for example, are popular fields of research. There have been summarizing surveys for procedural noise [Lagae et al. 2010a], landscapes [Smelik et al. 2014] and urban spaces [Vanegas et al. 2009] as well as in the context of games [Hendrikx et al. 2013; Togelius et al. 2011], including even a short summary of models for ornamentation [Whitehead 2010]. However, the overall investigation of generating and designing decorative patterns and ornamentation is less prominent. This could be credited to ornamentation being an ill-defined domain due to it involving creative-artistic considerations. Nonetheless, its diverse design aspects make ornamentation a rich and compelling topic.

On the one hand, ornamentation includes repetitive and ordered structures that are often considered as textures, thus demanding automatic and procedural creation. On the other hand, ornaments require a global layout, adapt to the space they are filling and include visual hierarchies and highlights that are singularly placed with creative intent. This artistic challenge either requires computational creativity, or an artist’s creativity must be supported with meaningful digital tools.

Ornamentation is also an interesting testing ground for addressing the delicate balance between giving artists as much control as is needed without burdening them with unwanted details. Procedural representations are well suited for expressing repetitive and ordered structures and enable parametric control. However, traditional
procedural modeling approaches are in need of novel control mechanisms that are intuitively navigable, flexible and engaging.

The contribution of this survey is twofold. On the one hand, its categorizes the state of the art with regard to control mechanisms, translates this categorization to overall control paradigms and investigates their means for creative control. On the other hand, it selects the work not by its underlying algorithms and creation techniques but by its design goal, namely ornamentation. Ornamentation includes a variation of representative creation challenges, such as combining ordered fillings and repetitive structures with individual global layouts and highlighting components that might break that underlying order.

The focus on the visual output instead of specific underlying algorithms allows for a novel and unifying discussion of techniques and merges a discussion of work that is traditionally studied separately. Thus this focus furthers a common understanding of creative control mechanisms.

**Contribution 1** – The identification, summary and classification of related work from various fields of research. The evaluation of the work in regard to its means and potential for creative control for ornamentation.

### 3.1 Terminology

The following clarifies the usage of terms that are relevant for ornamentation and its taxonomy of control mechanisms in the context of this work. Some aspects are discussed in detail in different sections but are included here for an overview of terms.

**Pattern:** Constitutes a generic term for any type of repeated, often regular, arrangement [oed 2017].

**Texture:** In the context of computer graphics, texturing is commonly understood as modeling a surface’s color (i.e., their color texture) with no implications for a design, while designing a surface’s interaction with light is understood as shading.

Texture refers in its traditional meaning to the character of a woven fabric [oed 2017] with properties such as fine or coarse. This work understands texture similarly with regard to potentially repetitive structures. Lin et al. [2006] define a spectrum for such structures. The spectrum ranges from regular deterministic textures with distinguishable texture elements recurrently placed to irregular placements to purely stochastic textures.

**Decor:** Refers to elements that generally embellish and beautify without implying any specific design rules in itself.
Ornament: Constitutes a specific type of decor adhering to certain design rules, such as order, hierarchical structures, space adaptation and visual contrast and accents (Section 3.2 Design Goals).

Creative: Refers to a task that intentionally produces a novel, non-standard outcome. Please note that this work refers to the academic usage of the term. In common language, a creative task is often misunderstood as one that produces a visual product.

Procedural: Refers to the production of output by evaluating an algorithm or a rule-based system.

Data-driven: Refers to the production of output based on given, and usually limited, data.

Parameterized: Refers in its original meaning to a system that is based on an implicit equation. However, in regard to control mechanisms and for this work, it simply means that a system offers separated, individually controllable characteristics. Parameterization commonly does not imply a procedural representation but can be part of any technique, including data-driven ones.

3.2 Design Goals

In this investigation of creative control for procedural modeling, the design goals of ornamentation trigger challenging questions. Ornamentation goes beyond the basic repetition of elements to create patterns, and requires the use of formative design principles.

The Oxford English Dictionary [oed 2017] defines ornaments as nonessential accessories intended to adorn. There is no functionality to an ornament other than to beautify a manufactured article without changing its shape or character [Ward 1896].

The term ornament can be found in a large variety of contexts, such as in architecture, music or poetry, but this work only refers to two-dimensional visual ornaments. While ornaments may carry symbolic meanings in the arranged elements [Wornum 1896], this work does not include semantics but focuses on visual qualities.

Different cultures and times resulted in various ornamental styles, with great differences in the details as Figure 3.1 shows. Nevertheless common underlying design principles for ornamentation can be identified.

Ornamentation can be understood as an accurately defined type of decor that follows a structural logic [Ward 1896; Moughtin et al. 1999; Arbruzzo et al. 2006]. In addition to its aesthetic appeal, an ornament is perceptually distinguished by a sense of order and by its alignment to the space it fills (as summarized by Wong et al. [1998] and originally stated by Ward [1896]; Dresser [1875]; Arbruzzo et al. [2006]).
Arbruzzo et al. [2006] elaborate on ornamentation as follows:

[An] ornament is inextricably linked to scale and proportion, to form and order. In this context, the modus operandi of ornamentation is always to reinforce an existing order: to conform to its partner in the ornament-object relationship, for ornament always has a partner in that which is ornamented.

An underlying perception of order in an ornament is established by even repetition and a balanced distribution of elements, with an intentionally designed and artificial quality [Ward 1896]. Balance can be achieved with a careful composition of elements, and such balance is built on symmetrical arrangements in most ornaments. Compositions are not limited to the repetition of the same element, but different visual qualities can create various relationships. Visual characteristics attract the eye differently and the visual weight of a feature can be used as a measure for the degree of attraction. For example, large, dark and highly saturated colored elements have a
greater visual weight than small, light and desaturated ones. These visual weights can be used to create visual correlations (for example, based on Gestalt psychology—a topic too wide for a discussion here) and can counterbalance each other. A larger and lighter colored element might have the same visual weight as a smaller, darker colored one. Hence, varied elements with different visual properties can be combined and still make a balanced whole.

Hierarchical compositions further increase a sense of order but are also used for creating contrasts (e.g., foreground vs. background) and accentuating structures (e.g., framing). These structures are often used to elaborate and accentuate the form of the space they fill, hence building the ornament-object relationship described by Arbruzzo et al. [2006]. The following differentiation of an ornamental decoration gives an intuitive understanding of this aspect [Arbruzzo et al. 2006]: Wallpaper can be trimmed for different rooms, but the design is not reproportioned or altered. An ornament, however, is fitted to and references the logic of the space it is designed for. Without adjustment, it cannot be transferred to a different space.

Contrasts and accents are crucial for the visual appeal of an ornament [Wong et al. 1998; Ward 1896; Moughtin et al. 1999]. Single, visually dominant elements and structures might not follow the underlying order of the ornament at all, breaking an otherwise too homogeneous appearance—again distinguishing ornamentation from wallpaper.

Figure 3.2 gives an example of how the described design principles are combine seamlessly into a coherent design.

It takes artistic expertise to balance the contrast between carefully chosen visual accents and to create a sense of order by applying compositional rules and by complementing the space. However, it is exactly this combination of qualities—rule-based composition and repetition on the one hand and the placement of visual accents and the breaking free from order on the other—that make ornamentation an interesting but highly challenging field of algorithmic research in the context of computer graphics.

Ornamentation exemplifies the common challenge of enabling control for tasks for which humans are indispensable in combination with the automation of tedious manufacturing and the computation of structuring rules.

Summary

<table>
<thead>
<tr>
<th>Ornamentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Perception of order</td>
</tr>
<tr>
<td>• Hierarchical compositions</td>
</tr>
<tr>
<td>• Adaptation to the space</td>
</tr>
<tr>
<td>• Contrasts and visual accents</td>
</tr>
</tbody>
</table>
Figure 3.2: Exemplary dissection of visual characteristics fulfilling ornamental principles. Single features often support several principles, as, for example, the frames and borders create a hierarchical composition, an adaption to the space the ornament fills and visual contrasts. Image source: [Morris and Dearle 1910].
3.3 Models

In the context of computer graphics, generation techniques are differentiated into procedural and data-driven approaches. This understanding applies equally to the generation of geometry, animations and texture, for example. Procedural techniques describe the visual output by evaluating an algorithm, while data-driven approaches rely on existing data, such as photographs.

The underlying regularity of an ornament is based on a repetitive and balanced distribution of elements, usually following hierarchical structures. These characteristics can be efficiently implemented by procedural approaches [Stáva et al. 2010] because they automatically fill a space based on generative rules. An artist should be freed from such tedious, non-inspiring and repetitive tasks. In order to execute order, computational generation techniques are not only an easement, but they also perform in a potentially more precise and less error-prone way than a human artist. Hence, procedural representations are an ideal basis for ornamentation. However, the creative demands of laying out space-specific designs and of placing highlights must also be considered. Procedural models must be augmented, and different approaches must be unified in order to enable the control and quality of manual creation as well as the efficiency and accuracy of computation. For this goal, this survey focuses on procedural models as a basis, but it also integrates and highlights promising or desirable characteristics of suitable data-driven techniques.

The following categorizes procedural models as stochastic, function- and rule-based, grammar-based, simulation-based, and artificial intelligence-based. A data-driven approach is discussed, and models specifically developed for ornamental designs are summarized.

3.3.1 Procedural

Ebert et al. [2002] describe procedural techniques as algorithms and mathematical functions that synthesize a model or an effect.

Solely equation-based representations are considered the “purest” form of procedural modeling [Smelik et al. 2014]. This approach gained immediate importance in the early days of computer graphics. Simple equations are able to reproduce many natural phenomena – such as wood, stone, water, smoke and plants – with only some lines of code in the range of kilobytes, hence being memory efficient. The main appeals of such procedural representation include its compactness in combination with being continuous, scalable and unbound to a specific resolution. While procedural generation techniques have been a constant basis for generating content for games, its characteristics of memory efficiency and unlimited resolution are more important than ever with the rise of virtual reality.

The compactness and efficiency of a procedural model also enable parameterization, resulting in the model being responsive and flexible. Parameters usually represent certain visual characteristics and their amplification. Parameterization brings the crucial benefit that, for example, textures remain editable throughout the entire visual effect production pipeline.
However, the effectiveness of traditional parameterization in helping an artist fulfill design goals is debatable. Ebert et al. [2002] argue that parameterization brings the benefit of a few parameters controlling large amounts of details. At the same time, this is potentially problematic for the realization of specific designs because these often require full individual control of all visual elements. Additionally, parameters are often non-intuitive due to representing overly abstract characteristics of the underlying functions and having overlapping effects [Bourque and Dudek 2004; Lagae et al. 2010b; Gilet and Dischler 2010; Beneš et al. 2011; Lasram et al. 2012b,a].

In addition to the disadvantage in the control of a procedural representation, the creation of the representation itself, the procedural model, requires considerable effort – even though it is only a one-time investment. For the appearance of a model, the focus usually lies on a more generic design, like a texture class. For procedural textures specifically, handling antialiasing efficiently can also be challenging. For a valuable and in-detail survey of function-based design principles of procedural models with focus on textures, the interested reader is referred to Ebert et al. [2002].

Procedural models are not limited to purely function-based designs. For example, the pioneering work of Prusinkiewicz [1990] applies the grammar-based L-system to algorithmically model plant growth, an approach extensively investigated by the computer graphics community and considered as procedural.

The following classification of core mechanisms for procedural generation is based on the taxonomy of Hendrikx et al. [2013] for procedural content generation in the context of games and the categorization of Smelik et al. [2014] for procedural modeling for virtual worlds.

**Stochastic Models**

For stochastic models, noise functions generate maps of random values. They can either be used in their original form as procedural model or as a basis function. Visual features can be added by combining multiple layers of the noise in different resolutions.

Perlin noise [Perlin 1985] is one of the most well known noise functions and can be used to directly create many natural phenomena. Typical noise functions are lattice value noise, lattice gradient noise (e.g., Perlin noise), sparse convolution noise and spectral noise [Ebert et al. 2002; Lagae et al. 2010a]. These “pure” procedural programs also have the advantage of being well suited for optimization, such as parallelization, because they can be randomly evaluated in constant time [Lagae et al. 2010a].

In the context of creative control for ornamentation, stochastic models build a basis for many designs but their design spectrum and controllability are limited.

**Function- and Rule-based Models**

Function-based models extend the class of stochastic models by layering and combining a variety of functions to form a visually complex pattern. Typical building blocks are periodic, spline, step, clamp and conditional functions [Ebert et al. 2002].
Rule-based models are part of individual, and often quite complex, generation systems that can be context-dependent and/or design-specific. Rule-based models are programs that relate to and partition the space to fill and follow propagation rules. The algorithmic core often handles proxy shapes, while for the result graphical elements, such as vector graphics, are mapped to the proxies.

Rule-based procedural models are the most suitable for ornamentation and novel control mechanisms because their iterative generation logic is the most open and flexible [Wong et al. 1998; Měch and Miller 2012]. They can implement any designs and include any elements. Moreover, within a suitable pipeline, they can potentially take global constraints into consideration and build structural hierarchies.

Grammar-based Models

Grammar-based models are also considered “purely” procedural [Smelik et al. 2014]. These models form grammatically-correct sentences from individual words, based on a system of rules. Originally introduced in theoretical linguistics by Noam Chomsky in the late 1950s, grammars are applied in computer graphics to generate objects such as plants from elements encoded as letters or words [Hendrikx et al. 2013]. Prominent techniques are L-systems and shape grammars. An emerging subgroup of grammar-based models includes probabilistic inference into the derivation of correct sentences from a grammar.

In recent years, there have been a variety of successful grammar-based approaches for certain aspects of ornamentation [Beneš et al. 2011; Talton et al. 2011; Ritchie et al. 2015]. However, grammars are difficult to set up and to design [Št’ava et al. 2010]. Because the execution process is inherently hierarchical, grammar systems have difficulty in supporting creative control from a global to local scale.

Simulation Models

Simulation models are based on techniques that approximate complex phenomena for which an analytical solution is unmanageable or unavailable. Hendrikx et al. [2013] further group simulation techniques into cellular automata, vector and tensor fields, and agent-based simulations.

In the context of ornamentation, simulation models have been less relevant, with the exception of vector and tensor fields [Ijiri et al. 2008; Li et al. 2011; Saputra et al. 2017]. For simulation models usually interactive performance is a challenge, as well as the control on an element level. However, the potential to create a layout within a space and adapt to the characteristics of that space within a simulation system as well as the possible design variability call for further investigation.

Artificial Intelligence Models

Artificial intelligence models represent approaches that go beyond the direct execution of specific rules. For example, they automatically optimize results based on fitness or error functions, or they apply planning steps. Hendrikx et al. [2013] group this class into genetic algorithms, neural networks and constraint satisfaction and planning.
In recent years, machine learning has been introduced into procedural content generation with the same impact as in all other computer science fields [Summerville et al. 2017]. The potential of machine learning techniques in regard to creative control and ornamentation seems almost limitless and is further discussed as outlook of the chapter (Section 3.5 Machine Learning).

Summary

Procedural Models

- Stochastic: maps of random values
- Functions and Rules: partition and fill a space
- Grammars: substitution system of rules and elements
- Simulation: approximation of complex generation systems or phenomena
- Artificial Intelligence: planning, adaptation and optimization

3.3.2 Data-Driven Models

In contrast to procedural techniques, data-driven methods can be used in two ways in the context of ornamentation. First, they describe the processing of input pixel data, such as a photograph. Second, they refer to the output of a method, which is again pixel data. Data-driven models traditionally do not include underlying design models, as procedural representations do. Consequently, data-driven approaches are flexible in terms of possible designs and can achieve photorealism by processing real photographs.

At the same time, photographs bring the disadvantage of potentially including visual features, such as illumination effects, which are unwanted and difficult to remove. Moreover, further down a production pipeline, pixel data is usually not editable anymore. Working with data such as high-resolution images leads to high memory requirements, and without additional algorithms, data is fixed to its given resolution and scale.

Addressing the issue of resolution example-based synthesis is a well established field of research and aims to create infinite amounts of pixel data based on a given exemplar. The pyramid-based texture synthesis of Heeger and Bergen [1995] is an early famous example. Wei et al. [2009] present a comprehensive summary of such example-based texture synthesis techniques, discussing statistical feature matching, neighborhood matching, patch-based and optimization methods. Overall, example-based methods for texture synthesis have achieved similar results in data size, random accessibility and editing and resolution options as procedural textures – but only within specialized contexts and not in an unified manner. Procedural textures offer these capabilities as inherent and combined characteristics.
Data-driven models are numerous and diverse because they can use and produce any input and output data without an underlying procedural model. Their classification is out of scope of this work. However, we do include in the following various techniques that offer further meaningful control mechanisms in the context of ornamentation. These techniques include the tiling and distribution of elements and drawing and brush mechanisms.

3.3.3 Ornamental

For a review of models that output ornamental patterns, we focus on the output of the models and not on their underlying generation principles in order to come to a summarization. Work on the generation of ornamental patterns is vastly spread over various research communities in an isolated and sparse manner [Whitehead 2010].

For development of models Whitehead [2010] differentiates between two motivations. First, he identifies the goal to reproduce existing patterns such as Islamic and Celtic designs. Such work is mainly found in the communities of mathematics and computer science. Second, Whitehead identifies the goal of generating novel pattern designs, which is held mainly by algorithmic computer artists. Such designs are usually not executed in an academic context and, beyond the presentation of the results, are unfortunately not well documented. Only a few exceptions, such as the work of Takayama [2016] in regard to 3D-printed ornate shapes, stand out, and we do not further investigate computer artists’ work.

In the following, we summarize and extend the analysis of Whitehead [2010] regarding what he calls mathematic/scientific ornamentation. Although the work included in Whitehead [2010]’s survey does not necessarily follow ornamental design rules, all examples constitute at least a subpart of an ornament – for example, the background fillings. These publications focus on the analysis of a pattern and the development of a generative model for a specific pattern type rather than its controllability. Hence, we do not include these in the taxonomy of control mechanisms.

Rigid design rules enable formal models and Whitehead [2010] describes tiling and symmetry as the most relevant constituting rules. In combination with interlacing parts of the pattern while repeating and tiling elements, these principles are able to systematically describe Islamic [Ostromoukhov 1998] and Celtic [Cromwell 1993] patterns (Figure 3.3).

The seminal work of Kaplan and Salesin [2004] presents an algorithmic representation of Islamic star patterns, a topic still of appeal [Khamjane and Benslimane 2018]. Readers further interested in this line of work are referred to the dissertation of Kaplan [2002]. Etemad et al. [2008] and Hamekasi and Samavati [2012] also focus on Islamic flower patterns. Further work (e.g., [Wong et al. 1998; Chen et al. 2012; Zehnder et al. 2016]) producing similar aesthetics in combination with offering control mechanisms are discussed in detail in the Section 2.2 Taxonomy of Control Mechanisms. Celtic designs were, for example, successfully computed by Kaplan and Cohen [2003] and Doyle and Semwal [2013].
In addition to Islamic and Celtic designs, a variety of other pattern designs have been algorithmically formalized, such as Gothic window tracery [Havemann and Fellner 2004], M. C. Escher patterns [Dunham et al. 1981; Kaplan and Salesin 2004], woodwork [Gulati et al. 2010, 2012], optical illusions [Chi et al. 2014], and general patterns [Ouyang et al. 2015; Gdawiec 2017].

Summary

In order to produce the complexity of ornamental designs that include not only repetitive structures but also visual hierarchies and highlights, most procedural models in their “pure” form are too limited in their design variability. Data-driven models are more flexible in both their design space and creative control mechanisms. However, data-driven models are also tedious and not suitable for automatically executing certain design rules. The following taxonomy gives a detailed comparison of the capabilities for the creative control of the current state of the art.

3.4 Analysis of the State of the Art

The analysis framework was executed in regard to creative control for ornamentation. Creative means are required to solve the complex task of ornamentation. It is important to note that in the following, we do not analyze the specific aesthetic value of an ornament except with regard to its coherence with the defining design principles.

For the context of procedural ornamentation, we further specify the categories of the creative means where needed:

- **Configuration:** If a technique requires the supply of a procedural model, this is counted as global configuration step. However, for techniques, which integrate a novel procedural model as part of the system, the supply of a procedural is not counted as a configuration requirement.
We counted both, configuration and initialization as control mechanisms, adding to the creative mean of \textit{Control Quantity}. However, these controls are mainly given as values within a configuration file or in the code itself as Table 2.1 shows. Whether such setup requirements add to or rather hinder creative creation and whether they should be counted as control mechanism is debatable.

- \textbf{Shapes:} It is the nature of procedural generation to automatically fill a space upon execution. Hence, for all procedural systems the \textit{shapes} category is counted even though this control might not be mentioned in a publication. If a publication does not specify how the shape to be filled is provided, we count the control as code given by a file.

- \textbf{Interactive Performance:} A technique might give a broad range for the performance and more complicated cases might not be computed with the interactive timings according to our rankings. However, if the approach includes results achieved with timings within our ranges, we do count the performance as \textit{weak} as the potential for interactivity is given. It would be worthwhile to investigate a more precise ranking that is able to reflect on visual complexity as well. The visual quality of the results should be related to the presented timings for each technique. The results with interactive performance might visually be too simple for actual use in a design context.

If a publication considers its work as interactive (often without giving specific timings), we count it as interactive. If there is reasonable doubt about the actual performance we address it in the analysis text.

- \textbf{Size of the Design Space:} As design types we apply the the texture classes of Lin et al. [2006] in combination with the design principles of ornamentation (Section 3.2), namely \textit{discrete element distribution, hierarchal compositions, adaptation to the space} and \textit{visual accents}.

For a more refined categorization of patterns, also the classification of Cimpoi et al. [2014] could be applied. However, we leave an analysis in regard to the author’s 47 classes to future work.

\section*{3.4.1 Texturing Methods}

Texturing methods focus on creating a repetitive and homogeneous pattern as automatically as possible. These methods provide only a fraction of the controllability needed for ornamentation. They are solely applicable for the subparts of an ornament with a texture-like quality to it, such as background regions and fillings.

But as the investigation of procedural texturing has been the driving force behind the development of procedural representations in general, it produced manifold approaches and noteworthy control mechanisms. Even though texturing methods are not one-to-one transferrable to ornamentation, their rich research history and solutions must be included when investigating procedural ornamentation and might inspire ornamentation specific controllability.
Example-Based Control

Example-based approaches compute a separate output based on a given example and provide a goal-oriented control. The motivation behind using these techniques is mainly to generate a specific and predictable output as efficiently as possible. Example-based and inverse approaches have a long history in the control of procedural representations. They remain a dominant research field and are relevant for any discussion about controlling procedural models. In this context, the control of a model often directly derives from a new model definition, and the focus of the related work is usually the latter. In regard to creative control, example-based approaches detach the design task to a data-driven image generation techniques, such as taking a photograph or designing a sample in an application such as Adobe Photoshop or Illustrator.

Relevant factors for differentiating example-based techniques are the size of the design space, hence their expressiveness, performance and initialization requirements. The following investigation is roughly sorted by increasing expressiveness.

Stochastic Textures For procedural texture generation, stochastic textures have been the foundation of both research investigations and many complex models. Stochastic textures are generated with noise functions, and Lagae et al. [2010a] present the state of the art for work before the year 2010. In terms of the controllability of the textures, the authors identify three main approaches. First, the indirect access to the noise through the control of the power spectrum. Second, the direct access to its appearance through function parameters, and third, example-based techniques. The first two approaches are based on specific function characteristics and are hardly generalizable for decorative pattern design. In the context of ornamentation, noise functions are seldom used in their initial state but more often as a basis for pattern design. We do not investigate the specific noise function parameters further but only include the overall example-based control mechanism. In addition to performance and input requirements as common characteristics, the expressiveness of stochastic textures can be split into representations that approximate a Gaussian texture and textures including global structures [Galerne et al. 2017; Lagae et al. 2010a].

For Gaussian-like textures, the input analysis and function parameter derivation methods are specific to the targeted noise functions. The method of Lagae et al. [2010b] matches noise bandwidths for isotropic multi-resolution noise with the performance described as “rapid”, given by Gilet et al. [2012b] as a few milliseconds. In addition to the cropped exemplar, no artist input is required. Galerne et al. [2012b] present a bandwidth-quantized Gabor noise matched by estimating the power spectrum of the exemplar through its decomposition into a sparse sum of Gaussians. Their fitting performance is about 2 minutes per texture with no input in addition to the exemplar. The noise can be further adjusted with an interactive visual editor in which the power spectrum of the noise is represented by individually modifiable sets of Gaussians. Layers can be rotated, scaled, translated and cloned. Due to the abstract nature of the visual features of a power spectrum (which is used in the editor) and the for artists not directly intuitive connection between a power spectrum and the visual features of the noise, the editor has a strong explorative
nature to it. However, as the editing itself is interactive and visually appealing, it is inviting to do so. Recently, the same authors [Galerne et al. 2017] introduced an efficient sparse convolution noise based on textons. A texton, which is a bilinearly interpolated function, summarizes the power spectrum of a given texture exemplar, and the final noise is generated by placing and summing up textons. The example match takes a couple of seconds, and no further artist input is required.

By introducing a noise that permits for the approximation of arbitrary spectral energy distributions, Gilet et al. [2012b] increase the expressiveness of their model toward more structural texture designs. For a straightforward noise by example computation, the method of Gilet et al. [2012b] successively decomposes noise frequencies to match the power spectrum of a multiple kernels noise, and, depending on the number of artist-defined convolution noises, it takes up to 20 seconds. For greater control and expressiveness, a perturbation function and a multi-layer approach are presented. The perturbation can be an additional artist-defined image map of the desired pattern, showing how the pattern should be repeated, breaking the regularity and possibly creating a more structural pattern. How the magnitude of the perturbation is defined is not described by the authors, but it could easily be an input parameter. Furthermore, Gilet et al. [2012b] interpret texture design as hierarchical composition and employ a layer function that assigns positions to different textures. For this function a artist-defined map can be used.

Further pursuing the topic of greater expressiveness and a more structured noise, Gilet et al. [2014] introduced a local random phase noise. The key aspect of their approach is the separation of structure and noise. The noise function itself blends a sum of cosines with random phase, locally centered on a regular spatial grid. A artist-controlled parameter relates to the number of cosines and the visual quality of the noise. Examples of Gaussian patterns can be given by a spectrum or a discrete noise image. For structured designs, specific phases in the power spectrum are fixed independently from the spatial domain. The amount of structure in comparison to noise is controlled with a parameter by the artist. The authors do not report performance times for the matching step. Pavie et al. [2016] also focus on extending the expressiveness of noise-based representations. The authors argue for control mechanisms being more intuitive in the spatial domain instead of the commonly used editing of the power spectrum. Local random phase noise [Gilet et al. 2014] is extended by aligning the noise on a regular grid with a spot noise model based on a random distribution of structured kernels. The artist has interactive control of the spatial structures by modifying the spot functions and their distribution, thus increasing the range of possible designs.

Guingo et al. [2017] base their work on an underlying novel noise model and a separate handling of structures with a bilayered setup. Their method improves spatial variation and visual quality in comparison to other methods. A spatially varying Gaussian noise can be matched to a suitable exemplar by computing a structure layer, extracted by filtering the exemplar, blending masks derived from clustering the spectral domain and different spectra from an auto-correlation technique. In order to procedurally represent the discrete structure and mask layers, a method based on tiling is applied. In order to control the synthesis, the artist needs to adjust two parameters, the number of different random patterns in the input and the size
of the local spectra weighting faithfulness to spatial variability of the exemplar. The performance of matching a 512 × 512 input image can take up to 1 hour (with the current implementation not parallelized).

Kang and Han [2017] decompose the power spectrum of an input image into so-called “feature” and “non-feature” parts. Non-features are obtained by a noise-by-example method. The authors do not mention whether the noise can be further adjusted. Feature parts, such as edges, can be edited in the feature image and are combined with the noise based on a artist-controlled ratio. For the procedural representation of the feature parts, the authors employ data-driven tiling. The feature extraction for a 257 × 257 input image, and therefore the texture matching, ranges from few seconds to 2 minutes, depending on an additional frequency clustering exploiting spatial coherences in the input.

Gilet and Dischler [2010] apply a more general optimization strategy for choosing the parameters of a noise-based procedure. They minimize an image distance metric computed with a multi-resolution Gabor filter bank and a windowed Fourier transform with gradient descent. With the help of the artist estimating the light source direction in the input, Gilet and Dischler [2010] can create displacement map textures, with the parameter computation taking from 1 to 3 hours. With a given rough approximation of the geometry and choosing a representative pattern patch in the input, even volumetric representations can be created from the exemplar.

Unrestricted Texture Designs All the above discussed noise-based methods control a single stochastic procedural model. Even though recent advances greatly increase their expressiveness, the design space of noise-based models is too limited for ornamental patterns. In addition to methods dealing with generalizable stochastic models, methods employ procedural textures optimized for specific design goals. These textures can potentially be visually more complex to meet the requirements of their intended task. For brick and wood textures, the early work of Lefebvre and Poulin [2000] presents an example-based control by transferring specific measured properties of an input to corresponding parameters for the procedural representation. The algorithm takes a suitable reference image, a binary mask, and the texture class as input and produces results for these two structural texture types. The authors describe the matching performance from a few minutes up to an hour.

[Gilet et al. 2012a] focus on the interactive creation of procedural semi-structured texture models. We include their work in this discussion because it also handles the control of visual features. With an improved point distribution function that can consider hierarchical spatial relationships, random variations of statistical shape models are generated from artist input. In order to do so, an artist needs to give multiple exemplary object distributions.

Bourque and Dudek [2004] allow for the whole procedural texture spectrum with their parameter retrieval technique. In so doing, they employ two types of similarity metrics, one based on the Fourier transform of the images and the other one utilizing histograms of the Laplace pyramid of the images. For optimization, they apply the Nelder-Mead and the gradient descent method. As input, an artist needs to individually select the distance metric and optimization strategy for each fitting
task. As initialization for the optimization, the authors propose “on the order of 200” pre-computed random choices to choose from. The authors report an average optimization time of 12 minutes, not specifying for how many parameters. Gilet et al. [2012b] report more than an hour for the performance times. For such a search-based approach, the parameter count is highly influential on the performance for both visual quality and computation time. With a higher number of parameters the current form of the approach quickly becomes unfeasible.

Element Arrangements An example-based control can be used to arrange elements, which refer to individual visual entities that are the smallest unit for these techniques. From an example arrangements, relationships between elements are extracted, and results are reproduced for the synthesis. Arrangements are often function-based distributions and hence can be considered rule-based procedural models, while the elements themselves usually come from input data, such as vector files. Because many ornaments contain areas of formally arranged elements, this is a relevant sub-goal for designing an ornament.

Barla et al. [2006] and Hurtut et al. [2009] focus on example-based element arrangements of stroke-based vector elements. Barla et al. [2006] map vector data to an intermediate representation based on proximity and continuation, which the authors call clusters of strokes. To synthesize a similar arrangement, elements are transferred by local neighborhood matching to a global seed distribution computed by Lloyd relaxation. Computing arrangements takes up to 10 seconds, and artist-input is used in addition to the stroke patterns. A choice between two modes for processing strokes and the amount of variation added is a post-processing step. Hurtut et al. [2009] extend that work by categorizing elements as appearance units and transferring their spatial statistical interactions to new arrangements in the order of seconds, also being able to capture non-uniform distributions. As a possible artist input, one exemplary shape input and density map are shown, and other input options are discussed in principle. The authors clearly state their focus to be on automation.

Ijiri et al. [2008] analyze a given element distribution by local neighborhood comparisons and synthesize output with interactive performance with incremental rule-based local growth. Hence, the technique combines data-driven texture synthesis with procedural generation. Element attributes that go beyond the positions of the elements and orientation cannot be controlled. Artists can choose between three element orientation modes, and as a global design constraint, artists can use an interactive spray tool to define areas to grow in, a flow field tool to define overall alignments and a boundary tool. Moreover, the reconstructed topology can manually be adjusted. The combination of tools that allow the artist to work on the canvas support the immersion in the creative tasks because an artist can think less about abstract setups and instead focus on the actual output.

The technique of Ma et al. [2011] is based on a sample of a discrete element distribution and an output shape to fill both in two and three dimensions. The exemplar has to contain the actual elements in their domain and cannot be basic pixel data. In its broadest sense, this underlying distribution model can be seen as a procedural model. Even though there are no generative rules, characteristics of the discrete elements and their distribution can be parametrized, and changes can be automatically processed
and reproduced in the output. In order to fill the output shape with elements, an energy optimization is processed with a novel neighborhood similarity metric. In addition to element positions, the metric includes variable features referring to orientation, geometry, appearance and type, for example. Hence, the metric is capable of reproducing global aggregate distributions that go beyond local element placements. The authors also extended their work to the spatial-temporal domain [Ma et al. 2013]. In regard to the available control mechanisms for artists, necessary inputs are the exemplary element distribution, the neighborhood size to consider and the output shape. Further distribution constraints based on element attributes are optional. Examples for the inclusion of a vector field and element drag and drop are given. The authors report seconds to minutes for performance times with a non-optimized implementation.

Grammar Generation

Grammars are a classical procedural representation. Grammar-based output can be designed through the generating grammar, the included visual elements and often custom-made parameters for visual features. To translate a desired visual output to an abstract grammar is a daunting task for most artists, and first efforts have been made to provide an example-based technique for the grammar generation itself. Št’ava et al. [2010] present a context-free L-System that is able to recreate a given two-dimensional vector image consisting of groups of line segments. The algorithm creates similarity groups of these basic elements, computes spatial relationship clusters and iteratively translates these into rules. An artist is required to define a similarity threshold and significance weights for the different clusters, such as element distance or similarity, for example, thus guiding their representation according to the L-system rules. The time needed for the inverse step, depending on the number of elements in the input, is reported to range from a few seconds up to 20 minutes. Talton et al. [2012] further generalize the idea of inverse grammar generation and interpret it as a probabilistic interference problem. Their system induces a probabilistic formal grammar from a hierarchy of labeled components with a Bayesian model merging technique.

Summary

The investigation of example-based techniques shows valuable achievements for goal-oriented control and for increasing design spaces within specific contexts. With regard to creative control, in addition to the gain in *variability* being a crucial step, the presented work also improves *navigability* through interactive performances.

Element arrangements potentially enable greater visual variation because they do not need to adhere to any rule formulation. At the same time, the generation of a sample arrangement, usually done in an external application, is potentially tedious. A sample that is too small might lead to uniform results. Moreover, elements are often carefully connected in ornamentation, and ornaments include a hierarchy of structures. Hence, element arrangements can only provide a subset – albeit an important one – of the design space needed for ornamentation.
Table 3.1: Interaction means for example-based techniques.

<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CONFIG.</td>
<td>INITIALIZATION</td>
<td>IMAGE</td>
<td>ARRANGEMENT</td>
<td>ELEMENT</td>
<td>VISUAL OUTPUT</td>
<td>SYSTEM</td>
</tr>
<tr>
<td><strong>Stochastic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lagae et al. [2010b]</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galerne et al. [2012b]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galerne et al. [2017]</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2012b]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2014]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pavie et al. [2016]</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guingo et al. [2017]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kang and Han [2017]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet and Dischler [2010]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Unrestricted Designs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lefebvre and Poulin [2000]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2012a]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bourque and Dudek [2004]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Element Arrangements</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barla et al. [2006]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hurtut et al. [2009]</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ijiri et al. [2008]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Ma et al. [2011]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Grammar Generation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Št’ava et al. [2010]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Talton et al. [2012]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The related work is overall uniform in working toward the classical requirements of finding the most efficient goal-oriented control as Table 3.2 shows. With the exception of Ijiri et al. [2008] and Galerne et al. [2012b] little effort has been made towards improving visual control for an artist. Ma et al. [2013] present various powerful control functionalities but do not show their capabilities within an artist usable scenario. Gilet et al. [2012a] also offer comparatively more mechanisms but as required configuration for their computation not necessarily as variable controllability.

Even if they are example-based, many techniques still require considerable non-creative effort for an artist, such as working with a power spectrum or predicting how changes in the exemplar, such as element arrangements, affect the output. The potential of these methods for creative control of ornamentation lies in furthering interactive performance, reducing initialization requirements and experimenting with the spatial influence of controls. The presented work only focuses on global designs, such as the whole canvas and repeating regions. Methods for which regions could be defined, models layered or the placement of single elements integrated constitute valuable directions for ornamentation.
Table 3.2: Creative means for example-based techniques. Please note that performance times consider the whole creation processes for an artist, including the example matching and possible parameter interaction (for noise generation techniques usually only the noise evaluation times are considered).

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVID[Interactive]</th>
<th>NAVI[Control]</th>
<th>NAVI[Quantity]</th>
<th>NAVI[Navigation]</th>
<th>NAVI[History]</th>
<th>NAVID[Control]</th>
<th>NAVID[Domain]</th>
<th>NAVI[Communication]</th>
<th>NAVI[Design]</th>
<th>NAVI[Space]</th>
<th>NAVI[Size]</th>
<th>NAVI[Design]</th>
<th>NAVI[Space]</th>
<th>NAVI[Openness]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lagae et al. [2010b]</td>
<td>•</td>
<td></td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galerne et al. [2012b]</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galerne et al. [2017]</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2012b]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2014]</td>
<td>?</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pavie et al. [2016]</td>
<td>□</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guingo et al. [2017]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kang and Han [2017]</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet and Dischler [2010]</td>
<td>o</td>
<td></td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unrestricted Designs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lefebvre and Poulin [2000]</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gilet et al. [2012a]</td>
<td>□</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bourque and Dudek [2004]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element Arrangements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barla et al. [2006]</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hurtut et al. [2009]</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ijiri et al. [2008]</td>
<td>□</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ma et al. [2011]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grammar Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Št’ava et al. [2010]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Talton et al. [2012]</td>
<td>o</td>
<td>o</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.4.2 Shapes and Masks

The most basic control requirement is to define an area to be filled. Methods that focus on the development of novel underlying procedural systems with no acknowledgment of control mechanisms, also need to define the space to fill and a relationship of the system to that space. Many approaches take the idea of simply outlining a space further and carefully design growth constraints, offer masking and the sketching of areas to be filled, thus leading to complex designs.

The following section discusses procedural techniques that consider global shapes and masks. One group of methods transform growth constraints to a probabilistic inference problem. Due to the decorative quality of the results and their parameterized control, a data-driven approach is also included.
Rule- and Grammar-Based Methods

Wong et al. [1998] introduced a programmable procedural system that employs a greedy rule-based strategy to generate floral ornaments. A procedural model is created with artist-defined elements and with a set of growth rules that handle the selection, appearance and connections of elements. The process iterates, finding tentative places for elements by testing them against constraints in the procedural model and, where suitable, placing elements in the found spaces, optionally and connecting them to existing elements. Possible ornament designs are technically restricted only by this iterative creation logic. All adjustments to the design and layout of an ornament have to be done by writing code, with the exception that a “region specification” for the filling can be given. The authors do not report any performance times.

Santoni and Pellacini [2016] present the procedural generation of tangles, which are repetitive black-and-white hand-drawn patterns made from dots, straight lines, simple curves and circles. Tangle elements usually align to the shape they fill, for example, by outlining it. A stochastic group grammar with grouping, geometric and decorative operators composites recursive patterns at different scales, filling two-dimensional shapes as well as handling holes. A tangle generation usually takes a few seconds, with a complex example taking about 3 minutes. The authors demonstrate the applicability of their method with an interactive system based on a parameterized artist interface, including history navigation, rule re-expansion and sketch-based operator modification. A user study evaluates the system as accurate, controllable and easy to use after a reasonable training time.

Loi et al. [2017] present a procedural framework for a large variety of element texture designs. The authors aim for designs that are unrelated to their spatial location and the space they fill, calling it stationary. Their programmable method is developed for technical artists and requires programming expertise. Generating pattern scripts are built with partitioning, mapping and merging operators. These operators enable both global and local design control and the composition of designs. The operator-based technique would enable a node-based interface design, which is not explicitly demonstrated in the article. The execution time for most designs is a few seconds, with some examples taking more than 1 minute. A user study with technical artists carefully evaluates the system’s scripting interface, concluding positive results overall.

Beneš et al. [2011] offer a complex shape-filling and masking system for procedural open L-system models by dividing a target space into artist editable guide shapes. Seeds for the L-system are interactively given by an artist as a position and orientation. The guide shapes determine what types of patterns grow in different areas. The connections between the shapes are manually specified by the artist and in turn guide the connections between elements. Based on a mass-spring system, the guides can be intuitively edited as a whole. The authors report on pattern generation performance for most scenarios as less than a second, with up to 45 seconds for only one complex scenario.
Probabilistic Interference Other systems provide global outlining shape control on procedural processes by interpreting the modeling task as a probabilistic inference problem.

Talton et al. [2011] present for grammar-based procedural models, as example for their flexible analytic objective functions, non code-based global controls through image and volume matching. The authors stress that in principle any control mechanism can be matched with any grammar through their decoupling of the growth control from the grammar itself. The authors discuss that to come to the desired design goal, some experimentation might be needed, making the approach less transparent. Performance depends on the complexity of the grammar and the number of optimization steps needed. The authors report performance times ranging from a few seconds to several hours. For their examples, the authors manually terminated the optimization iteration.

Ritchie et al. [2015] controlled rule-based hierarchical and iterative procedural models similar to Talton et al. [2011] with image-based matching and target volumes. The authors present a sequential Monte Carlo variant that is able to score incomplete model states, thus improving convergence behavior and final scores. The reported performances range from around 3 seconds to 12 minutes, and the authors show that the number of included primitives scales reasonably.

Data-Driven Fillings

For filigrees, which are thinly structured repetitive patterns, Chen et al. [2016b] present a mainly data-driven approach. Their method automatically distributes and assembles a set of suitable independent input elements for which an up vector is specified into a pattern in both 2D and 3D. The authors implement an optimization of a packing problem under specific constraints, mechanically creating strengthened fillings. This method does not rely on an underlying procedural model, but it also processes control parameters for the filigree generation. Due to the nature of the optimizations, a randomization and a distortion ratio parameter are required input. Additionally a field of directional strokes can be drawn on the canvas, controlling element orientation and size. When multiple elements are combined into one common pattern, percentages for appearances of the elements can be given. The performance in two-dimensional space runs from 6 to 26 seconds.

Summary

The above discussed procedural generation techniques offer novel systems that decouple control mechanisms from the implementation of individual models. This enables more possible results for one specific technique, thus improving the size of design spaces, as Table 3.3 shows. Sophisticated masks and growth constraints lead to visually interesting and complex designs. However, it is not directly predictable how a space will be filled exactly. Because most of the presented methods only offer quite limited interactive performance, even a basic trail and error exploration is hardly feasible; hence, the navigation of the design space becomes cumbersome, and stimulation becomes hindered. The one technique [Santoni and Pellacini 2016] that offers the means for a transparent navigation is also the one with the most restricted
Table 3.3: Interaction and creative means for shape-filling methods.

<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CONFIGURATION</td>
<td>INITIALIZATION</td>
<td>IMAGE</td>
<td>ARRANGEMENT</td>
<td>ELEMENT</td>
<td>VISUAL OUTPUT</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Wong et al. [1998]</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Santoni and Pellacini [2016]</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loi et al. [2017]</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beneš et al. [2011]</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Talton et al. [2011]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ritchie et al. [2015]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chen et al. [2016b]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVL.</th>
<th>TRANS.</th>
<th>VARI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive</td>
<td>Control</td>
<td>Quantity</td>
<td>Navigation History</td>
</tr>
<tr>
<td>Wong et al. [1998]</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Santoni and Pellacini [2016]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Loi et al. [2017]</td>
<td>o</td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>Beneš et al. [2011]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Talton et al. [2011]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Ritchie et al. [2015]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Chen et al. [2016b]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

design space. Santoni and Pellacini [2016]’s consideration of a navigation history stands out from all related work in this survey. In terms of stimuli, the mass-spring system for editing control guides offered by Beneš et al. [2011] is a promising direction because it is intuitive, enjoyable to use and encourages exploration.

In terms of control mechanisms for a decorative design goal, these techniques do not permit hierarchical or element-level local controls or the control of element connections needed by artists who want to use ornamentation without having to write code.

### 3.4.3 Vector Fields

Fields constitute a powerful tool for combining an automatic procedural filling by individually designing regions on the canvas. In the context of two-dimensional ornamentation, these fields are usually vector fields. The streamlines of a field can create curves as part of the pattern that fill and structure a space. The design of a vector field requires less manual work than the manual creation of curves. Other global design choices, such as an overall growth direction or the alignment of elements, are simple to translate from a vector field to procedural generation rules.
### Table 3.4: Interaction and creative means for techniques with vector fields as underlying control mechanism.


<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config.</td>
<td>Initial.</td>
<td>Image</td>
<td>Arrangement</td>
<td>Element</td>
<td>Visual Output</td>
<td>System</td>
</tr>
<tr>
<td>Ijiri et al. [2008]</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Li et al. [2011]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saputra et al. [2017]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVI.</th>
<th>TRANS.</th>
<th>VARI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive</td>
<td>Control History</td>
<td>Control Domain</td>
<td>Design Space Size</td>
</tr>
<tr>
<td>Control Quantity</td>
<td>Communication</td>
<td>Design Space Openness</td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ijiri et al. [2008]</td>
<td>•</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Li et al. [2011]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Saputra et al. [2017]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

Already included in the described work of Ijiri et al. [2008] above, the authors employ vector fields to define the overall growth direction and alignment of elements within an example-guided arrangement.

Li et al. [2011] present a shape grammar that is guided by either a vector or tensor field. The field can influence the grammar’s translation command, potentially leading to globally pronounced structures. The field can furthermore guide rotation, scaling, and color parameters. The artist can specify a priori field constraints, such as regular and singular field elements, on the surface to be filled. Once the field is computed, local Laplacian smoothing can be applied. The authors report a synthesis performance for geometric surfaces from less than a second up to 3 minutes.

Saputra et al. [2017] optimize a flow-based ornamental packing of elements into a two-dimensional outline. For each element, a predefined spine controls the element’s deformation. The artist defines direction guides and optionally fixed elements that control the computation of evenly placed streamlines. Elements are placed and deformed along streamlines. An iterative refinement step optimizes for a dense and balanced filling. First, streamlines are slightly shifted to cohere to the space available. Second, elements are re-placed with rotational adjustments and possible overlaps into free space of neighboring elements, reducing negative space. An average packing takes about an hour.

Vector fields are further employed in various other specific procedural modeling contexts. For example for procedural street modeling [Chen et al. 2008a], micrography [Maharik et al. 2011] or botanical models [Xu and Mould 2015].
Summary

The discussed work in Table 3.4 shows that fields allow for greater visual variation by opening the design space and transparent control for filling a space automatically. When designing a vector field, artists do not work with the pattern directly, but fields are intuitive to understand. Their abstraction translates to the model in a straightforward manner. Thus, using flow within a vector field to design is a suitable control mechanism, especially for ornamentation and its characteristic element alignments to the space.

3.4.4 Curves, Sketches and Painting

Curves and hand-drawn paths give an artist more direct control than the previously discussed methods to fill a space. In addition to the visual output being further constrained, the control is put onto the actual canvas. Curves are needed for tasks such as creating an ornamental frame or structuring the space. Some techniques consider the whole curve before computing the ornament, optimizing the filling of the curve based on certain design goals, enabling a form of global planning.

Painting-tool-like methods create output along curves but do so directly without taking an a priori completed curve into consideration, as if using a spray can or a brush. Painting techniques usually include a brush diameter, hence the size of the area to be filled along the curve.

Besides the value of the indirect use of curves as a control tool, their direct employment as a visual element is also relevant. Formed curves, such as circles, spirals or hearts, are essential components for ornamentation.

The following section first discusses work that enables the direct control of curves as pattern elements. The state of the art using curves as control mechanisms are then discussed for both procedural and data-driven methods.

Curves As Basis Elements

Anderson and Wood [2008] adapted the design principles for ornamentation discussed by Wong et al. [1998] as well as their core growth mechanism of “finding the largest space to fill next.” Anderson and Wood [2008]s’ technique places discrete elements on the sides of an artist-given curve, while not filling the curve itself. The artist can input masks not to be filled, proxies controlling the size and type of elements to be placed and to equal the sum of radii on both sides of the curve. Two input interfaces exist, the interactive view and the buffer view. The authors do not report a user study or specific performance times but call their system interactive.

Also incorporating an artist-defined curve as the spine of a pattern, Chen et al. [2012] use an interactive L-system to attach decorative spiral designs to the curve given by an artist. Xu and Mould [2009] use the space-filling algorithm of Wong et al. [1998] in combination with particle tracing in simulated magnetic forces for the generation of decorative curves. The physical properties of the charges, the magnetic field and the initialization of the particles are the parameters for designing the curves. The computation takes less than 5 seconds. The authors acknowledge the non-intuitive
parameterization of the system and give an example timing of 2 minutes for finding the parameters of a specific example. Merrell and Manocha [2010] generated a set of curves in the same style of a given parametric example curve. A style is defined by local properties, such as tangents and curvatures that are derived from a local shape analysis. The new curves are computed with a rule-based system that allows artists to interactively edit the result. Interactivity is somewhat diminished by computation times of a few minutes for a curve set. Zehnder et al. [2016] provide artists with a tool to directly assemble structurally sound curve networks on a three-dimensional surface. The components of the network are spline curves defined by the artist. Components can be placed manually or are repeated semi-automatically. The curves can be moved on the surface while having an elastic quality to them. To prevent structural weaknesses, the system indicates problematic areas and suggests improvements, seamlessly combining the design task with engineering requirements.

**Curves As Control Mechanism**

Méch and Miller [2012] present examples of painting methods for different aspects of generating procedural models, from painting growth constraints, such as masks, to having a pattern grow along the strokes. This discussion only refers to the actual examples given by the authors. However, these are only selective examples for the flexible Deco procedural engine. The engine opens up and generalizes environments for interactive control mechanisms for various types of procedural models. For the programming of decorative pattern models within the engine, helpful functionalities, such as symmetry objects and control guides, are predefined. All artist control mechanisms have an interactive performance. Overall performance mainly depends on the pattern generation scripts. The engine offers to load pattern codes as a dynamic library, optimizing performance. In theory, the Deco engine could allow for the editing both of single elements and their connections. This is crucial for decorative patterns, for example, in setting visual highlights. In Méch and Miller [2012] however, no examples for this feature are given.

Jacobs et al. [2018] developed the programming and drawing environment Dynamic Brushes, in which an artist can create individual procedural brushes for a stylus pen. General programming logic and relevant mathematical functions for creating patterns are translated into a visual programming interface. The evaluation of the system by two professional artists shows that once initial struggles to learn the system were mastered, the artists were able capture their personal analog styles with the procedural brushes. Overall, the authors and the artists open many valuable questions about the usage of current tools and about alternative approaches that seek to seamlessly blend manual and procedural creation processes.

More painting-like methods can be found, for example, in procedural botanical modeling [Anastacio et al. 2008; Chen et al. 2008b; Palubicki et al. 2009], procedural landscape generation [Emilien et al. 2015], as part of a procedural water color engine [DiVerdi et al. 2013] or for dynamic effects [Xing et al. 2016].

**Data-Driven Approaches** In order to create an ornament along a sketch, Lu et al. [2014] present a data-driven approach. Given vector pattern exemplars are placed
and deformed along a artist-given curve. Boundaries between element segments and visual soundness are optimized through graph cut and hierarchical texture synthesis. For the exemplars, an artist has to define the start and end point of their spines. If needed, the whole spine can be sketched as an input. The artist can refine results with add and erase constraints that are drawn on the pattern. The authors report a synthesizing performance from 1 to 8 seconds. A related data-driven approach for synthesizing example-based vector patterns along a curve was presented by Zhou et al. [2014] in the same year. In this work, the authors focus on ensuring a structurally sound output pattern and an extension to fill a surface. Topology descriptors and artist-given topological constraints are included in the element assembling optimization process. Additionally, local pattern orientations and a variation value can be defined by an artist. Once a pattern is generated, an artist can interactively adjust the underlying curve, with the pattern being updated accordingly. Generation performances are reported to be around a few seconds, with complex models a little more than 2 minutes.

Kazi et al. [2012] present a multifaceted tool to create textures from pen-and-ink drawings with sketch-based control mechanisms, mixing data-driven and procedural modeling. Basis drawings can be repeated along paths, used for brushes, fill regions, optionally consider perspective and propagate modifications of the drawing to all repeated elements. A user study confirms the system’s usefulness to efficiently create repetitive textures while maintaining the natural workflow and artistic control of an artist. Xing et al. [2014] build upon that work by automatically detecting and suggesting possible repetitions to the artist, aiming for a less regular, more painting-like quality. The presented system also offers various brush options and navigation tools in order to combine automation with artist control.

Similar approaches have also been investigated in the context of texture painting [Lukáč et al. 2013], creating mosaics [Igarashi 2010; Abdrashitov et al. 2014] and data visualization [Xia et al. 2018]. These ideas cohere to the needed control principles for the creation of ornaments while focusing on their specific design tasks.

**Feature Exploration** Even though not a generating technique in itself, exploration is an important characteristic of a creative process. Todi et al. [2016] present a tool for exploring sketches and the automatic optimization of common layout types. With the method of Chen et al. [2016c], an artist can browse a collection of texture images by sketching highly abstracted pattern features. The represented structural features of reflection, rotation, and translation symmetries adhere to important design principles for ornamentation. One could imagine a similar intuitive approach for exploring the parameter space of an ornamental procedural representation.

**Summary**

Curves and sketch-like methods offer a well communicated, hence transparent navigation, as Table 3.3 shows. The discussed techniques are mostly interactive, artists are familiar with their functionality from the real world and they work directly on the canvas. The ease and directness of usage also constitute a foundation for
Table 3.5: Interaction and creative means for techniques with curves and sketches as visual elements and underlying control mechanism. *Please note that Měch and Miller [2012] present a procedural modeling engine, which in principle can be programmed to include almost any control type.

<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>As Basis Elements</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson and Wood [2008]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Chen et al. [2012]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xu and Mould [2009]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Merrell and Manocha [2010]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Zehnder et al. [2016]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td><strong>As Control Mechanism</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Měch and Miller [2012]*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Jacobs et al. [2018]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Lu et al. [2014]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zhou et al. [2014]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kazi et al. [2012]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Xing et al. [2014]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVI.</th>
<th>TRANS.</th>
<th>VARI.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>As Basis Elements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson and Wood [2008]</td>
<td>●</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Chen et al. [2012]</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xu and Mould [2009]</td>
<td>●</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Merrell and Manocha [2010]</td>
<td>●</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Zehnder et al. [2016]</td>
<td>●</td>
<td>o</td>
<td>●</td>
</tr>
<tr>
<td><strong>As Control Mechanism</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Měch and Miller [2012]</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Jacobs et al. [2018]</td>
<td>●</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Lu et al. [2014]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Zhou et al. [2014]</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Kazi et al. [2012]</td>
<td>●</td>
<td>●</td>
<td>o</td>
</tr>
<tr>
<td>Xing et al. [2014]</td>
<td>●</td>
<td>●</td>
<td>o</td>
</tr>
</tbody>
</table>
possible immersion. Using painting-like methods can allow for smoother navigation by integrating brush settings and increasing the quantity of controls.

However, because creation techniques and design spaces are open, it could lead to manual and tedious creation requirements for ornamentation, such as when filling a background. Here, the incorporation of procedural creation principles for automatic fillings into a data-driven process by Kazi et al. [2012] and Xing et al. [2014] is a promising direction. Instead of fostering a free painting-like quality, design principles for ornamentation could be added to create a more organized output.

### 3.4.5 Element Placement

The placement of single elements onto the canvas maximizes artist control and is on its own a trivial data-driven control principle. However, in combination with procedural modeling, this mechanism becomes interesting. Separately placed elements that do not follow any rules should be integrated and processed to remain part of the underlying global scene structure. Even though this functionality can be compared to using the tip of a brush, paint-like procedural modeling techniques often have a more spray-can-like quality [Měch and Miller 2012] and do not include this option.

For ornamentation, this type of variation is needed for preventing a monotonous, texture-like output. The placement of single elements as highlights should visually break the underlying order of repetition, while still being a homogeneous part of the global layout. The following work presents steps toward this demanding goal by integrating the placement of single elements into a global control mechanism.

By detecting symmetries and curvilinear element arrangements in a given vector pattern, Yeh and Měch [2009] extend the manual data-driven design processes with procedural-modeling-like editing options. Based on the detected element groups, an artist can adjust the spacing, location and scale of one element directly and propagate that change to the all other elements in the group. The authors also offer a brush that recreates recognized element groups.

The technique of Guerrero et al. [2016] offers suitable design variations of the vector pattern an artist is working on. An artist can select and continue with one of the offered alternatives. The system constantly re-selects from an exponential number of relevant variations based on the artist’s modifications. The user interface is carefully laid out in order to offer design variations in an intuitive and efficient manner while at the same time not hindering an artist’s own workflow. The authors thoroughly evaluate their system quantitatively and qualitatively – for example, with a user study. Overall, participants agreed on the usefulness of technique.

### Summary

The discussion of this section is closely related to the data-driven sketch-based techniques, and it shows a further promising approach for integrating procedural modeling functionalities into a data-driven process. As Table 3.6 shows, Guerrero et al. [2016] present a overall transparently navigable and stimulating control mechanism. With a carefully designed workflow, it further fosters an artist stimulation by offering novel but suitable design variations.
Table 3.6: Interaction and creative means for techniques that allow for the placement and modification of single elements in combination with applying procedural design functionalities.

<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeh and Měch [2009]</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guerrero et al. [2016]</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.7: Pie charts showing the relative distributions of the creative means interpretations for the discussed control mechanisms. Strong (in black in the pie chart) and weak (in gray) characteristics are individually analysed in the previous section for each publication.

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVI.</th>
<th>TRANS.</th>
<th>VARI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Quantity Navigation History</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Space Size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desired Space Openness</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Yeh and Měch [2009] • o o o o o
Guerrero et al. [2016] • o o o o o

3.5 Discussion

The summary of all control mechanism groups in Table 3.7 shows that there is no approach that fulfills all creative means at least weakly.

The employment of curves and sketching can be identified as promising creative method for the current state of the art. This confirms the general appeal of trans-
fering real-world tools, such as a pen and eraser, into the digital workspace in combination with automatization. However, this is not the only promising approach as the creative use of vector fields shows. This is a mechanism that is not directly found in analog creation processes. A particular surplus of controllability and expressiveness is generated, in contrast to mechanisms available in the analog world. Vector fields are still easy to understand through characteristics such as flows and directions, which again stem from real-world experiences. In this combination of unique digital functionality with intuitively assessable analogies lies great potential, and it should be further investigated.

The summary in Table 3.7 also shows that there has been comparatively little attention on an overall development of transparent navigation techniques. It is interesting to note that only two of the discussed publications mention a navigation history for their creation process. There has been some specialized work focusing on editing histories in a data-driven context [Hu et al. 2013; Chen et al. 2016a]. However, even though this is an essential mechanism in common digital tools and a true surplus for an efficient and creative process for artists, most past research does not consider this challenge. Reasons range from techniques where a navigation history is simply a development task and of no interest to researchers to techniques where such a history is hardly possible. Nonetheless, a careful investigation of actual capabilities and limitations for editing histories is in order.

Moreover, the above mentioned tendency of improvements to navigation and transparency that potentially reduce possible design variations is affirmed. Joint research investigations that combine the development of algorithms, control mechanisms and interfaces might be able to resolve this contradiction.

Similarly, there is always a trade-off between the different ornamental design subtasks. As of now, there is no one method capable of creating an ornament with all its characteristics, either data-driven or procedural. Research usually specializes in a specific domain, thus neglecting the challenge to unify necessary control and design aspects in an effective manner [Smelik et al. 2014].

Overall, data-driven approaches that integrate procedural modeling features seem to be the most successful in providing creative control. We believe that this is due to data-driven approaches being overall a more restrained and accessible context for the development of novel control functionalities. Even though they are powerful, the underlying algorithmic structure of procedural models make them fairly complex and thus limiting.

Novel combinations of data-driven and procedural approaches have been called for (e.g., by [Zehnder et al. 2016; Xia et al. 2018]), allowing for an efficient representation and the automation of tedious tasks while offering creative control to artists in a unified manner.

Outlook

The review of the state of the art shows that there are various limitations and possibilities for future work within the specific contexts of the work. However, there are also novel paradigms for creative control and their underlying algorithms
that uniquely add to the state of the art. The most prominent development is the integration of machine learning techniques, which are discussed in more detail in the following. More insights regarding the possibilities of the usage of semantic attributes are also given below.

In addition to machine learning and semantically driven approaches, collaboration is a valuable future line of investigation for enabling creative work. With regard to technology, more and more aspects of common tools either are fully browser-based or are in some way connected to the cloud-based storage of assets, settings and results; therefore, they function online and are easily shared. Collaboration is closely connected to the previously discussed issue of navigation histories. This is not only relevant for individual work processes but also for more general production pipelines in a commercial context. In this regard, the sharing and collaborative work on iterations, which involves multiple persons referencing different versions, is essential. Some work has been done (e.g., [Salvati et al. 2015; O’Leary et al. 2018]) but further investigations of collaboration for creative control are called for.

As discussed in Section 3.3, control techniques are closely inter-related with the representation of the underlying models. Therefore, a more unified development of models across research communities would be beneficial. Expert knowledge of usability should especially be considered. However, further automation for the creation of complex ornamental models also poses interesting challenges, such as abstraction [Nan et al. 2011], symmetry computation [Cullen and O’Sullivan 2011] and design space variations.

**Machine Learning**  To integrate a machine learning framework into a procedural system is a fairly novel development. In the context of ornamentation, a summarization of the state of the art is not yet representative. The following describes some relevant work.

Phan et al. [2016] offer a data-driven recommendation system for circular ornamentation, employing a learned style and composition feature vector. Based on a custom ring-based layout system that represents, for example, plates, vases and a first decorative element chosen by the artist, the system completes a design. The artist can also choose to incrementally add elements manually, while the system accompanies this by suggesting suitable elements and placements. This work indicates the promising direction of using learned characteristics to further stimulating tools, which, for example, generate meaningful design suggestions.

Ritchie et al. [2016] make use of machine learning to improve the performance of the image-matching grammar-based models of Ritchie et al. [2015]. The updated system increases performance up to 10 times by integrating a neural network and sampling a learned constraint-satisfying approximation. Reported performances are overall below 3 seconds. Interactive performance is the foundation of all creative control means and hence of great importance.

The procedural content generation (PCG) for games community is pushing the general integration of artificial intelligence (AI) into a procedural creation process. A new paradigm of mixed-initiative creative interfaces is rising and is actively fostered, as an ACM Conference on Human Factors in Computing Systems (CHI) workshop
under the same name in 2017 shows [Deterding et al. 2017]. As the workshop summary states, it is the goal to “put human and computer in a tight interactive loop where each suggests, produces, evaluates, modifies, and selects creative outputs in response to the other.” In order to achieve this, AI enables computer agency, and novel interfaces enable collaboration between computers and human users. The workshop brought PCG and interaction design researchers together, stressing the importance of bridging disciplines. Similarly to games, the context of ornamentation also constitutes a challenging but fruitful testing ground for the investigation of mixed-initiative creative interfaces and for the task of balancing artist control and automation, as this survey shows. In general, the involvement of the computer graphics community with its various topics and rich algorithmic knowledge would be promising.

**Semantic Attributes** The usage of semantic attributes presents a highly intuitive navigation technique, which so far has been successfully applied in the context of shape modifications, for example by Yumer et al. [2015].

In the context of ornamentation, procedural textures constitute the most related field of investigation. For the control of procedural textures, methods are based on the analysis and description of texture in regard to human perception, which has a long research tradition. In his influential work, Julesz [1981] defines textons as the basic units of pre-attentive human texture perception. Since then, this line of research has continued, and texture descriptions with perceptual [Liu et al. 2015] and semantic [Matthews et al. 2013; Cimpoi et al. 2014] attributes have been investigated. Dong et al. [2017] and Liu et al. [2018] employed such features in first experiments for the navigation of a procedural texture space and for the generation of suitable textures by given features. However, the results of such studies are still limited and of varying quality – and the authors themselves [Liu et al. 2018] call their results experimental.

Nonetheless, these works present an interesting approach that is worth further investigation. Because ornamental designs are structured and follow an internal logic, it seems feasible to come up with a collection of suitable attributes. The ornamental design space is much smaller in comparison to all “textures in the wild” [Cimpoi et al. 2014], and ornamentation could constitute a valuable context for further investigations into the incorporation of semantic attributes into a creative creation process.

### 3.6 Conclusion

In order to achieve the goal of creative control for procedural ornamentation of specific design goals, an understanding of creativity and creative means, underlying models and the identification and classification of the related state of the art are required.

Specific ornamental design goals are a perception of order through the structured repetition of elements and hierarchical compositions. Ornaments adapt to the space they fill and integrate contrasts and highlighting accents for visual appeal.
This work follows the definition of creativity as intentionally producing a novel and surprising product and identifies navigation, transparency, variation and stimulation as analyzable means for engaging in creativity.

While the focus of this work is on procedural models, various relevant data-driven approaches are integrated and work that solely specializes on ornamental designs highlighted. For a better understanding of the creative control capabilities of the state of the art, the control paradigms of how, what, where and when are analyzed and broken down into specific mechanisms with exemplars, parameterization, handling, filling, guiding and placing and their respective interactions.

Our analysis shows that current work mainly focuses on specific and separated single aspects, which can not support overall creative control for ornamentation. For more complete and meaningful solutions aspects of both, data-driven and procedural techniques are needed and must be merged to a unified whole.
Goal-oriented control for procedural pattern generation aims for the efficient generation of a predefined design. The development of such a design is separated from the control of the procedural representation and is, for example, given by a photograph as a target.

It is a common task when composing computer-generated imagery that a given visual context needs to be complemented. In the visual effects industry, digital assets must match real-world footage, and asset creation might be controlled by reference images from a set. There is still a certain degree of freedom in the design process, but above all, the digital assets need to look believable in the given context. One of the most prominent domains for the usage of procedural patterns in the industry is the development of surface appearances, specifically for creating naturalistic textures. For such textures, both the demand for visual quality as well as for efficient handling and compactness become increasingly crucial.

Historically, modeling techniques focused on one of these requirements while having trade-offs for the others. On the one hand, the procedural modeling approach surfaced, which has the most compact storage requirements but also an unmanageable parameter control at times. On the other hand, image-based modeling techniques were developed that store and render natural images and achieve great realism at the cost of expensive recording and storage and due to limited options for editing.

This chapter investigates the principle of a goal-oriented control through example-based parameter retrieval. A fusion of the two major texturing principles implements the automatic choice of the parameters of procedural texture models so as to match the appearance of an input texture image. The final control should still remain with the artist, and the system should enable an interactive performance with efficient handling.
The technique presented in this chapter enables an artist to start with an automatically retrieved feasible parameter set for a procedural model because a manual initial exploration of the parameters is tedious. The parameter space often appears uncontrollable because the parameters might behave non-linearly with overlapping effects. By automating this non-artistic effort, the artist is able to fully focus on the creative task of finalizing the look. The system seeks applicability in real-world production scenarios that require intuitive and real-time interaction mechanisms as well as robust setups that are maintainable in a generalized fashion. This means that there cannot be any assumption made regarding the patterns and examples to use; instead, the system needs to handle all types of standardized input.

| Contribution 1 | The identification of visual texture features that compose the overall impression for a human observer and the abstraction of those features as different texture descriptors. |
| Contribution 2 | A two-stage comparison of texture descriptors and optimization strategies for navigating texture patterns. On the one hand, the performance and quality of the results are evaluated numerically. On the other hand, a user study shows the alignment of the different texture descriptors with human perception of both natural and synthesized images. |
| Contribution 3 | The implementation of the optimal choices that were identified in the previous step, with a similarity measure between input images and the structural parameters of a procedural two-tone texture. The measure is based on a perceptually motivated image-distance metric calibrated to the end user of the technique. The metric works robustly within very different texture model classes, which we demonstrate for noise textures, regular grids and special-purpose texture models for tiled and wooden surfaces. |
| Contribution 4 | The interpretation of the matching parameter search as a configuration-free retrieval task, making it possible to precompute databases of texture descriptors, which in turn enables interactive performance. |

The results of the chapter show that this novel technique matches production textures (see Section 7.4 Image References, e.g., CgTextures, 3dtotal, Turbosquid) reasonably well. Preliminary discussions with members of the visual effects industry have revealed interest in this work – for instance, as a technique to support rapid pre-visualization in production. The improvement of the variability of the possible design and the novel interactive performance are also crucial steps toward more creative creation processes. Creativity requires efficient navigation. Hence, interactive performance is
mandatory as well as a design space that is flexible enough to enable results that are surprising to the artist.

4.1 Related Work

For a discussion of texture categories, their procedural representations and example-based texture synthesis, please refer to Chapter 2 Analysis Framework for Creative Control.

In the following we briefly repeat and contextualize the review of the related work for the different steps of our preliminary analysis and interactive parameter retrieval pipeline.

Texture Descriptors

Ever since Gabor [1946] showed that certain functions minimize the time-frequency uncertainty product, such two-dimensional Gabor filters are, among other applications, a popular choice as descriptors. Field [1987] showed that the responses of a Gabor filter bank, with filters that differ in orientation, frequency and width, relate to the responses of the human visual system to image elements. Manjunath and Ma [1996] introduced a robust image browsing and retrieval application based on a Gabor filter bank and a distance measure from the accumulation of the filter responses. They compared their implementation with other classification algorithms and concluded that Gabor filters show slightly better performance and retrieval accuracy. Jain and Healey [1998] implemented a similar distance measure but focus on color textures, which outperform gray-scale based retrieval applications. Gabor filters are also successfully used for texture segmentation [Jain and Farrokhnia 1990; Dunn and Higgins 1995].

A different and in terms of performance often preferable approach for texture analysis is the processing of the power spectrum of a texture. Zhou et al. [2001] described texture features with a Discrete Fourier Transform (DFT) and a local 8 pixel neighborhood search, evaluating it for retrieval and classification tasks with gray-level textures.

Burt et al. [1983] presented texture descriptors based on image pyramids. These pyramids decompose an image into its different frequency bands, each level sampled successively at a lower frequency, an approach further developed by Adelson et al. [1987].

Evaluation of Texture Descriptors for Retrieval

Ahmad et al. [2007] compared Gabor and Fourier descriptors for image retrieval tasks with 107 unique gray-level images in different orientations and level of noise contamination, giving Gabor descriptors the preference for noisy scenarios but favored in terms of performance their DFT based descriptor. Randen and Husoy [1999] evaluated a variety of filtering approaches for database applications, concluding that no single approach sticks out in terms of quality or performance.
4.1 • Related Work

As described above, previous work on distance metric evaluation was concerned with performance in natural image retrieval scenarios, while our more general problem requires high performance not only on natural images but also in the space of synthetic, procedural textures, which stem from a less constrained texture space. To our knowledge, we present the first comparative evaluation for this use case.

Example-based Texture Synthesis

One approach for example-based texture synthesis is data-driven, producing its output as an array of pixel data [Heeger and Bergen 1995; Efros and Leung 1999; Wei and Levoy 2000]. Wei et al. [2009] present a comprehensive summary of such data-driven techniques, focusing on neighborhood-based texture synthesis applications, also in regard to dynamic and solid texture synthesis. As all these techniques produce pixel data, they are fundamentally different to our approach of retrieving parameter sets for procedural textures: compact representations which can be evaluated independently per texel permitting parameteric control.

Related work on example-based procedural texture synthesis techniques is distinguishable regarding the underlying texture models, which distance metrics are employed and how its parameters are controlled. For the class of stochastic texture models, the generating parameters can be observed by computational analysis of the query images. This has been used as a parameter control strategy for stochastic textures in a variety of techniques. Lagae et al. [2010b] compute weights for the different noise bands of a multi-resolution noise to match isotropic stochastic procedural textures. Galerne et al. [2012a] automatically adjust the parameters of bandwidth-quantified Gabor noise. Gilet et al. [2012b] present a multiple kernel noise which they designed by defining the power spectral density. Successively decomposing and matching the noise frequencies enabled the creation of visually appealing procedural textures. All techniques above yield convincing results but, being based on purely stochastic textures, their expressiveness is limited.

Lefebvre and Poulin [2000] transfer measured properties of images to corresponding parameters for procedural brick and wood textures. The algorithm takes a reference image, a binary mask, and the texture class as input and produces persuasive results for these two structural texture types.

Gilet and Dischler [2010] apply a more general optimization strategy for choosing the parameters of a procedure. They minimize an image distance metric computed with a multi-resolution Gabor filter bank and a windowed Fourier transform with gradient descent. While their approach is limited to a specific class of stochastic textures, they can create volumetric representations from two-dimensional input images. Bourque and Dudek [2004] also employ distance metrics and non-linear optimization while allowing for the whole procedural texture spectrum when matching. Our work differs from theirs by reducing the search space by several dimensions: we are able to support more complex structural designs as we process a texture’s color information independently, an idea loosely based on previously implemented color space transformations [Heeger and Bergen 1995] [Vanhoey et al. 2013]. Furthermore our work is distinguished by presenting a robust and generalized pipeline for any structural texture model. In Bourque and Dudek [2004]’s work the user needs to
select the distance metric and optimization strategy for each fitting task individually. Finally, we present a larger variety of results with average matching times in less than one second during run-time due to our retrieval technique while Bourque and Dudek [2004]'s non-linear optimization cost around 12 minutes at the time of publication.

Wu et al. [2013] also implement a retrieval-based core for their inverse bi-scale material design pipeline. They split the appearance design task into a search in pre-computed small scale geometry and material libraries, employing an overall non-uniformly weighted Euclidean distance of the BRDF representations.

We base our distance metric on a more abstract appearance feature vector by implementing a Gabor distance metric.

4.2 Texture

Please refer to Section 3.1 Terminology for our understanding of texture and to Section 3.3 Models for a summary of procedural texture functions.

Textures can be understood as a two-dimensional function $T$. In the following, they are stored as finite and discrete quantities – namely as three-channel digital color images that represent a finite number of texture values over a integer grid. They are formally modeled as

$$T : \mathbb{R}^2 \to \mathbb{R}^3, \quad \vec{x} \mapsto (r, g, b)^T$$

(4.1)

with $\vec{x} \in \mathbb{R}^2$ as the grid coordinates, also commonly written out as $(x, y)$, and RGB as color space. For a gray level image, the texture value is a scalar.

Most publications about example-based texture synthesis focus on one specific texture type, such as stochastic texture models. For this work, there are no general restrictions on the structural spectrum of the textures.

4.3 Texture Descriptors

To find a parameter set of a procedural texture that matches the image input, the similarity between the rendering of the procedure and the input needs to be measured. Because the system needs to process uncontrolled input, for example, with some noise or photographs captured under varying lighting conditions, a pixel-wise comparison is inapplicable. The input should be reduced to only the features that are determining for a human observer, such as the structure of the overall pattern. These features should be matched with the procedural, while ignoring unwanted characteristics such as noise. Such an abstracted descriptor also needs to be smooth in the parameter space of a typical procedural texture, as compact as possible and fast to evaluate in order to allow for a numerical search to identify as a plausible match.

The following evaluates three texture descriptors for finding a suitable abstraction and for computing a meaningful texture distance. These, on the one hand, pick up
the measures from Bourque and Dudek [2004], namely Fourier space- and Laplace pyramid-based distances. On the other hand, this work includes a newly adapted distance based on a Gabor filter bank [Manjunath and Ma 1996] that emphasizes relevant texture features for human perception and compactness.

### 4.3.1 Fourier Descriptor

The most basic texture feature is the repetition of visual structures. An intuitive approach to characterize a repetition in an image is to count the occurrences of a given visual structure. Mathematically, this can be described by computing the correlation of the given image to the spatial relationships of pixel values in a mask, showing a certain pattern. Wherever the image is similar to the mask the cross-correlation function will be high. Instead of comparing the image to a given mask, the image can also be compared to a shifted version of itself. A strong correlation of the original image to its shifted version means that visual structures repeat. This cross-correlation of an image with a shifted version of itself represents the autocorrelation function and is defined by Pouli et al. [2013] as

$$f_a(x, y) := \sum_{i=0}^{n_i-1} \sum_{j=0}^{n_j-1} \frac{T(i+x, j+y)T(i, j)}{\sigma^2}$$

with a texture image $T$ of size $(n_i \times n_j)$, $T(i, j)$ as its pixel value at coordinate $(i, j)$ and $\sigma$ as the texture’s standard deviation. Wherever the shift $(x, y)$ of the autocorrelation function is equal to the distance between repeating visual texture elements, $f_a$ will have a peak. Therefore the number and the position of peaks and valleys in $f_a$ can describe the texture’s regularity, whereas a random texture will have the only peak at $(0, 0)$. The shape of peaks indicates the coarseness and the directionality of a texture [Lew 2001]. Fine and regular textures have peaks that rapidly drop off, and coarse textures are described in smoother peaks, dropping off more slowly.

The autocorrelation function compares every image pixel to the shifted image and is computationally expensive with a time complexity of $O(n^2)$. Typical sample sizes such as $256 \times 256$ pixels are already unmanageable. However, the transfer of the analysis to the frequency domain greatly improves the performance, with computational costs of $O(n \log n)$ for the discrete fast Fourier transform (DFT). The square of the amplitude spectrum of the DFT of an image, also called its power spectrum, reveals the relative presence of frequencies in the image and is defined as

$$P(u, v) := F_R(u, v)^2 + F_I(u, v)^2$$

with $F_R(u, v)$ as the real and $F_I(u, v)$ as the imaginary parts of the DFT and $u$ and $v$ as the frequencies along the $x$ and $y$ axes of the texture image $T$ [Petrou and Sevilla 2006]. The power spectrum represents the same texture features as the ones
that an analysis of the autocorrelation function exposes. The relationship of the autocorrelation function and the analysis of the power spectrum is grounded on the Wiener-Khintchine theorem, and Pouli et al. [2013] give a detailed explanation for the interested reader.

In practice, the magnitude of the Fourier transform is used instead of the power spectrum [Bourque and Dudek 2004], with

\[
M(u, v) := \sqrt{F_R(u, v)^2 + F_I(u, v)^2}. 
\]

(4.4)

The computation of a similarity distance between two images then comes down to

\[
D_F(T_1, T_2) := \|M_1 - M_2\|_2 
\]

(4.5)

where \(M_1\) and \(M_2\) are the magnitude maps of the DFT, calculated separately for each color channel (R, G, and B) of the texture images \(T_1\) and \(T_2\). The used DFT was implemented by Frigo and Johnson [2005]. As in Bourque and Dudek [2004], entries of the maps are weighted, so that the DC component is scaled by 1, and with an increasing radius, the scaling weight linearly falls to 0.

Summary

The Fourier descriptor equates to the application of an autocorrelation function in the frequency domain, comparing every image pixel to a shifted version of that image, describing repeating structures.

The Fourier distance computes the absolute, squared difference of the magnitude maps of the power spectra of two texture images.

### 4.3.2 Laplace Pyramid Descriptor

The magnitude of the DFT of an image determines its frequency components but gives no insights on its spatial relationships. Even though a DFT has a better performance than the computation of the autocorrelation function, it does not result in a condensed or abstracted representation but still requires the full resolution of the image as output. A leaner representation that detects different frequencies in the spatial domain is given by an approach based on the Laplace pyramid of an image.

A Laplace pyramid is given by the differences of the neighboring levels of a Gaussian pyramid [Burt et al. 1983]. In a Gaussian pyramid, each level is a smoothed and sub-sampled version of the level before. Since smoothing constitutes low-pass filtering, which increases the minimum wavelengths of each image level, the sampling resolution can be reduced accordingly. This is based on the Nyquist sampling theorem, which states that a lossless sampling of a signal can be achieved with a sampling rate of at least two times the maximum frequency contained. In order to build the levels of a
Gaussian pyramid, the kernel size of the smoothing operator is therefore progressively doubled while the pixel resolution is halved. As a result, the pyramid represents the image in multiple scales with low computational costs.

For a texture image $T$ of size $(2^N + 1) \times (2^N + 1)$, the $N$ Gaussian pyramid levels $g_0, ..., g_{N-1}$ are recursively reduced as

$$
g_0(i,j) := T(i,j), \text{ for level } l = 0$$

$$
g_l(i,j) := 2^l \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(2i+m, 2j+n), \text{ otherwise}$$

(4.6)

where $g_l$ is of size $(2^{N-l} + 1) \times (2^{N-l} + 1)$. For the generating kernel $w(m,n)$ for smoothing, Burt et al. [1983] formulated the following constraints: it is separable, has $5 \times 5$ elements, is normalized and is symmetrical, and all nodes at a given level $l-1$ must contribute the same total weight to the nodes at the next level $l$ [Kutulakos 2016]. Please note that we used the more performant implementation with a $3 \times 3$ kernel size, as it is common practice.

The Laplace pyramid is constructed from the Gaussian in that it saves exactly those frequencies for each level that the smoothing kernel filters out. Therefore, Laplacian levels are computed by taking the difference of the two adjacent Gaussian levels $g_{l-1}$ and $g_l$. As the Gaussian levels are progressively sub-sampled, the neighboring levels are of a different pixel size, and $g_l$ needs to be expanded with

$$
g_l(i,j) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(\frac{i-m}{2}, \frac{j-n}{2})$$

(4.7)

which up-samples level $g_l$ with size $(2^{N-l} + 1) \times (2^{N-l} + 1)$ and doubles it to $(2^{N-l+1} + 1) \times (2^{N-l+1} + 1)$ [Burt et al. 1983; Kutulakos 2016].

To compute a distance between two images, the histograms of the Laplace pyramid of both images are compared. Following Bourque and Dudek [2004], the distance is computed by employing the earth mover’s distance (EMD) of the histograms $H_1$ and $H_2$ of level $k$ of the Laplace pyramids over $l$ levels, computed separately for each color channel (R, G, and B) of the texture images $T_1$ and $T_2$:

$$
D_L(T_1, T_2) := \sum_k \text{EMD} \left( H_{1k}(L_{1k}(T_1)), \ H_{2k}(L_{2k}(T_2)) \right)
$$

(4.8)

with

$$
\text{EMD}(H_1, H_2) := \left( \min_{f_{ij}} \sum_{i,j} f_{ij} d_{ij} \right) / \left( \sum_{i,j} f_{ij} \right)
$$

s.t. $f_{ij} \geq 0$, $\sum_j f_{ij} \leq H_{1i}$, $\sum_i f_{ij} \leq H_{2j}$, $\sum_{i,j} f_{ij} = \min \left( \sum_i H_{1i}, \sum_j H_{2j} \right)$

(4.9)
where \( \{ f_{ij} \} \) defines the flows in which each \( f_{ij} \) is the amount that is shifted from the \( i \)th bin of the source to the \( j \)th bin of the target histogram. \( d_{ij} \) is called the \emph{ground distance} between the bins \( i \) and \( j \) [Rubner et al. 1998; Pele and Werman 2009]. Intuitively, the EMD represents the minimal cost of transforming one histogram into the other. In the following, the fast and robust implementation of Pele and Werman [2009] is applied.

The final descriptor considers the five least frequent Laplacian levels and computes the histograms on 256 bins mapped to the standard image values. Values outside this range, which cannot occur in the low dynamic range input images but during optimization, are counted in the outermost histogram buckets. The EMD is a cross-bin distance, including the neighborhood of a bin when computing distances between bins – therefore considering the spatial relationships of image features.

Summary

The \emph{Laplacian pyramid descriptor} is based on the responses of a Gaussian kernel filter bank, varying in scale and orientation, taking out high frequencies in each pyramid level, decomposing the image into different frequency bands.

The \emph{Laplacian pyramid distance} computes the difference of the Laplace pyramid histograms of two texture images with the Earth Mover’s Distance.

**4.3.3 Gabor Filter Bank Descriptor**

Orientation information is crucial for the plausibility of an image distance for a human observer. In order to achieve this, the oriented Laplacian pyramid [Greenspan et al. 1994] applies an oriented filter to each level, differentiating texture features of specific scales and orientations. As the Laplace pyramid represents the responses of a bank of different band-pass filters, a more controlled version of this approach consists in using an intentionally composed filter bank. Daugman [1985] presents such a filter family based on two-dimensional Gabor functions. Daugman [1985] shows that these filters represent the receptive field profiles of simple cells in the mammalian visual cortex, optimizing the joint localization of visual properties in the spatial domain and in the spatial frequency domain [Daugman 1985; Grigorescu et al. 2002].

A Gabor filter is composed from a harmonic, sinusoidal function weighted by a Gaussian distribution, which it uses to extract localized descriptions for differently oriented frequency bands. The two-dimensional Gabor function is defined as

\[
g(x, y, \lambda, \theta, \varphi, \sigma, \gamma) := \exp \left( -\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \right) \cos \left( 2\pi \frac{x'}{\lambda} + \varphi \right)
\]

where

\[
x' = x \cos(\theta) + y \sin(\theta) \quad \text{and} \quad y' = -x \sin(\theta) + y \cos(\theta)
\]

(4.10)
with \( \lambda \) as the frequency of the sinusoid, \( \theta \) as the orientation of the filter kernel, \( \varphi \) as the phase offset of the sinusoid, \( \sigma \) as the standard deviation and \( \gamma \) as the aspect ratio of the Gaussian distribution [Grigorescu et al. 2002]. The filter bank contains a collection of filters that differ in orientation, frequency and width.

The response for one of the individual Gabor filters \( g_i \) is calculated by

\[
R_i := \frac{\iint_{x,y \in A} |(T \ast g_i)(x, y)| \, dx \, dy}{|A| \cdot \iint_{x,y \in A} |g_i(x, y)| \, dx \, dy}
\]  

(4.11)

where \( A \) is the maximal inner image area in which all filters on all scales can sample without crossing the image boundary. All convolution is performed in the frequency domain and computed by the DFT [Frigo and Johnson 2005].

In the preliminary distance computation evaluation, four different orientations and four different frequencies are used on six filter scales, except of the highest frequency on the lowest scale. In addition to computing the responses on the individual color channels \( R, G \) and \( B \), they are also computed on their differences \( R-G, R-B \) and \( G-B \) based on the method from Jain and Healey [1998]. This makes it possible to observe covariance in the color channels and tell, for instance, red-green checkerboards apart from black-yellow ones. This setup results in a total of 588 filter response images \( R_i \) (Figure 4.1). In the final retrieval technique, described in Section 4.6 Interactive Parameter Retrieval, the setup is improved with sixteen different orientations and
color and structure are separated, computing the Gabor descriptor only for the
gray-valued structure images. This reduces the descriptor vector size to 374 filter
response images $R_i$.

Intuitively, two images are similar if and only if the distributions of filter responses
are similar. In order to achieve a compact representation, these histograms are
modeled as Gaussian distributions, storing only arithmetical means $\mu_i$ and standard
deviations $\sigma_i$.

In order to compare distributions, the $W_2$ Wasserstein distance is typically applied.
In the case of 1D, which does not analyze the covariance between different responses
$R_i$ and $R_j$, $W_2$ can be expressed compactly in closed form [Givens and Shortt 1984].
The comparison then corresponds to evaluating an $L_2$ distance in the first two statistical
moments [Dowson and Landau 1982] in a vector space with $\mu_i$ and $\sigma_i$ in separate dimensions.

The final vector computes the distance between texture images $T_1$, $T_2$ by an adapta-
tion of Equations (6) and (7) in [Manjunath and Ma 1996]):

$$D_G(T_1, T_2) := \sum_i d_i(T_1, T_2)$$

where

$$d_i(T_1, T_2) = \frac{|\mu_i^{T_1} - \mu_i^{T_2}|}{\alpha_i} + \frac{|\sigma_i^{T_1} - \sigma_i^{T_2}|}{\beta_i}$$

with the mean $\mu_i$ and the standard deviation $\sigma_i$ of the respective feature vectors for
the input images.

$$\alpha_i := \max_{T \in \mathcal{T}} \mu_i^T - \min_{T \in \mathcal{T}} \mu_i^T \quad \text{and} \quad \beta_i := \max_{T \in \mathcal{T}} \sigma_i^T - \min_{T \in \mathcal{T}} \sigma_i^T$$

normalize the descriptor entries with the minimal and maximal means and standard
deviations from a large natural texture database $\mathcal{T}$ with 10,021 images, which was
also used in the user survey (Section 4.3.4 User Survey on Natural Images).

Summary

The Gabor filter bank descriptor is based on the responses of a Gabor filter
bank, with filters on four different orientations and four different frequencies on
six scales computed for each color channel (R, G, B) and their differences (R-G, R-B and G-B).

The Gabor filter bank distance compares the distributions of filter responses by
evaluating an $L_2$ distance of the first two statistical moments, the arithmetical
means $\mu_i$ and standard deviations $\sigma_i$. 
4.3.4 User Survey on Natural Images

As preliminary evaluation step of the Fourier, Laplace and Gabor descriptors, their similarity measures $D_F$, $D_L$ and $D_G$ between different textures are compared to the similarity assessment of humans. The test was a user survey with a natural image retrieval scenario, for which a texture image database was built from scratch. In total 100 anonymous participants took part.

Survey participants were asked to choose which of the three texture descriptors selected for a given natural texture image the most similar one, retrieved from a sizable database of natural texture images.

The results give insights on how well the three descriptor respectively capture and abstract visual features that are relevant for a human observer. Hence the algorithm should rank the same textures as similar as a human would.

Image Database

The validity of the descriptors was evaluated on a database of 10,021 natural color textures, extracted from photographs. In order to set up the database, all photographs tagged with the word “texture” and licensed under the Creative Commons BY 2.0 license were downloaded from the community photo collection www.flickr.com (57,150 images in total). From this image selection, a centered $256 \times 256$ was cropped for each photograph, and a manual selection of 10,021 textures was made from these candidates. In order to do so, an intentionally loose definition of what a texture constitutes keeps the system setup and its evaluation as generic as possible. The definition used states that a texture is a two-dimensional image showing close to no perspective with a repeating pattern, which can be random to a certain degree. The image must contain local areas of a specific repeating pattern but the content can vary globally, as described by Wei et al. [2009].

Target Images

As target images, a sub-selection of 30 images was chosen manually from the database (the Target images of Figure 4.2). The selection covers a wide range of texture features, including stochastic and deterministic patterns, certain color characteristics as well as possible failure cases, such as written text, which is not representable by the design space of the test textures.

Task

For evaluating the performance of the descriptors in the natural texture retrieval scenario, the best match was determined for each of the 30 target images according to each of the distance metrics. The metrics agreed in only two cases; in nine cases, two out of three agree. For these nine and for the remaining 19 cases, the user survey asked participants to vote on which of the identified matches agreed best with the target image.
Results

The percentages of the user preference for each target / result image pair for each descriptor indicate whether the metrics perform differently. The results are shown in Figure 4.2. In the cases where metrics agree about the best match, the preference value for both are counted. Hence, the total agreement can numerically exceed 100%.

On average, 49.17% of the participants found the $D_F$ to have delivered the best match, 37.90% voted for the $D_G$ and 42.93% for $D_L$.

However, the standard deviations of the vote percentages of 36.07%, 33.80% and 34.57%, respectively, suggest that a winner could not be identified. Applying the Friedman rank sum test to the data supports this conclusion with a $p$-value of 0.6217. Hence, we cannot reject the null hypothesis that the descriptors perform equivalently.

Summary

For natural images, on average 49.17% of the participants agreed with the Fourier similarity measure $D_F$ the most. However, the statistical evaluation does not allow for a generalizable conclusion and ranks the results as potentially random.

Hence, the user survey with a natural image retrieval scenario can not identify a superior texture descriptor.

4.4 Optimization Strategies

With the means to compare textures and define their similarity, techniques to search a parameter space have to be established. A method should identify, from the space of all possible parameter combinations, the parameter set of a procedural texture that makes its rendering most similar to the target texture.

A common approach for finding minima in a function is an iterative search with numerical optimization. For the formulated problem, this implies minimizing the distance between the target and the rendered output of the procedure for the current parameter set as Bourque and Dudek [2004] have applied it. In order to evaluate such a numerical optimization, two different strategies – the Nelder-Mead and an implementation of the Levenberg-Marquardt method – are compared.

4.4.1 Initial Parameters

Both Nelder-Mead and Levenberg-Marquardt optimization depends only on local information during their search for minimizing function parameters. Hence, they are sensitive to the optimization start, lest they end up in a non-global, local minimum. When starting an expensive local search, it is crucial to initialize it with parameters
### Figure 4.2: The best matching natural texture pictures for the target images, retrieved from the 10,021 database images. The percentages indicate the user preference for each metric. For equal percentages, the value is counted for both metrics, leading to agreements that can exceed 100%. Image sources: please refer to Section 7.4 Image References.
as close to the minimum as possible in order to overstep possible local minima of the function. A common approach is to sample the search domain randomly and to select the parameters with the lowest cost as starting points. For the $D_F$ distance, this requires computing and evaluating all texture outputs for the random samples before the actual fitting takes place or alternatively precomputing those images and storing them in full in a database for look-up before starting. $D_L$ and $D_G$ permit a much more efficient strategy because the intermediates $H_i$ and $(\mu_i, \sigma_i)$ can be stored, respectively.

### 4.4.2 The Nelder-Mead Algorithm

The downhill simplex, also known as the adjusted Nelder-Mead-method [Nelder and Mead 1965], is a direct search algorithm that compares cost function values with no need for derivative information. It is a simple heuristic with the potential disadvantage of slow convergence.

The downhill simplex algorithm constructs from a set of $n$ parameters an initial simplex with $n + 1$ vertices, each vertex representing a different parameter set. The following summarizes the description of the method by Nelles [2001]: In each iteration, the costs for the simplex vertices are evaluated. The vertex with the highest cost is then reflected at the centroid of the simplex, and the original point is deleted. Repeating this vertex adjustment moves the downhill simplex until a termination criterion is reached and a local minimum is found. As convergence is a relevant issue with the method, a variety of termination criteria have been developed. However, originally, Nelder and Mead [1965] stop when the size of the simplex becomes too small in comparison to the curvature of the cost function. This criteria is defined as

$$\sqrt{\frac{1}{n+1} \sum_{i=0}^{n} (f(\vec{v}_i) - \bar{f})^2} < \epsilon$$

where

$$\bar{f} := \frac{1}{n+1} \sum_{i=0}^{n} (f(\vec{v}_i))$$

with $\vec{v} = \{v_0, ..., v_n\}$ being the current vertices of the simplex, $f$ the cost function to be minimized and $\epsilon$ a predefined constant threshold. The original downhill simplex algorithm only allowed for equidistant points. Nelder and Mead extended the method by including optional expansions and contractions of the simplex when placing a new vertex to find the lowest cost reachable. This adjustment significantly improves the convergence performance [Nelles 2001]. As implementation of the Nelder-Mead-method [Nelder and Mead 1965] we use the one by Jia [2010].
Summary

The Nelder–Mead algorithm is a multi-dimensional heuristic search method that only compares function values, not requiring derivatives. In return it suffers from slow convergence.

4.4.3 The Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [Levenberg 1944; Marquardt 1963] is a robust non-linear least square solver with a high convergence probability. It combines ideas from gradient descent and the Gauss-Newton method. In this work, Google’s Ceres library by Agarwal et al. [2016] is employed. The following section gives a summary of the algorithm, mainly based on the tutorial by Ranganathan [2004] but also including Agarwal et al. [2016]; Nelles [2001]; Madsen K. and O. [2004]; Velho et al. [2008].

On a basic level, the optimization problem is defined as minimizing

\[ f(\vec{x}) = \frac{1}{2} \| \vec{c}(\vec{x}) \|^2 \]  

(4.15)

where \( \vec{x} \in \mathbb{R}^n \) is the \( n \)-dimensional parameter vector and \( \vec{c} \) the \( m \)-dimensional cost function \( \vec{c}: \mathbb{R}^n \rightarrow \mathbb{R}^m \) of \( \vec{x} \) as \( \vec{c}(\vec{x}) = (c_1(\vec{x}), ..., c_m(\vec{x})) \), with \( m \leq n \). The multiplication of \( \frac{1}{2} \) is introduced for convenience, omitting the inclusion of a factor 2 when expressing derivatives in the following.

To make the optimization manageable, it is broken down to finding a local minimum of \( \vec{c}(\vec{x}) \), calculated by an iteration of approximations. For each iteration, a correction term is computed and the parameter set \( \vec{x} \) accordingly adjusted to move closer toward a local minimum of \( \vec{c}(\vec{x}) \). An intuitive solution to this problem is to follow the negative of the scaled gradient, given as

\[ x_{i+1} = x_i - \lambda \nabla f. \]  

(4.16)

Here, the Jacobian matrix \( J \) of \( \vec{c} \) with regard to \( \vec{x} \) is employed, computing the derivatives of \( f \) as \( J(\vec{x}) = \frac{\partial \vec{c}}{\partial \vec{x}} \), where \( 1 \leq j \leq m, 1 \leq i \leq n \). In the non-linear case, this leads to

\[ \nabla f(\vec{x}) = \sum_{j=1}^{m} c_j(\vec{x}) \nabla c_j(\vec{x}) = J(\vec{x})^T \vec{c}(\vec{x}). \]  

(4.17)

However, such a gradient descent approach suffers from convergence issues. First, the step size of the correction term at each iteration relates to the size of the gradient, leading to unfavorable step sizes. Large steps are taken at steep slopes, easily overstepping local minima, while small steps in areas of low slopes increase the number of iterations. Second, the curvature of the cost function might be different for different directions, and following the gradient might not lead directly to the
nearest local minimum. Employing the curvature of the cost function in addition
to the gradient information greatly improves convergence. In doing so, the Gauss-
Newton method minimizes a twice-differentiable function by approximating it with a
second order Taylor series around a point $\vec{x}_0$. By assuming the cost function $f$
to be quadratic around the current point, the Gauss-Newton method solves for $\nabla f(\vec{x}) = 0$. Setting

$$\nabla f(\vec{x}) = \nabla f(\vec{x}_0) + (\vec{x} - \vec{x}_0)^T \nabla^2 f(\vec{x}_0)$$

(4.18)
to 0 and replacing $x_0$ with $x_i$ and $x$ with $x_{i+1}$, gives the update rule

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1}\nabla f(x_i).$$

(4.19)

The quadratic assumption made above constitutes a simplification, and an approxi-
mation of the Hessian $\nabla^2 f(x_i)$ further reduces the problem. Close to the solution,
for small enough costs $c_j(\vec{x})$, linearity is approximated so that $\nabla^2 c_j(x)$ is small as
well. Then the linear computation

$$H = \nabla^2 f(\vec{x})^2 = J(\vec{x})^T J(\vec{x})$$

(4.20)
can be employed. The better convergence behavior is bound to small cost function
values and the resulting linearity around the current point. If the current position is
still far from a minimum, the Gauss-Newton method might diverge, and a gradient
descent is favorable. The Levenberg-Marquardt algorithm is based on this considera-
tion and combines the Gauss-Newton and gradient descent approaches, depending
on whether the previous iteration moved closer to a minimum and decreased the cost.
Levenberg presented this idea, and Marquardt’s adjustments ensured reasonable step
sizes. A Levenberg-Marquardt update is defined as

$$x_{i+1} = x_i - (H + \lambda \text{diag}[H])^{-1}\nabla f(x_i).$$

(4.21)

with $H$ as the Hessian matrix and $\lambda$ as the damping parameter, which is dynamically
changed. If the costs decrease with the taken step, $\lambda$ is divided by a given constant,
implementing a Gauss-Newton step for small values of $\lambda$. If the costs increase, also
$\lambda$ is increased and multiplied by a given constant, adapting the more conservative
gradient descent optimization for large values of $\lambda$. Marquardt’s inclusion of the
diagonal of the Hessian (instead of Levenberg’s original formulation with the identity
matrix) implements the consideration of the curvature even for large values of $\lambda$,
ensuring that the gradient descent takes large steps in flat cost function areas and
small steps within high curvatures.

In modern versions of the Levenberg-Marquardt algorithm, including the configura-
tion of the Ceres library, the optimization is not solved by modifying the damping
factor $\lambda$ but by a trust-region approach. This approach is parameterized by $\Delta$, the
size of an area in which a quadratic model of the objective function is trusted to ap-
proximate the function closely. Instead of deciding on a direction and step size as the
previously described line-search approach does, the trust region algorithm jumps in
each iteration to the minimum of the modeled linearization. The agreement between
the predicted and actual model values controls the heuristic for the trust region size,
a similar approach to the update of $\lambda$ as described above. If the model predicts the
actual values of $f$ closely, $\Delta$ is increased; otherwise, it is decreased. Agarwal et al.
[2016] show that the trust-region approach corresponds to the parameter update
equation of the line-search method.

The Levenberg-Marquardt algorithm is terminated when multiple criteria are smaller
than given user constants. The criteria are the gradient norm, the norm of the
step change in the parameter values, the cost function change and the number of
iterations.

Summary

The Levenberg-Marquardt algorithm is a multi-dimensional and robust non-linear
least square search method with a high convergence probability for moderately
sized search spaces. As the algorithm requires a matrix inversion, which is
usually approximated with methods such as singular value decomposition, these
computations become unmanageable with more than a few thousand search
parameters.

However, even with small scale optimization tasks such as ours with only 9-11
search parameters, the runtime of the non-linear optimization can not archive
interactive performance.

4.5 Preliminary Evaluation

The survey on the natural image database could not identify a preferred texture
descriptor and further investigation is in order.

The following preliminary evaluation aims to ensure that the chosen distance metric
identifies visual similarities in the space of the procedural texture space according
to human perception. The representation of a metric also needs to be efficiently
computed and stored to allow for interactive performance. Similarly, when considering
an optimization strategy, the visual quality of its search result in combination with
its performance are crucial.

To investigate the alignment of the descriptors and similarity metrics to human
perception and their relative performance, a second user survey was conducted.
In total, 100 anonymous participants took part, and a retrieval scenario in the
procedural texture space was evaluated.

As a prerequisite, the optimal combination of metric and optimization techniques
was numerically established. Based on the findings of the survey the favored setup is
then also further investigated numerically.
4.5.1 Procedural Test Textures

For the preliminary evaluation, a set of exemplary procedural test textures was implemented. The textures depend on 9–11 scalar parameters, as shown in Figure 4.3. All textures include two RGB colors and an angle for planar rotation, accounting for seven of those parameters. The texture structures include:

- **Perlin Noise** – the classical noise function by Perlin [1985]. It is parameterized with the noise frequency and the amount of anisotropic stretch, leading to a total of nine parameters.

- **Turbulence Noise** – a Perlin noise [Perlin 1985] with a persistence parameter and anisotropic stretch, leading to a total of nine parameters.

- **Rigid Turbulence Noise** – the extension of a turbulence noise with ridges [Perlin 1985], adding a ridge offset parameter, leading to a total of 10 parameters.

- **Lined Wood** – a custom implementation that is parameterized by the density of sharp lines, the density of underlying low-frequency streaks, line transparency and the transparency of additive grain. With a total of 11 parameters, this is the most complex texture in the test set; it represents a layered, complex interplay between different parameters.

Aside from natural constraints for values such as color and frequency ranges and the rotation angle, the possible input parameter ranges were kept as large as possible.

4.5.2 User Survey on Procedural Textures

Numerical Groundwork

In order to perform a similar user survey on synthesized images generated from the procedural textures, an optimization technique first needed to determine the most
4.5 • Preliminary Evaluation

Figure 4.4: Box plots of the optimization costs and timings for the three descriptors, each for both the Nelder-Mead (NM) and Levenberg-Marquardt (LM) optimizers and their differences in red. As for the difference NM timings are subtracted from LM’s, the red plot has high values when NM performs better. The whiskers cover 95% of the data, the boxes are at quartiles and the bisector identifies the median.

suitable parameter set to match the target images. For that possible descriptor / optimization setups were evaluated for generating the data for the survey. For different cost functions, non-linear optimization method performances may significantly vary. In order to decide which one of the two methods, Nelder-Mead or Levenberg-Marquardt, to use in the evaluation of the descriptor performance, each optimizer was run for four chosen procedural textures for the data set of target images. Each optimizer started with the parameter set that led to the smallest distance within a million tested random parameter set samples. The plots of Figure 4.4 show that the Levenberg-Marquardt delivered lower distances between the rendered and the input images for Gabor and Fourier type descriptors, whereas Nelder-Mead achieved better results for the Laplace pyramid histogram. This can be attributed to the latter distance function being less smooth.

The termination criteria, especially the convergence cost threshold, were set equivalently for both optimizers as far as possible. Furthermore all settings were conservative, with a maximum iteration count (500) and a low termination threshold for cost difference between subsequent iterations ($10^{-6}$ for $D_G$, $10^{-12}$ for the others), causing outliers in the runtime when minimal improvements were sought. Therefore, the worst-case runtimes were not directly comparable. However, Levenberg-Marquardt terminated more quickly than Nelder-Mead on most samples with the Gabor and
Laplace descriptors, while the latter had the advantage when using the Fourier descriptor.

Task

According to the above findings, the second user survey scenario employed the Nelder-Mead method for searching with $D_L$ costs and Levenberg-Marquardt for the others. For each of the 30 target texture images and for each descriptor, the overall best matches found in the parameter spaces of the Perlin, turbulence, ridged turbulence and lined wood textures were presented. Again the participants were asked to identify the image they determined to be closest to the target, thus voting for the best performing distance metric.

Results

In order to determine the best matches of the synthesized images generated from the procedural textures, the behavior differs from the natural images task, as the results in Figure 4.5 show. For the Fourier, Gabor and Laplace pyramid histogram-type metrics, the averages user preferences were 21.37%, 69.70%, and 8.93%. At 20.74%, 22.58%, and 14.83%, the standard deviations were still high, but the Friedman rank sum test indicates that the null hypothesis may be confidently rejected at a $p$-value of $8.295 \cdot 10^{-8}$. Analyzing the user preferences on a per-image basis, it can be seen that the user preference for the Gabor-type metric was most dominant in cases where the other metrics led to results with pronounced differences in color hues. This suggests that the added consideration of color channel differences in the Gabor-type metrics considerably contributes to its performance.

4.5.3 Numerical Comparisons

As the user survey resulted in a preference for the Gabor-type metric $D_G$ and as the Levenberg-Marquardt provides on average the better runtime, this specific combination of distance metric and optimization method is further evaluated.

Convergence Behavior

While the conservative thresholds discussed above can lead to high iterations, a plausible result is achievable after only 10 iterations in most cases. Examples of the convergence behavior are given in Figure 4.6, showing the results of the texture that gives the closest match for that target. The targets are exemplary for the range of starting close to a local minimum of the cost function to when a larger distance had to be covered by the optimizer. The start images are the result of choosing the best parameter set out of a million random samples.

Starting Parameters Search

The choice of the starting parameters is crucial for a successful convergence of non-linear optimization algorithms. The following comparison evaluates the effect of different database sizes. Figure 4.7 plots for databases with $10, 100, 1000, 10^4, 10^5$
4.5 • Preliminary Evaluation

<table>
<thead>
<tr>
<th>Target</th>
<th>$D_F$</th>
<th>$D_G$</th>
<th>$D_L$</th>
<th>Target</th>
<th>$D_F$</th>
<th>$D_G$</th>
<th>$D_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
<td><img src="image16" alt="Image" /></td>
</tr>
<tr>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
<td><img src="image21" alt="Image" /></td>
<td><img src="image22" alt="Image" /></td>
<td><img src="image23" alt="Image" /></td>
<td><img src="image24" alt="Image" /></td>
</tr>
<tr>
<td><img src="image25" alt="Image" /></td>
<td><img src="image26" alt="Image" /></td>
<td><img src="image27" alt="Image" /></td>
<td><img src="image28" alt="Image" /></td>
<td><img src="image29" alt="Image" /></td>
<td><img src="image30" alt="Image" /></td>
<td><img src="image31" alt="Image" /></td>
<td><img src="image32" alt="Image" /></td>
</tr>
<tr>
<td><img src="image33" alt="Image" /></td>
<td><img src="image34" alt="Image" /></td>
<td><img src="image35" alt="Image" /></td>
<td><img src="image36" alt="Image" /></td>
<td><img src="image37" alt="Image" /></td>
<td><img src="image38" alt="Image" /></td>
<td><img src="image39" alt="Image" /></td>
<td><img src="image40" alt="Image" /></td>
</tr>
<tr>
<td><img src="image41" alt="Image" /></td>
<td><img src="image42" alt="Image" /></td>
<td><img src="image43" alt="Image" /></td>
<td><img src="image44" alt="Image" /></td>
<td><img src="image45" alt="Image" /></td>
<td><img src="image46" alt="Image" /></td>
<td><img src="image47" alt="Image" /></td>
<td><img src="image48" alt="Image" /></td>
</tr>
<tr>
<td><img src="image49" alt="Image" /></td>
<td><img src="image50" alt="Image" /></td>
<td><img src="image51" alt="Image" /></td>
<td><img src="image52" alt="Image" /></td>
<td><img src="image53" alt="Image" /></td>
<td><img src="image54" alt="Image" /></td>
<td><img src="image55" alt="Image" /></td>
<td><img src="image56" alt="Image" /></td>
</tr>
<tr>
<td><img src="image57" alt="Image" /></td>
<td><img src="image58" alt="Image" /></td>
<td><img src="image59" alt="Image" /></td>
<td><img src="image60" alt="Image" /></td>
<td><img src="image61" alt="Image" /></td>
<td><img src="image62" alt="Image" /></td>
<td><img src="image63" alt="Image" /></td>
<td><img src="image64" alt="Image" /></td>
</tr>
<tr>
<td><img src="image65" alt="Image" /></td>
<td><img src="image66" alt="Image" /></td>
<td><img src="image67" alt="Image" /></td>
<td><img src="image68" alt="Image" /></td>
<td><img src="image69" alt="Image" /></td>
<td><img src="image70" alt="Image" /></td>
<td><img src="image71" alt="Image" /></td>
<td><img src="image72" alt="Image" /></td>
</tr>
<tr>
<td><img src="image73" alt="Image" /></td>
<td><img src="image74" alt="Image" /></td>
<td><img src="image75" alt="Image" /></td>
<td><img src="image76" alt="Image" /></td>
<td><img src="image77" alt="Image" /></td>
<td><img src="image78" alt="Image" /></td>
<td><img src="image79" alt="Image" /></td>
<td><img src="image80" alt="Image" /></td>
</tr>
<tr>
<td><img src="image81" alt="Image" /></td>
<td><img src="image82" alt="Image" /></td>
<td><img src="image83" alt="Image" /></td>
<td><img src="image84" alt="Image" /></td>
<td><img src="image85" alt="Image" /></td>
<td><img src="image86" alt="Image" /></td>
<td><img src="image87" alt="Image" /></td>
<td><img src="image88" alt="Image" /></td>
</tr>
<tr>
<td><img src="image89" alt="Image" /></td>
<td><img src="image90" alt="Image" /></td>
<td><img src="image91" alt="Image" /></td>
<td><img src="image92" alt="Image" /></td>
<td><img src="image93" alt="Image" /></td>
<td><img src="image94" alt="Image" /></td>
<td><img src="image95" alt="Image" /></td>
<td><img src="image96" alt="Image" /></td>
</tr>
<tr>
<td><img src="image97" alt="Image" /></td>
<td><img src="image98" alt="Image" /></td>
<td><img src="image99" alt="Image" /></td>
<td><img src="image100" alt="Image" /></td>
<td><img src="image101" alt="Image" /></td>
<td><img src="image102" alt="Image" /></td>
<td><img src="image103" alt="Image" /></td>
<td><img src="image104" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 4.5: The best matching procedural texture images for the target images, chosen from the Perlin, turbulence, ridged turbulence and lined wood textures. The percentages indicate the user preference for each metric. For equal percentages, the value is counted for both metrics, leading to agreements that can exceed 100%. Image sources: please refer to Section 7.4 Image References.
<table>
<thead>
<tr>
<th>Start</th>
<th>10 Iterations</th>
<th>Final</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td>(#29, 9m) 0.023</td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td>(#34, 7m) 0.116</td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td>(#30, 9m) 0.029</td>
</tr>
<tr>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td>(#41, 16m) 0.032</td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td>(#41, 16m) 0.0320</td>
</tr>
<tr>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td>(#63, 23m) 0.200</td>
</tr>
<tr>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td>(#52, 17m) 0.163</td>
</tr>
<tr>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
<td>(#256, 132m) 0.239</td>
</tr>
<tr>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td>(#30, 9m) 0.037</td>
</tr>
<tr>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
<td>(#19, 7m) 1.726</td>
</tr>
</tbody>
</table>

Figure 4.6: Convergence behavior examples of the Levenberg-Marquardt optimization technique. The start images have been retrieved from a million random samples. The examples show the variety of starting close to the optimum and when a larger distance had to be covered by the optimizer. The costs are given on the right of each rendering and the final iteration counts and timings in minutes in parentheses. For most cases, the Levenberg-Marquardt optimizer arrives after only 10 iterations at parameters close to the local minimum. Image sources: please see Section 7.4 Image References.
and \(10^6\) entries the start distance and the achieved end distance once the optimizer
finished.

Because the appearance space of the procedural textures is much smaller than that
of all natural textures, it cannot be expected that results perfectly match the target
queries. Accordingly, their distance distribution can never reach 0. For sufficiently
large database sizes, an unchanging distribution would be expected. As this is not
yet the case for \(10^5\) and \(10^6\) images, it cannot safely be concluded that the database
size was sufficient to guarantee a globally optimal solution. Nevertheless, for both
the starting distance and the end distance, increasing the random set size quickly
brings diminishing returns, and it seems that for the textures tested here, with 8 to
11 parameters, a cache size of \(10^5\) would likely be reasonable.

For reference, a typical database takes 9GB of data for \(10^6\) entries (double data
type, nine texture parameters, binary storage). A un-optimized brute-force search
on this database takes about 30s, of which considerable time is spent on input and
output.

### 4.5.4 Preliminary Conclusions

In order to decide how to proceed with solving the problem of interactive param-
eter retrieval for procedural textures, the preliminary evaluation gives valuable
insights.

When comparing the Fourier-, Gabor- and Laplace-descriptor distances to human
perception of image similarities, the Fourier distance is rated best for natural images.
However, the dominance of that distance could not be statistically supported, meaning
the results are potentially random. In the case of comparing synthesized images
to natural target images, there is a statistically grounded preference for the Gabor
distance. As the ultimate goal is to search in the synthesized image space, the Gabor
distance is the preferable choice in terms of visual quality.

When considering the achieved visual quality of the optimization, the comparison of
the resulting images in Figure 4.6 is central, changing the chosen course toward a
solution. The preliminary evaluation reveals that a non-linear optimization does not

---

Figure 4.7: Effect of the initial search for start parameters. Shown are the distribu-
tions of initial (blue) and final (orange) distances for 10 target images and all test
texture types, for \(10^1, 10^2, \ldots, 10^6\) sizes of starting database. The whiskers cover
95\% range of the data, the boxes are at quartiles and the bisector identifies the
median.
necessarily lead to a compelling visual improvement that is directly noticeable to a human observer. Even though the non-linear optimization reduces the distances numerically, as the comparisons of the right and left plots in Figure 4.7 show, these further improvements are visually less noticeable, as indicated by the comparison of the renderings of the start parameters, the renderings after 10 Levenberg-Marquardt iterations and the final images in Figure 4.6. Especially in regard to the matched structures of the pattern, the randomly chosen start parameters are already a reasonable match in most cases. The prominent deficits of the start parameters are in some cases due to the color of a pattern and a global rotation.

In addition to the visual quality, the runtime of the non-linear optimization must be deliberated. With the Gabor descriptor, the non-linear optimization has a mean of about 28 minutes to compute a result (Figure 4.4). There are some options are left to improve on the runtime with algorithmic or code optimizations. But overall in this context non-linear optimization proofs to be not applicable because interactive performance is the goal.

As next steps, the approach of a random search is investigated and further improved instead of applying a non-linear optimization. An even and dense sampling of the parameter space of the textures replaces the random sampling of the synthesized image space for finding a match to the target image. Additionally, the re-formulation of the distance metric in a texture descriptor space is crucial. A descriptor vector can be independently computed per parameter set and compactly stored. Building upon research in texture retrieval, a pre-computed database of parameter-descriptor pairs can then be searched upon runtime, drastically improving performance.

The filter responses of the Gabor distance metric and the histograms of the Laplace pyramid can be interpreted as texture descriptor, pre-computed, and stored as tuples (588 · 2 = 1176 doubles for Gabor, 256 · 5 · 3 = 3840 for Laplace). This is not feasible for the Fourier-based metric in terms of storage requirements because the whole power spectrum image of the rendering of a parameter set needs to be stored (256 · 256 · 3 = 196608 double values).

It can be concluded that the Gabor distance is fully comparable to the distance metrics of the closely related work from Bourque and Dudek [2004]. A solution based on applying a compact Gabor texture descriptor in combination with an iterative search through a precomputed lookup table is presented in the next section.

Summary

<table>
<thead>
<tr>
<th>The preliminary evaluation results in the following insights:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• For synthesized images from the procedural textures in contrast to the survey on natural images, the user study results in a statistically grounded preference for the Gabor-descriptor distance.</td>
</tr>
<tr>
<td>• Gabor and Laplace descriptors are pre-computable and can be compactly stored.</td>
</tr>
</tbody>
</table>
• The visual improvements from non-linear optimization do not outweigh its non-interactive computation times.

Grounded in these results, the Gabor texture descriptor is chosen and non-linear optimization disregarded. An iterative search through a pre-computed lookup table constitutes a novel compromise between visual quality and performance for example-based procedural texture synthesis.

4.6 Interactive Parameter Retrieval


However, if no further information on the interplay between structural parameters and output image properties is available, potentially high-dimensional parameter spaces are an inherent problem of search-based solution strategies. With an increasing number of parameters the search space becomes infeasibly large for identifying a visually satisfactory match.

To alleviate the dimensionality problem, the following solution addresses color and structure parameters separately. The approach reduces the search space by all color dimensions and makes a near-exhaustive random sampling of the parameter space feasible for structurally interesting texture models. Based on the previously evaluated Gabor texture descriptor, a perceptually motivated distance metric is constructed and calibrated to the end user of the technique. This allows to precompute a large ($10^5 - 10^6$ entries) set of descriptor vector / structure parameter vector pairs in an advance step and to achieve interactive optimization performance at runtime with only moderate memory requirements. The proposed technique is outlined in Figure 4.8.

Aside from the descriptor calibration, the approach is configuration free and works robustly for fundamentally different texture model classes. The evaluated classes are noise textures, regular grids, and special-purpose texture models for tiled and wooden surfaces.
Figure 4.8: Parameter retrieval pipeline: Principal Component Analysis reveals the constituent colors of a two-tone input image and a corresponding blend map. By identifying the most similar image from a large database of images generated by a procedural texture model according to a texture descriptor, the structural parameters can be retrieved. Together with the colors, they produce an image closely matching the input. Image source: [3dtotal 2014].

4.6.1 Texture Models

Restructuring the previously used test textures, a three-channel color texture is in the following formally modeled as

\[ T_{\vec{a}, \vec{c}_1, \vec{c}_2} : \mathbb{R}^2 \to \mathbb{R}^3, \quad \vec{x} \mapsto (r, g, b)^T \]  \hspace{1cm} (4.22)

as convex combination of two RGB colors \( \vec{c}_1, \vec{c}_2 \in [0, 1]^3 \). Each pixel position \( \vec{x} \) may be expressed as

\[ T_{\vec{a}, \vec{c}_1, \vec{c}_2}(\vec{x}) = (1 - s_{\vec{a}}(\vec{x})) \cdot \vec{c}_1 + s_{\vec{a}}(\vec{x}) \cdot \vec{c}_2. \]  \hspace{1cm} (4.23)

The structure function

\[ s_{\vec{a}} : \mathbb{R}^2 \to [0, 1] \subset \mathbb{R} \]  \hspace{1cm} (4.24)

controls the blend between the two colors dependent on a parameter vector

\[ \vec{a} \in ([a_{\text{min,1}}, a_{\text{max,1}}], \ldots, [a_{\text{min,n}}, a_{\text{max,n}}]) \subset \mathbb{R}^n \]  \hspace{1cm} (4.25)

and is implemented as a procedural texture.

Thus, \( T \) is defined by the discrete choice of a structural function (manually selected by the user or automatically chosen by the algorithm), two color tones, \( \vec{c}_1, \vec{c}_2 \), and a structural parameter vector \( \vec{a} \) depending on the concrete choice of the structural function. This formulation expresses colors using RGB values. The RGB color system is often times favorable for the intended place of the technique deep in a digital asset production pipeline. While perceptual uniformly scaled color spaces would improve predictive capabilities of perception, they would be most helpful only after lighting and color grading has already taken place.

The application of two-tone texture models also assumes that the input picture shows a two-tone texture. This implies that the distribution of its color values in the RGB cube follows a straight line. A principal component analysis (PCA) of the color values reveals the line’s location as the first principal component. This principal component is the one corresponding to the eigenvector of the covariance.
matrix with the largest eigenvalue. The extremal pixel values of the input picture define its two constituent tones \( \vec{c}_1 \) and \( \vec{c}_2 \). An affine transformation maps the color with the smaller luminance value to 0 and the greater one to 1. If the luminance comparison fails, the three channels are used in lexicographical order for comparison. This mapping transforms the input picture to a gray valued structural image \( s_{\text{target}} \).

In the next step only the structural information of this gray valued structural image needs to be matched, as the determined tones \( \vec{c}_1 \) and \( \vec{c}_2 \) constitute the final colors for the texture program.

For input images that deviate from the two-tone assumption, the pixel values are projected onto the straight line \( \vec{c}_1 \vec{c}_2 \). In the case of the projected values falling outside the RGB unit cube, the intersection points of the line with the RGB cube are chosen as the two constituent colors.

Summary

For texture \( T \), color and structure are separately matched.

- **Color** is assumed to consist of two RGB color tones, \( \vec{c}_1 \) and \( \vec{c}_2 \), which are identified by a principal component analysis.

- **Structure** is controlled by a function, \( s_a \), and a parameter vector, \( \vec{a} \).

### 4.6.2 Distance Metric

When mimicking human perception, both global statistics and local structure of an image are of relevance and have been successfully used in the past [Pouli et al. 2011; Manjunath and Ma 1996]. Global statistics relate to features such as overall brightness and contrast. Local structures identify frequencies and their distributions, which correspond to structural orientations, such as described by the Gabor filter bank responses.

The developed metric, based on the findings of the preliminary descriptor evaluation, incorporates both global and local structures into a descriptor vector, which can be independently computed per input image and compactly stored.

As global statistical features of the structure functions in the observed domain, the mean and standard deviations \( \mu_{\text{global}} \) and \( \sigma_{\text{global}} \) are employed.

In order to model the distributions of frequency content in the structure maps, the previously evaluated Gabor descriptor vector is used. An individual response \( R_i \) of a Gabor filter is calculated as in Equation (4.11). The following retrieval technique employs sixteen different orientations and four different frequencies. An isotropic kernel on six filter scales each is additionally added, except for the highest frequency on the lowest scale. This setup covers scales from the sampling limit up to the maximum filter size for which the kernels observe quasi-repetitive structures with the \( 256^2 \) textures, which are used later on. This results in a total of 374 filter response images \( R_i \).
Note that full color texture descriptors would need at least three separate descriptor vector entries for each of the color channels, possibly more in order to resolve color correlation as the Gabor descriptor in the preliminary evaluation has shown (Section 4.3.3). The presented approach of explicitly mapping to a two-tone space remains 1-D and thus compact while being descriptive.

The complete $m$-dimensional descriptor vector for the image $I$, including both global statistics and local structures, is then assembled from the mean $\mu_i$ and standard deviations $\sigma_i$ of the filter responses $R_i$ interpreted as individual sets of numbers concatenated to the global values $\mu_{\text{global}}$ and $\sigma_{\text{global}}$. This induces a perceptually motivated $L_2$ distance metric

$$D(I_1, I_2) := \sqrt{w^2_{\text{global}} \cdot d^2_{\text{global}} + \sum_i w^2_i \cdot d^2_i(I_1, I_2)} \quad (4.26)$$

where $d_{\text{global}}$ and $d_i$ are the Wasserstein distances of the respective distributions (the square root being omitted in practice). $D$ depends on weights that express the relative importance of the respective descriptor entries. These weights model the frequency-dependent contrast sensitivity of the human visual system [Campbell and Robson 1968]. As the specific weights depend on both the observer and on viewing conditions, they need to be calibrated individually in production. For the presented experiments, $w_i$ is chosen dependent on the scale of the Gabor filters proportional to the location of beginning contrast sensitivity as determined by a contrast test chart displayed on the monitor. Exact values and peaks vary between observers, but follow a similar distribution for different individuals. The weights used for all results in this work are reported in Figure 4.9.

This leaves the balancing of local feature sensitivity vs. global brightness and contrast, as expressed by $w_{\text{global}}$. As Figure 4.10 shows, there is a trade-off to be made. For low values of $w_{\text{global}}$, local structure dominates the descriptor, and while orientations

Figure 4.9: The top 15% of the total test chart used to determine contrast weights, with the weights used reported below the corresponding sinusoids. The full chart is 4096 pixels wide and was used in actual size with scrolling.
and edge densities are recovered well, average brightness is lost. For high values, the opposite effect occurs.

Such a balancing of features again greatly depends on production requirements. In order to provide a comparable evaluation, this work uses the same $w_{\text{global}}$ for all texture classes. Following the intuition that global statistics and local structure are equally important, exactly the sum of weights for all descriptor elements representing local structure is chosen as global weights

$$w_{\text{global}} := \sum_{i} w_i = 354.82.$$  \hfill (4.27)

Summary

The novel distance metric is based on a descriptor vector of 374 Gabor filter responses and the global mean and standard deviation of the texture structure function.

The perceptually driven weighting of the vector elements balances local feature sensitivity versus global brightness and contrast.
4.6.3 Retrieval

With the image distance defined above, the ideal choice of structural parameters \( \vec{a} \) for a given input image \( s_{\text{target}} \) and a given texture model are those which result in the minimal distance, i.e.

\[
\arg \min_{\vec{a}} D(s_{\text{target}}, s_{\vec{a}}).
\]

The compactness of the Gabor texture descriptor vector and the pre-processing of the colors allow to fully pre-compute and store a map

\[
\mathbb{R}^n \rightarrow \mathbb{R}^m,
\]

encoding the descriptor vectors for a dense sampling of the parameter space \( \mathbb{R}^n \). For moderately complex texture models, this sampling can be exhaustive in practice. Where more than \( 10^6 \) entries are required, a random sampling of the parameter space is applied instead.

As Gabor filter kernels for a given frequency are identical up to discretized rotation, this map also contains the descriptors of rotated versions of all structure images it stores descriptors for. Hence, while searching this map for the optimal match, the search for rotated versions is executed simultaneously by permuting descriptor values during comparison. Therefore no explicit sampling of texture rotations is required.

Most texture programs produce additional useful results when swapping the roles of the colors \( \vec{c}_1 \) and \( \vec{c}_2 \). Instead of encoding this behavior as a dimension of the structural parameter vector, it is searched simultaneously for both descriptors of \( s_{\text{target}} \) and \( 1 - s_{\text{target}} \). If the latter produces a solution with a smaller distance, \( \vec{c}_1 \) and \( \vec{c}_2 \) are exchanged. This technique cuts storage requirements in half in comparison to a solution with an additional parameter for all texture models.

Summary

Retrieval as sequential search in the sampled parameter space \( \mathbb{R}^n \) of the structure function:

- Pre-computed database of descriptor vector / structure parameter vector pairs \((d_{\vec{a}}, \vec{a})\).
- Exhaustive for moderately complex texture models.
- Otherwise random sampling with a database size of \( 10^6 \).

4.7 Results

Overall, a usage scenario is considered in which an artist is tasked with choosing a texture model and defining its procedural parameters for matching a reference
Figure 4.11: Results with automatic model selection for noisy textures. Target image sources: please refer to Section 7.4 Image References.

picture (such as a photograph or drawing). The developed retrieval technique provides substantial support for this task by suggesting the best-matching parameters for several possible texture models from a pre-selected class. Within a class, the results are presented in increasing distance $D$ to the reference.

The pre-selection of a model class is easy for the human artist to do and makes the descriptor more discriminative. As a consequence, its compact representation remains meaningful and comparable as the individual texture models constrain the search space.

The evaluated results represent several common texture classes and the procedures follow standard texture models available in commercial modeling packages, such as Autodesk’s Maya. In total implementations have between one and seven structural parameters, in addition to six scalars for the two color tones and one rotation parameter. These parameter counts are representative for both, in research [Lasram et al. 2012b], and for tools used in content production such as the count of structural parameters of Autodesk’s Maya texture nodes.

Figures 4.16, 4.17 and 4.18 give detailed description of parameter counts, value ranges, sampling densities and exemplary visualizations for each model.

The implemented texture models (names in cursive) represent these classes:

- **Noise** textures, including the classic Perlin noise, a turbulence noise, and a turbulence noise version with a ridge offset (perlin, turbulence, turbulenceridge) [Per-
lin 1985]. These models support anisotropic stretch and global scaling, and are suitable for a large variety of surface materials, including concrete, wood, rust, clouds or even vegetable skin (Figure 4.11).

- **Grids** of regular structures, with models covering grids of variable spacing, edge sizes and edge softness (*grid, cellular grid*) (Figure 4.12).

- **Tiled** textures, including a fake-shaded turbulent surface type such as found in a brick wall and a texture mimicking hardwood floor (*brick, woodplanks*) (Figure 4.13).

- Cut and polished **Wood** surfaces, implemented with varying base structures (*woodlines, woodplanks* and a one parameter model *woodstreaks*) (Figure 4.14).

- **Rings** with turbulent lines of variable width, as in some wood and marble-like structures (*rings*) (Figure 4.15).
4.7 • Results

<table>
<thead>
<tr>
<th>Target</th>
<th>1st Choice</th>
<th>2nd Choice</th>
<th>Target</th>
<th>1st Choice</th>
<th>2nd Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Brick" /></td>
<td><img src="image2.png" alt="Woodplanks" /></td>
<td><img src="image3.png" alt="Woodplanks" /></td>
<td><img src="image4.png" alt="Brick" /></td>
<td><img src="image5.png" alt="Woodplanks" /></td>
<td><img src="image6.png" alt="Brick" /></td>
</tr>
<tr>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
<td><strong>woodplanks</strong></td>
<td><strong>brick</strong></td>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
</tr>
<tr>
<td><img src="image7.png" alt="Woodplanks" /></td>
<td><img src="image8.png" alt="Brick" /></td>
<td><img src="image9.png" alt="Brick" /></td>
<td><img src="image10.png" alt="Woodplanks" /></td>
<td><img src="image11.png" alt="brick" /></td>
<td><img src="image12.png" alt="woodplanks" /></td>
</tr>
<tr>
<td><strong>woodplanks</strong></td>
<td><strong>brick</strong></td>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
</tr>
<tr>
<td><img src="image13.png" alt="Brick" /></td>
<td><img src="image14.png" alt="Woodplanks" /></td>
<td><img src="image15.png" alt="Brick" /></td>
<td><img src="image16.png" alt="Woodplanks" /></td>
<td><img src="image17.png" alt="brick" /></td>
<td><img src="image18.png" alt="woodplanks" /></td>
</tr>
<tr>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
<td><strong>brick</strong></td>
<td><strong>woodplanks</strong></td>
</tr>
</tbody>
</table>

Figure 4.13: Results with automatic model selection for tiles. Target image sources: please refer to Section 7.4 Image References.

4.7.1 Visual Quality

Noisy queries (Figure 4.11) yield good results for each of the texture models in many cases. Perlin does not support isolated peaks, and turbulenceRidge does not support very high frequencies. In these cases, anisotropic stretch is chosen as compromise, and the results are automatically downranked and better results from other models preferred.

As grid queries (Figure 4.12), the mixture of targets illustrates the behavior when abstraction of shapes is required. Neither of the grid texture implementations support diamond shapes; rotated rectangular grids are chosen as best-effort match. Both the grid and cellular grid implementations have a large underlying parameter space to search in; cellular grid illustrates the behavior for stronger quantization of scales, grid demonstrates random sampling of the larger space; as a result, exact reproductions of scales may not be possible; however, the resulting scales are always similar to the scales in the target image, and would only require minimal modifications by an artist to finalize.
Chapter 4 • Goal-Oriented Control for Interactive Parameter Retrieval

The tiled textures (Figure 4.13) woodplanks and brick implement models where spacing parameters need to be selected. In the presence of hard-coded design choices – not uncommon in production textures – a selection needs to be made between wood-like and stone-like surface appearance. Excepting the blue cartoon tiles, which are outside the appearance space of the texture models, grid spacings are robustly found. The optional phase-shift between rows and columns is not strongly represented in the texture descriptor.

For the wooden examples (Figure 4.14), implementations with small parameter space volumes are tested in addition to the wooden planks from the tiled textures. Even though woodlines has only two, and woodstreaks only one parameter which needs to be sampled, targets which are close to the appearance space of the textures are approximated well. In target images where additional structures are present, such as the fence or wooden planks, the orientation and scale of estimated structures provide plausible matches.

The rings texture (Figure 4.15) illustrates results for a complex texture, the parameter space of which cannot be exhaustively precomputed, requiring random sampling. The results with $10^6$ randomly sampled database entries match the structures of the target pictures closely. For an interactive application, a smaller sample with $10^5$ entries results in fast approximations an artist could use interactively.

An additional symbol texture model evaluates extreme mismatches between target query and texture model appearance space. The model follows testing purposes and
is rather uncommon in a production setting. One model arranges symbols from the Wingdings font, and a star model shows regular arrangements of stars of variable number of tips. As Figure 4.19 shows, structures are still recognizable even when the appearance space of the model and the target have no overlap. An impressive but lucky result constitutes the matching of the marguerites with an arrangement of flowers.

In total, in terms of visual quality the results show that the presented technique is successful: for each requested target picture, the preferred match is a feasible approximation of the input, requiring at most minimal fine-tuning by an artist for optimal results (note that all results shown are the direct, automatic result of the algorithms as-is, though). Also, the selected texture model approximates the input best from the models in the same class.

### 4.7.2 Sizes and Timings

Table 4.1 shows the size of the precomputed database for the tested texture models and the associated retrieval times. Limits for the scales of the models prevent the generation of structures which are either finer than the sampling limit or too large for multiple repetitions in the $256^2$ pixel texture window of the experiments. This window size is expected to sufficiently cover procedural models with any practical

<table>
<thead>
<tr>
<th>Target rings ($10^6$)</th>
<th>rings ($10^5$)</th>
<th>Target rings ($10^6$)</th>
<th>rings ($10^5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 4.15: Results for the texture class rings for different numbers of random samples in the database. Target image sources: please refer to Section 7.4 Image References.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>BrickX</th>
<th>BrickY</th>
<th>Mortar</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>1.1</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Step</td>
<td>0.009</td>
<td>0.009</td>
<td>0.1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PointDensity</th>
<th>Stretch</th>
<th>InnerWidth</th>
<th>OuterWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Step</td>
<td>0.2</td>
<td>0.6</td>
<td>0.05</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Density</th>
<th>GridH</th>
<th>GridV</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>1</td>
<td>0.25</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>0.5</td>
<td>0.01</td>
<td>0.01</td>
<td>0.5</td>
</tr>
<tr>
<td>Step</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Frequency</th>
<th>Stretch</th>
<th>Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>Max</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Default</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Step</td>
<td>0.01</td>
<td>0.1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Figure 4.16: Parameter counts, value ranges, sampling densities and exemplary visualizations for the procedural texture models brick, cellulargrid, grid and perlin. The renderings represent the min and max value of that parameter with all other parameter with their default value.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Tips</th>
<th>Radius</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Max</td>
<td>10</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Default</td>
<td>5</td>
<td>0.2</td>
<td>4</td>
</tr>
<tr>
<td>Step</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scale</th>
<th>Letter</th>
<th>Randomize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>80</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>50</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>Step</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Persistence</th>
<th>Stretch</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>0.3</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Step</td>
<td>0.05</td>
<td>0.1</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Persistence</th>
<th>Offset</th>
<th>Stretch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Default</td>
<td>0.3</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Step</td>
<td>0.05</td>
<td>0.05</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 4.17: Parameter counts, value ranges, sampling densities and exemplary visualizations for the procedural texture models star, symbol, turbulence and turbulenceRidge. The renderings represent the min and max value of that parameter with all other parameter with their default value.
Figure 4.18: Parameter counts, value ranges, sampling densities and exemplary visualizations for the procedural texture models woodlines, woodplanks, woodstreaks and rings. The renderings represent the min and max value of that parameter with all other parameter with their default value.
Figure 4.19: Results for mismatches between query texture and supported model appearance. Image sources: please see Section 7.4 Image References.

Table 4.1: Database dimension, size and retrieval times with single precision floating point values. The retrieval time excludes the computation of the target texture descriptor (about 0.6 s) and is reported for an Intel® Core™ i7-2600K CPU @ 3.40GHz.

<table>
<thead>
<tr>
<th>Texture Model</th>
<th># Entries</th>
<th>Db Size</th>
<th>Time / s</th>
</tr>
</thead>
<tbody>
<tr>
<td>brick</td>
<td>224 422</td>
<td>645.5 MB</td>
<td>0.80</td>
</tr>
<tr>
<td>cellular grid</td>
<td>101 871</td>
<td>293.0 MB</td>
<td>0.80</td>
</tr>
<tr>
<td>grid</td>
<td>$10^6$</td>
<td>2876.3 MB</td>
<td>7.79</td>
</tr>
<tr>
<td>perlin</td>
<td>12 221</td>
<td>35.1 MB</td>
<td>0.11</td>
</tr>
<tr>
<td>symbol</td>
<td>44</td>
<td>0.1 MB</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>star</td>
<td>16 830</td>
<td>48.3 MB</td>
<td>0.14</td>
</tr>
<tr>
<td>turbulence</td>
<td>23 331</td>
<td>67.0 MB</td>
<td>0.22</td>
</tr>
<tr>
<td>turbulenceridge</td>
<td>4 851</td>
<td>13.9 MB</td>
<td>0.07</td>
</tr>
<tr>
<td>woodlines</td>
<td>121</td>
<td>0.3 MB</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>woodplanks</td>
<td>20 402</td>
<td>58.6 MB</td>
<td>0.2</td>
</tr>
<tr>
<td>woodstreaks</td>
<td>11</td>
<td>&lt; 1 MB</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>rings</td>
<td>$10^6$</td>
<td>2887.7 MB</td>
<td>7.97</td>
</tr>
<tr>
<td>rings</td>
<td>$10^5$</td>
<td>288.8 MB</td>
<td>0.80</td>
</tr>
</tbody>
</table>

* random sampling
The parameter space is roughly perceptually uniformly distributed, for instance, by exponential scales for structure sizes. An automatic scaling, such as investigated by Lasram et al. [2012b], is outside the scope of this article. Also, the parameter space is discretized to create visible differences between stops on each parameter scale.

The precomputation time of the sampling ranges from one minute single thread CPU time for an exhaustive search on simple models to 1377 hours for a $10^6$ random sampling. The precomputation is parallelized, which constitutes a common optimization with industrial render farms. However, the performance would further benefit from a GPU implementation. These timings do not pose practical concerns, as they are only one-time investments. The resulting databases fit in-core at runtime and a retrieval time of well below 1s is achieved for an exhaustive search of the entire parameter space for most textures.

4.7.3 Applicability in a Production Context

To promote the transfer of novel research to an industry applicable version, the presented technique has been integrated into Autodesk’s Maya as a prototype.\footnote{This section has been published in: Gieseke, L., Koch, S., Hahn, J.-U., and Fuchs, M. Applying state-of-the-art parameter retrieval for procedural textures. In Proceedings of the European Conference for Visual Media Production, 2014a.} The results in Figure 4.20 show that such an integration is feasible.

To create a comparable representation of a Maya node, for example for a two-dimensional texture, the node is setup as image-plane of a camera and with no further adjustments Software-rendered. Any Maya node could be connected to the pipeline, but the derived feature vector for the comparison of the rendering is currently optimized for texture characteristics.

In a first step, the chosen parameter space of the Maya node is sampled as a one-time computation, best done in parallel with any common render farm. In the second
step, for the actual fitting, the retrieved structural parameters, rotation, and up to two matched colors are automatically set for the node.

The current implementation of the retrieval pipeline consists of Python scripts with a call of an executable for the feature vector computation. At this point, the retrieval times of below one second as in the original application are not archived within Maya, but can be further improved with code adjustments.

The prototype shows that the application of a state-of-the-art parameter retrieval technique is promptly applicable, proving an easy transfer from an academic context to an industrial one.

4.7.4 Material Parameters

The work presented so far was further developed by Grüner [2016] in his master thesis3, supervised by the author of this thesis. The master thesis investigates whether the interactive and example-based retrieval technique can be expanded to retrieve material parameters.

Grüner [2016] developed a “Example-based Parameter Retrieval for Procedural Materials through Abstraction of Visual Feature”. It is an interactive system that in addition to matching the structural parameters of a texture pattern also matches the parameters for geometric structures as well as light properties. As input a single image captured under uncontrolled conditions is used.

The problem of matching materials with unstructured input is not solvable because it is ill-posed. Grüner [2016] therefore proposes some assumptions to make a solution manageable and to constrain the problem. First, a single directional light source with a intensity of one is assumed, representing the dominant light direction on the image. Second, the technique is optimized for Lambertian surfaces with a non-varying surface albedo in an input image.

The system employs procedural models that add to the structural parameters of Section 4.6.1 light and height field parameters. A Gabor descriptor similar to the one of Section 4.3.3 is used for the retrieval of these structural parameters. Furthermore, for the Lambertian shading the albedo color and the diffuse and ambient coefficients need to be determined. As the distance to optimize for these shading parameters, Grüner [2016] interprets the comparison of two image histograms as the difference of their mean and standard deviation in each color channel and argues for retrieving them in the sRGB color space.

Figure 4.21 shows that Grüner [2016]’s technique matches structural and shading parameters well. The examples of different lighting further validate the structure of the computed surface. Figure 4.22 shows that the lighting direction is usually also matched well. All results are retrieved with multiprocessing within seconds.

---

3 Published in:
Figure 4.21: Results with varying lighting. Included are renderings of the surface under different lighting conditions in Example 1 and 2. Shown are matches for the model *Turbulence*. Figure adapted from Grüner [2016].

Figure 4.22: Example of matched lighting. Rotating the image leads to a rotation in the lighting as well. The direction is matched well, although the model itself is ill-equipped to approximate the geometric structure. Figure adapted from Grüner [2016].
In alignment with Grüner [2016]’s conclusion, it can be said that the interactive retrieval technique of Section 4.6 can be extended to include shading information in order to interactively match not only diffuse textures but also surface structure.

4.8 Limitations

The presented sample-and-retrieve approach is fundamentally different from continuous methods. They may apply non-linear optimization and interpolation methods [Bourque and Dudek 2004], and thus precisely find a local optimum in the parameter space. However, as discussed in Section 4.5.4, the numerical evaluation of the gradient of a cost function alone, involving several computations of an image difference function, may take more time than the entire retrieval step of this solution. Additionally a continuous approach needs at least locally continuous maps from the parameter into the distance function space. In choosing a discrete, sampled model, we believe to have found a compelling alternative.

The texture descriptor focuses on maintaining relevant texture features while enabling fast evaluation. As a consequence, it does not reach the full expressiveness, which could be expected from state-of-the-art texture analysis but follows the intent to construct a compactly storable descriptor vector. Modeling the distributions as Gaussians, disregarding phase sensitivity and covariances between the individual responses, may seem as a coarse oversimplification. Procedural models are, however, usually constrained in the effect of their parameters: a brick model, for instance, may offer expressive control on brick size, placement, and dimensions of mortar, but will not permit independent phase shifting of low and high frequencies at its edges. A representation of correlations between features on different scales and orientations would require scaling the descriptor vector length quadratically in the filter bank size, but could be expected to improve the automatic selection of texture models across the different texture classes.

A more general challenge in aligning automatic texture analysis with perceptual expectations of a human observer lies in semantic understanding: searching for a picture containing a happy smiley, the result shown in Figure 4.19 is retrieved. Objectively, it is a good match for the input in terms of density, line structure, global and local contrast. Nevertheless, taking the semantic associations of a human observer into account, the match can be perceived as greatly dissimilar. Nonetheless, this work presents a useful balance of accuracy and efficiency.

4.9 Outlook

As Gabor filters kernels are identical across scales, one could apply a similar strategy of transposing descriptor vector elements instead of storing descriptors individually.

---

4 Sections 4.8–4.10 (here extended) have been published in:
for each kernel scale. This would, however, require excessively large sample pictures for equal detail on all scales. Also, following the idea of pre-processing the color matching within the pipeline, it could be promising to estimate a continuous rotation before employing a structural texture descriptor. The sampling of a continuous rotation might give more accurate fittings in terms of the overall rotation and faster retrieval performance.

Individually implemented texture metrics for cluster of texture types, combined in a common pipeline could enable the application to determine a texture class automatically. This would be useful for computing the layering of different texture models into one rendering, determining the influence of each texture model to match a target.

Many textures, which cover an application-relevant space, are representable with two-tone models. Almost two-thirds of the presented target images come from online resources of the industry. Without general restrictions for the structural design of a texture model, the evaluated texture classes exemplify the large variety of classes, which the technique supports. Nevertheless, an extension to the texture model could be investigated which still allows for the separation of color and structure, but also spans a colorful appearance space beyond two-tone textures.

4.10 Conclusion

Based on a thorough evaluation of possible texture descriptors and optimization strategies, this chapter presents an interactive parameter retrieval technique for parameter sets of two-tone procedural textures. The achievements in improving the retrieval performance and the variability of the possible design space enable the applicability of the technique in a real-world production context. Artists are freed from a tedious initial exploration of the entire parameter space but can start with a close match to the design reference. It is then up to the artist to work on possible individual adjustments in the details. The procedural nature of the representation allows for flexible adjustments at any time.

While being a meaningful control mechanism in its own right, the presented solution also advances creative control for procedural representations, which is discussed in Chapter 2. The preceding chapter constitutes the building block of a creative control pipeline that enables efficient and transparent navigation by automating the purely rule-based manufacturing steps and contributes to opening up a possible design space. The example-based mechanism could be used with different target images for layers of a design, combined in various compositions, with the option of creating a novel and surprising result. It could also potentially be used for the programming of procedural patterns by example, a novel and complex concept that may fundamentally change the development of procedural representations.
Creative Control for Organized Order in Ornamentation

Creative control for procedural pattern generation needs to provide a flexible pipeline that combines various control mechanisms in a unified manner. This enables artists to develop individual creation processes and diverse outputs.

As thoroughly discussed in Chapter 2, ornamentation as a design goal offers a meaningful but challenging proving ground for tackling creative control. Before presenting a novel creation pipeline that comes one step closer to supporting the creative means of navigation, transparency, variability and stimulation (Section 2.3) within one framework, we briefly recap the context of ornamentation in the following.

One common class of decorative ornaments, as discussed by Wong et al. [1998], creates an underlying perception of order by placing individual components in a repetitive and balanced way. However, ornaments also include hierarchical structures, visually dominant elements and connections as accents. These often do not follow the underlying order of the ornament, breaking an otherwise overly homogeneous appearance. Additionally, ornaments adapt to the space they fill by aligning to its boundaries.

It takes an experienced artist to balance the contrast between carefully choosing visual accents and creating a sense of order by applying compositional rules and complementing the space, as shown in commercial examples Figure 5.1 and in the historic examples in Figure 3.1.

While artists are indispensable for the creative task of creating an overall layout and placing accents, executing structuring rules and completing an ornament into a

---

1 This chapter has been published in:
cohesive whole is tedious and worth automating. We propose a hybrid technique that gives artists artistic control through familiar tools like sketching while computing ordered structures automatically, unburdening the artists from tiresome tasks. We aim to offer both the control and quality of manual creation and the efficiency and accuracy of computation.

**Contribution 1** – An optimization strategy that incorporates customizable and modularly combinable placement functions putting global design constraints explicitly under artist control.

**Contribution 2** – A ready-made set of placement functions that fulfill design principles for ornamentation [Wong et al. 1998] with a balanced element distribution and symmetry constraints.

**Contribution 3** – The control of element connections through the translation of visual input into connection strategies under the given global design constraints, combining elements and connections into a cohesive whole.

**Contribution 4** – The use of path planning to efficiently route the ornament around obstacles.

**Contribution 5** – The incorporation and combination of control mechanisms at all scales, ranging from taking high-level guidance from example images down to placing single elements and making local edits within the computed ornament while maintaining its procedural nature.

**Contribution 6** – A designer survey that evaluates the usability of the technique and summarizes the designers’ thoughts on the involved creation processes. The feedback confirms the appeal, efficiency and further potential of the presented methods.

### 5.1 Related Work

For a detailed discussion of ornamental models and their control, please refer to Chapter 2 Analysis Framework for Creative Control.
Figure 5.1: Commercial examples of hierarchically ordered ornaments. They balance accent and order and complement the space they fill. Image sources left to right: [Colourbox 2011, 2016, 2013a; Rejke 2011; Colourbox 2013b].

In the following we summarize and contextualize the review of the related work for the different steps of our procedural ornamentation pipeline. Please also refer to Section 3.4 Analysis of the State of the Art for a discussion of further work in this context.

The pioneering work of Prusinkiewicz [1990] applies L-systems to algorithmically model plant growth. Various extensions [Prusinkiewicz et al. 2003; Parish and Müller 2001] demonstrate their expressiveness for different applications. However, since the execution process is inherently hierarchical, L-systems have difficulty supporting artistic control mechanisms that range from global to local scale.

Wong et al. [1998] introduced a programmatically controllable procedural system that employed a greedy rule-based strategy to generate floral ornaments. We take inspiration from their work but focus on enabling a usable tool by adding artist-friendly control mechanisms in a unified way. We generalize and technically improve their space-finding algorithm, enabling the explicit enforcement of ornamental principles and unburdening artists from implementing them for each pattern individually. Their method can only control the connections of elements by writing code; we add to their work by giving intuitive design options for directing connectivity. This, combined with the option to place single elements freely, enables ornamentations with hierarchical structures that go beyond repetitive patterning. Anderson and Wood [2008] adapted Wong et al.’s work by placing discrete elements along a user-given curve. However, they solely decorate the regions adjacent to the curves, offering only limited control and design options. Etemad et al. [2008] pick up a rule-based strategy for a dynamic recreation of Persian floral patterns, focusing on animation, not on controllability.

Beneš et al. [2011] offer certain global control on the procedural process by dividing a target space into user editable guide shapes. The shapes determine what types of patterns grow in different areas. The connections between the shapes are manually specified by the user and in turn guide the connections between elements. In our approach, element connections are automatically derived from visual guides. Other systems provide global control on the procedural process by interpreting the modeling task as a probabilistic inference problem [Ritchie et al. 2016, 2015; Talton et al. 2011; Šťava et al. 2014], or optimize a packing problem under specific constraints [Chen et al. 2016b]. They all control the overall shape of the resulting ornaments, but
do not permit the hierarchical or element-level local controls we support, such as specifying the locations of individual elements.

Various example-based approaches [Ma et al. 2011, 2013; Ijiri et al. 2008; Bradley et al. 2013] give artists indirect control over the resulting pattern by defining an exemplary element arrangement a priori. They extract the spatial relationship between elements and attempt to reproduce the relationship in their synthesis results. But their methods do not allow placing accents with single elements, which can break up the perception of a homogeneous texture and helps to create a compelling ornament.

Ijiri et al. [2008] use sketch-based user input to create global control structures such as an underlying vector field to guide procedural growth. We also make use of vector fields generated from artists’ sketches, but we use them to establish element connectivity and to guide to growth of a model. Vector fields are further employed in the specific context of procedural street modeling [Chen et al. 2008a] and micrographics [Maharik et al. 2011]. They have been integrated into formalized grammars, such as a vector-field guided shape grammar [Li et al. 2011] and L-system rules [Šťava et al. 2010]. These systems produce patterns that are more homogeneous than the decorative ornaments that we aim for. Xu and Mould [2015] trace shortest paths in vector fields to generate branches for botanic tree modeling. Aiming for a different application, we support global design goals and, in addition, plan growth paths of ornaments around layout obstacles.

Our different types of user input are inspired by painting-tool-like methods in procedural modeling [Chen et al. 2008b; Palubicki et al. 2009; Méch and Miller 2012; Chen et al. 2012; Emilien et al. 2015], in particular by the flexible Deco procedural engine [Méch and Miller 2012] upon which we constructed our implementation. We extend these methods by combining the ability to follow input curves while taking the whole environment into consideration, applying the chosen design principles when placing elements on the paths as well as when automatically filling the remaining space.

None of the work discussed so far integrates artist control on an element and connection level once the pattern is computed. There are procedural techniques that enable low-level control on the results themselves, developed in the context of architectural designs [Lipp et al. 2008], tree modeling [Pirk et al. 2012] and the creation of natural scenes [Emilien et al. 2015]. The move operator from the latter is similar to ours but their system is optimized for chaotic arrangements, which contrasts to our organized design goals for ornaments.

5.2 System Overview

Since our approach extends the technique introduced by Wong et al. [1998], we briefly summarize their approach. A procedural model is created with artist-defined elements and a set of growth rules that handle the selection of elements, their appearance and connections. The process iterates, finding tentative places for elements by testing them against constraints in the procedural model, and, where suitable, placing elements in the found spaces, optionally connecting them to existing
elements. Possible ornament designs are technically restricted only by this iterative creation logic.

Wong et al. [1998] find the next space to fill by computing the medial axis of a shape stencil using the Manhattan distance, then inflating circles centered on points on the axis until they collide with geometry proxies or the stencil shape. The new element is placed at one of the circles of maximal radius (Figure 5.3, top row) and connected to the closest existing element. The system then updates the distance map to incorporate the distances to the proxy geometry of the newly placed element and repeats.

Our approach performs a similar greedy iterative process but generalizes it by using placement functions. Wong et al.’s technique of placing the next element into the largest possible space is one possible placement function, but there are many others. As we aim for direct interactions, performance is crucial, so we furthermore modified their implementation to allow interactive performance.

Figure 5.2 shows an overview of our technique. As a first step, the system can be configured to express global design goals by using algebraic placement functions. Higher values of the functions indicate preferred locations for element placement (see Section 5.3). Placement functions have a variety of potential inputs, such as a stencil defining the areas to fill, or a desired type of symmetry.

The artist specifies these design constraints through our interface and can also provide other input such as sketched paths to guide the connections between placed elements (see Section 5.4). Based upon the input, we configure and combine a set of ready-made placement functions that implement the artist’s intent. The artist can also specify exact locations for certain elements by directly placing them in the space to be filled.

Our automated placement system then repeatedly evaluates the placement functions to find the locations with maximal values, and inserts elements into the output ornament accordingly. The artist can at any time interrupt the process and make changes using editing techniques like moving or deleting existing elements. The system immediately adapts to these local changes.

The following sections give more details on our process. For coherency, all our models were designed by the same artist and thus share visual traits specific to the artist’s individual style. By presenting comparable models with similar growth rules, we aim to highlight the variety of possible designs implemented by the system, not the model’s hard-coded rules.

5.3 Placement Strategy

Artists start by specifying global design goals for the automatic placement of elements. Some of these goals, like desired growth direction, take additional input through lower-level mechanisms like sketching.

Our set of supplied global designs aims to fulfill more explicitly underlying aesthetic principles of ornamentation. Wong et al. [1998] thoroughly discuss these but their method only indirectly and uncontrollably implements them. They summarize
Figure 5.2: Our technique can be configured to incorporate global design goals such as symmetry. The artist can optionally place specific elements and draw desired connectivity. At any time during the process, the artist can insert, delete and move elements on the canvas and the system adapts to the change.

the aesthetic principles of ornamentation as repetition, balance and conformation to geometric constraints. We support balance and repetition through symmetry constraints and facilitate conformation to geometric constraints by element connection strategies (see Section 5.4).
5.3.1 Placement Functions

We strive to support a wide variety of design goals with few limitations. For this, we modularize the creation of global order for placing elements through the definition of placement functions. A placement function $p : \mathbb{R}^2 \rightarrow \mathbb{R}$ takes higher values at preferred placement locations. $p$ is updated after every element placement and its values decrease. Once $\max \{p\}$ reaches or falls below 0, the ornament is completed.

We provide a number of fundamental placement functions, and the system combines them into one overall placement function based on the user-specified design goals. The functions can make use of user-supplied input like shapes, images, and paths. They can also use internal data structures like the locations of the centers of placed elements, and a map of rasterized proxies for them.

Our supplied placement functions implement:

- A stencil function accepting a binary stencil that defines the area to be filled.
- Symmetry functions for supporting different types of symmetry.
- Image data functions accepting a grayscale image that controls the desired element placement based upon image brightness.
- Path functions supporting element placement along paths.

In addition, users with scripting experience can readily extend the system by adding new placement functions. We provide a number of functional building blocks like $\min()$, $\max()$, $\text{translate}()$, and $\text{rotate}()$, which can be combined using the usual mathematical function operations and can be extended with custom code. These new functions have the same access to user input and internal data structures as our supplied placement functions.

While placement functions provide the framework for unifying and combining many sorts of design constraints, their presence is completely invisible in our sample interface. Users specify constraints and guidance in conventional ways like sketching and choosing among options. The system feeds their input to the placement functions and combines the functions to implement the desired constraints.

The following subsections briefly discuss the construction of the ready-made placement functions and how they interact to support design goals.

The Stencil The stencil function is a simple function that takes as input the value 1 for areas to be filled and 0 elsewhere.

Symmetry Wong et al. [1998] place elements by maximizing the distance between the new element to both the stencil border and previously placed elements. Figure 5.4, left, shows that this strategy when applied to a symmetrical stencil can lead to an ordered, highly symmetric ornament. In the example, the algorithm places elements on the symmetry axes of the rectangular stencil shape. The elements partition the space so that following insertions maintain the symmetry. As a result
Figure 5.3: The evolution of the placement function with an initial manually-placed element. The placed element proxies are in blue and the green dot indicates the proposed next location, with the permissible maximal radius in yellow. The gray levels in the map correspond to placement preferences, with brighter values corresponding to preferred positions. The top row shows a placement without explicit symmetry, the bottom row with explicit four-way symmetry.

of the greedy search for maximal spaces, larger elements are placed before smaller elements, creating a visible order hierarchy that is characteristic for many ornamental styles.

However, if the artist pre-places an element off-center, or a non-symmetrical stencil is used, or models contain randomized characteristics, applying this strategy without modification leads to unorganized patterns, as shown in Figure 5.4, center. Figure 5.3, top row, shows the sequence of element insertions for this case. To solve the issue, we allow the artist to explicitly express symmetry in a way that can integrate pre-placed elements for any stencil, as shown in Figure 5.4, right, and Figure 5.3, bottom.

The desired symmetry is supported by a placement function that implements a set of heuristics that work as follows:

1. If an element was placed at some location \((x, y)\), we prefer filling its transpositions under the symmetry transformation, placing the new elements in mirror or rotational symmetry. Hence, the placement function should have the highest values at transformed locations of previously placed elements.

2. If the placement at such a location is not possible for instance, because the symmetry set is already complete), we have some freedom. In addition to keeping a maximal distance to the stencil, we prefer placing the next element at a location that permits future elements to be placed at symmetry-creating locations without collisions with existing elements.

3. Finally, we exclude placements that are too close to existing elements or outside the stencil.
Figure 5.4: Greedily placing elements to maximize the distance to the frame and previously placed elements can sometimes generate tilings with high symmetry, shown left. With a single artist-placed first element, marked in red, this approach breaks down, shown center. By making the desired symmetry explicit, we can give artists the option to manually place elements while maintaining the symmetry, shown right. Unless otherwise noted, none of our results were edited locally after the computation.

Figure 5.5: Symmetry generation examples for different symmetry types with the same pre-placed elements in red. From left to right, reflection across an axis, reflection across the center point, three-way rotational symmetry using a three-way symmetric stencil, four-way rotational symmetry.

The bottom row of Figure 5.3 shows the mechanism working for four-way mirror symmetry. With the chosen settings, the first three insertions fulfill symmetries of the pre-placed circle according to heuristic 1 and the next two circles are placed following heuristic 2. Our interface allows the user to choose among various symmetry types; Figure 5.5 shows several results.

**Controlling Placements with Image Data**  To show the flexibility of our system, we provide a placement function controlled by an artist-specified example image, as shown in Figure 5.6. This function uses the brightness of the image to prioritize placement order and also uses it to guide the selection of elements. A cut-off for the size leaves the black parts of the example empty.

**Paths**  To place elements along an artist-specified path we interpret it as a stencil that sets all points off the path to 0. This forces the greedy placement process to put elements along the path according to all other given global design constraints,
Figure 5.6: Demonstrating the versatility of placement functions, gradient images in the middle are used to guide the ornaments, with the size and type of placed elements determined by the gradient’s brightness.

for example with symmetry constraints, as shown in Figure 5.7. After the potential positions on the path are exhausted, our algorithm completes the remaining background. These paths also affect element connections, as described in the following section.

Summary

Placement functions control the iterative space filling by defining preferred placement locations.

- Their generalized setup allow for any design goals with few limitations.
- Placement functions have access to artist input and internal data.
- Basic placement functions can be combined to create individual designs.
- Fundamentally new placement functions can be added by custom code.

The given exemplary placement functions implement stencils, symmetry, image data as input and element placements along paths.
Figure 5.7: Space is filled in an ornamental manner to complement an artist’s input, shown left. Stencils are indicated in gray and paths in blue. In the second row, the red flowers were manually placed. In the two bottom rows, elements are connected along artist-specified paths. Note that the paths were drawn quickly by hand or with a rectangle tool and the spacings are not optimized.
5.4 Connection Strategy

Spatial relationships between elements\(^2\) are often expressed by geometric curves or patterns that connect nearby elements. These interconnections add additional levels of order to an ornament in a structured way. Furthermore, global layouts, such as a frame around a text box, can be achieved by appropriate interconnections. In addition to singular global structures, such as specific frames (see Figure 5.7, bottom row), many complex real-world ornaments adapt themselves in their entirety to global designs, such as in Figure 5.8, middle row, where the whole space is structured with multiple lines.

In the work of Wong et al. [1998], the connections between elements arise from individual, model-specific growth rules. Our method also includes these programmatic growth rules but we add direct visual controls to define connectivity by drawn paths or sketched vector fields. This approach permits greater control than related techniques, as the artist-defined placement functions also guide the positioning of the elements on the paths.

5.4.1 Placement on a Single Path

Once all elements have been placed along an artist-specified path according to the global design constraints, we sort the elements according to their distance to the path’s starting point and connect neighboring elements to express the path. This connection strategy also works for self-intersecting paths, where a simple proximity-based strategy might fail.

5.4.2 Connecting Elements using Vector Fields

A single user-defined path is not sufficient to enforce the connectivity in a large area. While artists could carefully fill the entire space with paths by hand, we resolve this tedious task by letting them sketch a vector field, which structures a plane by storing orientation information at each plane position. The field can be created from a rough sketch or from an example image. We extrapolate sketched guides to dense vector fields by applying the method of Maharik et al. [2011]. For images, we compute the gradient and rotate it by 90\(^\circ\) to find the orientation of its isolines. The streamlines of the vector field form a natural, dense and connected organization of the ornament space. Streamlines are found by picking a seed point and tracing out a path with vector field integration.

In order to construct the ornament, we either pick a maximal-length streamline or start at an artist-given seed point. The elements are placed on the traced streamline as if it were a user-specified path (see Section 5.4.1). Then, we mask the area around it with a stencil (see Section 5.3.1) and zero out the corresponding areas in the

---

\(^2\) Sections 5.4–5.6 are basis for the patent:

Figure 5.8: These examples demonstrate the application of vector fields, the directions of which are indicated by the green arrows. Top row, the streamline of the field are traced iteratively. Middle row, a complex vector field is computed based upon contours from an input image. Bottom row, drawn paths define the connectivity of the foreground elements, but the vector field guides connections in the background, not with streamlines but by only allowing limited growth in the direction of the field.
Figure 5.9: Path planning around obstacles. Minimizing just the geometric path length makes the ornament avoid the obstacle (black line), but the path follows the obstacle’s shape only on the upper side, shown left. We prefer it to follow it on both sides, which we achieve with modified edge costs that pull the path towards the straight line between the endpoints, shown in the center. The path is pulled in between the lower obstacles, aligning the path better to the obstacles, shown right.

vector field. We repeat this process with the next-longest streamline, or the next seed, and iterate. Similarly to placing larger elements earlier than smaller elements, following longer stream lines before shorter ones contributes to the hierarchical space organization.

Figure 5.8, top and bottom row, show the synthesis results guided by vector fields constructed from sketched input and from an example image, middle row. A vector field can also be applied, if so desired by artists, to guide model specific growth characteristics. Figure 5.8, bottom row, shows background elements that are only allowed to grow in the direction of the underlying vector field.

Summary

<table>
<thead>
<tr>
<th>Connections create relationships between elements and enforce visual hierarchies and the conformation to geometric constraints.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• We interpret the creation of connections as path planning based on artist input.</td>
</tr>
<tr>
<td>• With the help of vector fields connections can be designed by the streamlines of the field. Additionally, the directionality of the field can influence growth characteristics of a pattern.</td>
</tr>
</tbody>
</table>

5.5 Resolving Collisions

As ornament designs are often part of a complex layout, an artist can specify geometric constraints either with stencils or by drawing obstacles. This might lead to an intersection of the defined paths and obstacles. To enforce the perception of an
ornament adapting to the space it fills, we guide element paths around intersecting obstacles. We interpret this as the classical shortest-path problem between the start and the end of the path. A permissible path can be found by any shortest path algorithm, where the underlying graph has the image pixels as nodes and 8-connections to non-obstacle pixels as edges. Using Euclidean distance between the adjacent pixels as edge weights leads to a short path. However, this path may not follow the user-intended shape (see Figure 5.9, left). Therefore, we add the distance between each node and the direct path, which ignores obstacles, to the movement costs from that node to the goal; see Figure 5.9, center and right. Though the resulting path might not be smooth, the size of the elements of the ornament in comparison to the jaggedness of the path at the pixel level empirically compensates for the roughness.

Summary

Path planning around obstacles with modified edge costs, which integrate the distance to a direct and unobstructed path, allows for greater alignment to the shape of the obstacles.

5.6 Local Editing

At any time during the computation process, or after the ornament is completed, the artist can directly interact with the elements of the pattern and their connections. Artists are familiar with manual interactions such as placing an element and moving it around on the canvas as part of their everyday workflow. By adopting these methods we narrow the gap between procedural modeling and manual creation.

Internally, we keep a directed graph of the ornament in which nodes represent elements and directed edges their connections. After each interaction the graph structures are updated. Connections that violate rules and orphaned elements are detected and reconfigured. Therefore the model adapts itself to the changes and keeps itself consistent with the growth rules of the model. This retains the powerful interaction capabilities of a procedural model, such as changing specific element characteristics for all samples at once, to the artist.

Specifically, for the local editing we offer deleting elements and/or their connections and picking up and moving single elements. New elements can also be added. Artists can move or add elements freely without influence from the underlying placement function, and we eliminate overlaps by deleting any elements or connections that intersect. As the artist moves or inserts an element we adapt the ornament to the changes interactively, recomputing its connections according to the rules of the model — for example connecting to the currently closest element. If elements are deleted the space remains empty, but to keep the model intact connections between remaining elements and the now deleted element are reconnected according to the connections rules of the model. See Figure 5.10, top.
Figure 5.10: The top row shows that after the manual deletion of the red flower in the middle, the stems that were connected to it are re-connected and aligned to the new connections. In the bottom row an artist created the second design on right solely by deleting an moving single elements in the original on the left. This is the only figure in which our results were edited locally after the computation.
Summary

The artist can make *local edits* such as the moving, deletion and insertion of single elements manually. The system adapts itself accordingly and maintains all procedural rules. Local editing not only allows for minor tweaks of the pattern but also for exploring variations of the pattern resulting potentially in fundamentally different designs.

5.7 Designer Feedback

To validate our approach, we performed a study to collect high-level feedback on our methods and evaluated their general strength and weaknesses. Because design tasks take considerable expertise, we sought primarily qualitative feedback from designers, in accordance with evaluations of similar techniques [Kazi et al. 2012; Nakagaki and Kakehi 2014].

Eight participants took part in the study, all of whom considered themselves to be professional designers. Six were students with a course of study in audio-visual media. Four of the participants rated their design knowledge as *Intermediate*, three as *Advanced* and one as *Expert*. As consequence of the small sample size we do not analyse a generalizable significance of the quantitative results with inferential statistics. Instead we focus on the qualitative analysis of the feedback.

The study took about an hour and consisted of a brief explanation of ornaments, a task comparing our results to related work, a tutorial session with our tool, three tasks to complete with our method, and an optional task with Illustrator for the participants that rated their Adobe Illustrator knowledge at least *Intermediate*.

5.7.1 Evaluation

Before knowing about our methods we had the participants compare three ornaments computed with our technique with three of Wong et al. [1998] as baseline (Figure 5.11). For the computation we used Wong et al.'s original implementation to embed the algorithm in our framework. We chose to compare visual quality and not the usability of the methods, as most designers, having no or little programming skills, would not be able to use Wong et al. [1998]'s system. Instead we compared the use of our system to Adobe Illustrator, which is a preferred tool for designers practicing the art.

For the visual comparison, our results were generated with underlying horizontal and vertical symmetry and had no local editing applied. Six of the eight participants preferred our results based upon their more symmetrical appearance and more ordered structure. The explanations for favoring Wong et al. [1998]'s results were “...more balanced in terms of the colours” and “I prefer result set 1 [Wong et al. [1998]], as there is less order”. Neither method specifically handles color, so for both
the color distribution is uncontrolled. The second comment indicates that it might be worthwhile to make the degree of order adjustable – another participant later said, in contrast to this, “More symmetry would have been great...”.

The survey included 17 Likert-scale questions about our tool in general, specific methods, and a comparison to Illustrator (answered by the 6 of the 8 participants who had sufficient Illustrator knowledge). The quantitative evaluation shows an overall approval of our system (Figure 5.12). However, the Likert-scale questions were mainly intended to motivate further comments on the topics in an open-ended fashion. Additionally, we asked the participants “What did you like about the tool?”, “What did you dislike about the tool?”, “Any ideas for new/missing features?”, and “Any further comments?”. We clustered the answers by the number of times that a specific topic was mentioned.

For positive feedback the most common comments were that our methods save time (mentioned 6 times) and are easy to use (4 times). There was praise for the general concept (2) and that it enabled an artist to explore designs (2). For specific methods, local editing (2) and the application of a vector field (2) were mentioned, with one comment saying “In particular I enjoyed the flow fields as it felt that they allow me to orchestrate the picture on a higher level.”

In terms of negative feedback, most arose from missing feature implementations (6) and the consequential lack of control (4). For missing features there were many requests for convenience features (6) such as undo functionality, grid alignment or better previews of actions. Further control of element and path characteristics, such as their size, was also desired (4).

From our analysis of the Likert-scale numbers and the open-end answers, we conclude that our methods were well-received overall. They are effective at saving creation time and effort, and the participants all agreed that they are fun to use, even more so than the known tool Illustrator. Our results hold up well in terms of visual quality when compared to Illustrator, with the mean of the responses slightly preferring the tool’s results to their Illustrator work. All negative comments were in regard to missing feature implementations; no one questioned our overall concepts. Adding more specific functionalities would also improve the controllability of the results. Nonetheless, one participant even said “I also liked that it gave results that probably would not have been my first choice, but might serve as inspiration for further exploration. Like looking at nature, processes out of our control can give new input into our own designs.”

Summary

The designer feedback shows that our methods were well-received overall. They are described as

- effective at saving creation time and effort,
- fun to use, and
Figure 5.11: In the top row three results computed with the technique from Wong et al., in the bottom three results from our technique with a fourway-symmetry and no manual adjustments. The images in each set differ because this model has many randomized features, such as the shape of the flowers or the stems. This randomization leads to drastically different results for Wong et al.’s technique. As they offer no control mechanisms, the only option an artist has is to execute the algorithm until a favorable design is archived. During the feedback session, 6 out of 8 designers preferred our results.

- hold up well in terms of visual quality when compared to Illustrator.

However, many missing feature implementations were noted, such as an undo functionality.

5.8 Discussion

5.8.1 Performance Discussion

A novel implementation for finding the next optimal insertion location, results in a total runtime of a few hundred milliseconds to fill $512^2$ pixels without symmetry constraints, a few seconds with symmetry, and up to several minutes for the most complex examples.\(^3\)

Our placement strategy has an innermost loop that repeatedly finds the next optimal insertion location after each element has been placed. To accelerate the process, we keep a rasterized version of the placement cost in memory and store it as an image pyramid $P$, in which each pixel stores the maximum value of four pixels on

\(^3\) This section is primarily attributable to M. Fuchs.
Figure 5.12: Quantitative evaluation of the Likert-scale questions. A box represents the second and third quartiles, with the red line indicating the median and the whiskers including a range of [(Q1-1.5 IQR), (Q3+1.5 IQR)], with IQR being the interquartile range. Values outside of that margin are indicated with blue + markers. The questions are categorized as applying to the tool in general, shown in red, to specific methods, shown in blue, and in comparison to Illustrator, shown in green.

the next-lower layer as well as the coordinate of the maximum pixel on the source layer. Thus, its single pixel at the top equals \( \max_p \), with \( p \) being the placement function.

Whenever one of the building blocks of \( p \) needs updating, we identify the changed region, track it through all its transformations, and recompute \( p \) only for the affected area \( A \). Then, we update \( P \), starting from the bottom with the pixels in \( A \), and recursively work our way to the top until no more updates are required or \( \max_p \) has been recomputed.

With this strategy, placing the first few elements is slow, as the entire function changes, potentially changing the location of the maximum. But subsequent elements are placed faster and faster, as the following discussion of the asymptotic runtime shows.
Asymptotic Placement Performance

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

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

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

(5.1)

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

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

(5.2)

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

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

$$O \left( \sum_{i=1}^{m} \left( \frac{n}{i} + \log n \right) \right)$$

$$= O \left( n \cdot \sum_{i=1}^{m} \frac{1}{i} + m \log n \right)$$

(5.3)

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

5.8.2 Creative Control

The control mechanisms and creative means of the presented framework are summarized in Table 5.1. In comparison to the discussed state of the art (Section 2.2), the framework includes the largest number of controls, with a focus on controls relating to the real world, such as curve drawing on the canvas.

The amount of controllability in turn improves the creative means, offering straightforward navigation, design variation and even some stimulation with the automatic adaptation of the pattern to manual edits, for example.
Table 5.1: Interaction and creative means for the presented procedural ornamentation framework. Please note that the configuration requirements are optional, as the provided exemplary configuration can be used out of the box.

<table>
<thead>
<tr>
<th>CONTROLS</th>
<th>INIT.</th>
<th>EXEMP.</th>
<th>PARAM.</th>
<th>HAND.</th>
<th>FILL.</th>
<th>GUIDE.</th>
<th>PLACE.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Configuration</td>
<td>Initialization</td>
<td>Image</td>
<td>Arrangement</td>
<td>Element</td>
<td>Visual Output</td>
<td>System</td>
</tr>
<tr>
<td>Gieseke et al. [2017]</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREATIVE MEANS</th>
<th>NAVI.</th>
<th>TRANS.</th>
<th>VARI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive</td>
<td>Control</td>
<td>Quantity</td>
<td>Navigation</td>
</tr>
<tr>
<td>Gieseke et al. [2017]</td>
<td>•</td>
<td>•</td>
<td>o</td>
</tr>
</tbody>
</table>

The transparency of the navigation still needs some improvements. Not all effects of the interaction are fully understandable, controls have overlapping effects and there is no navigation history. Even though such unpredictable behavior can at times be potentially stimulating, it should be possible to switch between full controllability with a clear understanding of cause and effect and inspiring exploration.

5.9 Limitations

Element groups in symmetric locations are occasionally constructed from different element types, especially if the elements are rather small. This limitation arises from our placement functions being processed on a discretized pixel grid. Accordingly, elements can not be placed at exact locations in the plane, but only at integer pixel coordinates. The example of three-way-rotational symmetry in Figure 5.5 shows the limitation: some of the smallest elements are not in perfect symmetry to copies of the same type, but to smaller elements. The reason for this lies in the way the placed element groups are constructed: while the first element may fit well into a particular place, the next may be partially occluded due to rasterization artifacts, and therefore a smaller element takes its place instead. The same problem arises from models including variability, such as the strawberry model, in which the types and sizes of the elements are randomized (Figure 5.7, right).

Aesthetics, even in the special case of ornaments, are subjective. We define our set of example placement functions to create ornamentation based on design principles found in related work. Nonetheless, perceptions differ and, as mentioned in the analysis of the expert feedback, while one participant prefers less ordered results, another one strives for absolute uniformity. Resolving this issue would require even more controllability, ultimately for all characteristics at all times during the process. This would have to be balanced against decreasing the ease of usage.
5.10 Outlook

In this work, we addressed the problem of placing elements according to global design constraints on their position and size. As future work, further visual properties could be constrained, such as the orientation of individual elements to better satisfy symmetry. Controlling the overall color distribution would also be worthwhile.

Our greedy method of placing the next element based on the placement function does not consider connections when computing the next space to fill. Here a global optimization or distribution strategy, taking the connections of the elements also into consideration, might be an alternative, but possibly at the cost of the interactive performance we aim for. Further research on this is called for, but beyond the scope of our paper.

After thorough testing of the local editing feature and a preliminary run-through with a designer, we deliberately chose the ‘what you see is what you get’ principle, as changes, for example to element positions, that were not directly triggered by the artist are hard to anticipate and the system would lose controllability. Exploring this trade-off is future work. We also would like to explore an idea proposed by one study participant regarding moving elements: rather than deleting elements that overlap the new location, push the elements around as with a mass-spring system.

This would be interesting in combination with our global design constraints and could be especially promising with an underlying vector field, letting the elements be pushed in a meaningful direction.

The implementation of the procedural ornaments themselves requires programming skills. Hand in hand with our contributions regarding the usability of the models, it would be equally worthwhile to investigate a more artist friendly creation processes for the underlying procedural models.

Lastly, Li et al. [2011]’s work applies their grammar in 3D space, a desirable extension for which we are aiming in the future.

5.11 Conclusion

The technique defines a general ornamentation framework that brings user interaction to a task that is currently either fully automated or fully manual. The pipeline stands out in comparison to related work by its extent of controllability and the resulting creative means. The uniform approach supports control on various levels of abstraction. It puts global design constraints, such as symmetry, explicitly under artist control, interrelated with visually specified input, such as strokes, down to control at the individual element level. The technique confirms the great potential of integrating the artistic control mechanisms that artists use every day into a procedural system. We hope it will inspire others to explore this direction further – for example, for the creation of the procedural models themselves.
Artistic control for procedural pattern generation makes it possible to create surprising and individualized results. Meaningful exploration and variability of a design space should stimulate artists and support them in their individual creation process. Artists often do not start with a specific goal or design in mind but intuitively navigate a design space until a result feels right to them.

Because such a creation pipeline needs to be highly flexible, it is especially challenging to integrate automatization or a procedural representation. Nevertheless, there is compelling potential in doing so – not only for reducing non-creative manufacturing efforts and for enabling artists to fully immerse themselves in creative tasks but also for computing inspiring triggers.

For this chapter, we are motivated by the ambitious objective of supporting artistic work and cover an exemplary first step toward artistic control mechanisms. We start by investigating an artistically driven parameterized algorithmic model in a carefully confined scope. Even though we present a model that feels artistic in comparison to being a texture or an ornament, we also aim for a pattern-like quality, embedding it in the overall context of this thesis. In order to do so, we first briefly analyze in the following what it could mean for a model to be considered artistic. Hence, in this chapter we lay the groundwork for future work that focuses on artistic control.

---

1 This chapter has been published in:

For this publication the implementation of the pipeline is primarily attributable to S. Klingel and the survey to L. Gieseke.
mechanisms. However, we concentrate less on the artist’s point of view. Instead, we first investigate the validity and effect of an artistic algorithmic model. Because producing a result with our model currently consists of multiple steps for an artist, we call this model parameterized, rather than procedural. A procedural model commonly implies a single underlying procedure.

We propose a method to automatically deconstruct an image into visually coherent constituents and to rearrange those pieces in a surprising, aesthetically pleasing and potentially informative fashion. Changing one’s perspective regarding artwork enables a deeper understanding of an examined subject and gives insights that might have been missed in the original. Our pipeline is adjustable, and artists can in part individualize their desired artistic expressions. We show with a survey that the visual appeal of the results vary in regard to the chosen parameter combinations. Lastly, we showcase a variety of examples that explore the design space and show that a reconfiguration in and of itself constitutes a new piece of art.

The automatic segmentation of images into meaningful components is challenging and is part of the fundamental problem of making algorithms understand what they see. By now, there are a variety of successful segmentation solutions that deconstruct an image into visually meaningful constituents. The usefulness of these techniques inspires the inversion of the original problem statement to whether we are able to employ the algorithms to make humans understand what they see differently.

In our everyday life we interact with multitudes of images. Our subconscious processes the visual input, and in matter of milliseconds we believe we know what we see. However, what the brain processes as seen is largely filtered and depends on predetermined conditions, such as personal experiences. To break this routine the visual data needs to be edited. A meaningful manipulation of the input enables a change of perspective regarding what was thought to be a given. Changing perspective and giving new insights is characteristic of artistic expression; indeed, art makes an observer experience the world differently. For our work, we are greatly inspired by the artist Urs Wehrli, who focuses on the idea of creating art by manipulating the constituents of an image by visually tidying them up in a semantically surprising fashion [Wehrli 2003].

Following Wehrli’s example, our automatic pipeline makes use of current algorithmic capabilities to offer an observer the experience of seeing input images differently but still in an aesthetically enjoyable way. By applying an automatically computed deconstruction and reconfiguration of the constituents of the image data, we open up a new point of view for an observer. This process of breaking an image apart and putting it back together again not only enables a better understanding of the image but also creates something entirely new.

Figure 6.3 summarizes our pipeline and Figure 6.1 shows the achieved aesthetic.
Figure 6.1: The input image is deconstructed and its constituents are rearranged with examples from the force-directed ($fd$), clustered-pile ($cp$) and clustered-radial ($cr$) arrangers. Control features from top left to bottom right in $x - y$ are as follows: *lightness-hue cp*, *spatial sd-green red fd*, *spatial sd-yellow blue fd*, *size-hue cr*, *compactness-green red cp*, *compactness-angle cp*, *green red-yellow blue fd*, *size-spatial sd fd*, *size-lightness cr*, *color sd-hue fd*. Input image: Personal photograph by the author.

Overall we make the following contributions:

**Contribution 1** – A novel parameterized algorithmic model motivated by artistic expression based on the mean-shift segmentation technique.

**Contribution 2** – A flexible pipeline allowing varying designs, controlled by 10 visual features for sorting in the spatial and color value domains, two arrangement methods and a radial or pile-based layout.

**Contribution 3** – A systematic design space exploration of the parameters of the model and a user survey for fostering further understanding of the design space, resulting in clear preferences for certain aesthetics.
6.1 Related Work

We now briefly discuss related work for the segmentation step, the included visual features and the arrangement techniques of our pipeline.\(^2\)

**Segmentation:** The identification of different image parts based on pixel data is a crucial preprocessing step for a variety of applications and it is an on-going research topic in computer vision. Raut et al. [2009] thoroughly review the different approaches for segmentation algorithms. Our problem requires the segmentation of color images, summarized by Cheng et al. [2001], as well as an unsupervised segmentation technique for which an overview is given by Zhang et al. [2008].

**Arrangements:** The organization of segments relates to work from an artistic point of view and methodically.

\(^2\) Sections 6.1–6.3 are based on and extend: Klingel, S. Automatisierte Zerlegung von Bildern und die geordnete Darstellung ihrer Konstituenten, 2015, Diplomarbeit. University of Stuttgart – supervised by the author of this thesis.
Artistically, we are inspired by the art of Wehrli [2003], who manually creates image decompositions and rearrangements. We aim for a similar aesthetic but also give enough freedom for an artist to explore and express individual design goals. Wehrli also often focuses on creating an appeal through semantic understanding, a concept that was not the focus of our work but appears by chance in some of our results (please refer to the discussion of our results, Section 6.5).

Technically, our work relates to the pipeline of Ufer et al. [2012], who present a fully automatic recreation of Wehrli’s art pieces that is as congruent as possible. The authors apply an unsupervised multi-region segmentation based on convex relaxation techniques and global k-means color model estimation. The result is a single, optimal and reproducible algorithmic rearrangement solution for each input image. Artistically, our work greatly differs from the approach of Ufer et al. [2012] because we offer a parameterized pipeline, aiming for artist explorations and individual results. In addition to offering a larger and controllable design space, we also add a user study pertaining to the aesthetic appeal of the different possible reconfiguration designs.

Also aiming for a fragmented aesthetic appeal by transforming original image data “piecewise,” Collomosse and Hall [2003] render from salient image features a cubist version of an input. Similarly, Lai and Rosin [2013] loosely aim for a cubist appearance by segmenting an input and by simplifying the segments into constant colors, while maintaining global structures. Adding a level of abstraction, Song et al. [2013] match base shapes to regions in image segmentation hierarchies. Because we arrange image pieces solely controlled by segment features, we focus on an even higher level of abstraction, a goal little investigated in related research.

Methodically, collage methods implement piecewise arrangements as well. These techniques usually strive to fulfill certain semantic rules, such as visualizing a topic, event or story [Goferman et al. 2010; Zhang and Huang 2012]. Huang et al. [2011] use thematically related image cut outs to fill any input shape, leading to the artistic appeal of their collages. Reinert et al. [2013] also densely fill a space with any kind of given two-dimensional graphical primitives following artistic goals. In addition to collage stylizations, mosaic layouts are methodically related to our pipeline. Hausner [2001] presents the placement of mosaic tiles based on image input, following the flow of edges and the coloring of the input. Dalal et al. [2006] present a mosaic packing solution that integrates a novel evenness metric. Hurtut et al. [2009] reproduce example element placements of stroke-based primitives. These mosaic techniques employ predefined visual primitives though, and the design space of arrangement options is limited. For our technique diverse primitive types are computed by image segmentation and we offer a range of possible arrangements.

### 6.2 Pipeline

In the following we are going to describe each component of our pipeline (Figure 6.3).

As color model we use the DIN99 color space DIN [2001]. It constitutes a non-linear color model, aiming for perceptual uniformity corresponding to human vision. It is
Figure 6.3: The decomposition and rearrangement pipeline: The artist chooses an image segmentation technique with its parameter settings, the arrangement method, a feature for the x- and y-axis and, depending on the arranger, an optional layout type.

similar to the $L* a* b*$ space and models color with a lightness, a red-green- and a yellow-blue-contrast dimension.

6.2.1 Segmentation

The quality of the segmentation of an input image highly influences the plausibility and aesthetic appeal of the reconfiguration results. For implementing our pipeline any segmentation algorithm could be applied. We implemented two techniques, the Mean-Shift segmentation method [Cheng 1995], a feature-space clustering technique, and the Watershed Transformation segmentation [Beucher and Lantuéjoul 1979], an edge detection-based method, and investigated the trade-off between over- and under-segmentation and performance in regard to our problem statement. Ultimately we decided on the more precise Mean-Shift method, which leads to longer computation times but its performance has overall no impact in comparison to the time the arrangers take.

The mean-shift algorithm locates the maxima of a density function within a feature space. For image segmentation, the feature space is defined as five-dimensional, with three color channels and the $x,y$-coordinates of a pixel, as presented by Comaniciu and Meer [2002], whose implementation we follow. After segmentation we represent the position of a segment by its center of gravity, which is the mean of all pixel positions within a segment.

In order to determine the background color, we compute two measures for each segment, the area in pixels and the number of neighboring segments. If one segment scores highest in both features, we select its mean color as background color. Otherwise, we divide the scores by the respective standard deviations and choose the segment with the highest overall quotient.

6.2.2 Reconfiguration of Image Constituents

The arrangement of image constituents should appear as reasonable as possible for a human observer. In this context, reasonable can be interpreted with a variety of specific semantic meanings. Ultimately we would like to control a design by how aesthetic it is, how tidy it appears and how informative in respect to understanding the structure and composition of the original image. Towards these goals we investigate
as control parameters ten visual features for which humans are sensitive for, two arrangement algorithms and two optional layout types.

Control Features

Visual features refer to characteristics in the color value and spatial domain of the segments.

In the spatial domain a segment can be characterized by its size, as number of pixels, and its spatial standard deviation. As the spatial standard deviation is dependent on the segment size, we also present the feature compactness as standard deviation of a segment in relation to the standard deviation of a circle with the same area, therefore representing the spatial standard deviation independently from the segment size. The orientation is given by the angle of the principal axis of the segment, determined by a PCA, to the x-axis of the canvas.

In the color value domain we employ the red-green contrast $a_{99}$, the yellow-blue contrast $b_{99}$, the lightness $L_{99}$, chroma as the saturation $C_{99}$ and hue as the angle $h_{99}$ of the DIN99 model and the color standard deviation.

In order to make the features comparable, we normalize them to the unit interval $[0,1]$.

Arranger

Arrangements sort and group segments by positioning them on the canvas according to the control features. By defining an order and a normed distance for the segments in the feature space, we are able to project their visual appearance onto the two dimensional spatial domain and to archive a visually plausible arrangement.

Specifically, a user selects two visual features as classification criteria, which are mapped onto the $x$- and $y$-axis of a canvas. Within that space, all segments are

Figure 6.4: Demonstration of the different arrangers and layouts: (a) and (b) employ a force-directed arrangement with the visual features size-spatial sd. (a) includes the additional rotation correction force for alignment with the x-axis. The clustered arranger is used with a radial layout in (c) and with the pile layout in (d), both controlled by the green red-yellow blue features. Input image: [Wright 2012].
positioned in order of the features respective specification. For the final layout all segments must be ordered and may not overlap. For making the search for a complying layout more manageable, we apply a two step process. In the first step, all segments are placed in order as initial layout, not factoring in possible overlaps. The second step then iteratively refines the layout until no segments intersect.

A refinement iteration needs to detect collisions and to push affected segments apart. For our pipeline we implement a simple constant correction force (Figure 6.5), which is applied in the direction of the connecting vector between the intersecting segment centers. To even further reduce the empty space between segments and to increase the appearance of tidiness, an additional rotation of the segments is possible so as to enforce parallel alignment. This is achieved through a constant force the direction of which depends on the angle between the principal axis of the segments and the $x$-axis of the layout.

**Force-directed arranger:** This is the combination of the above steps to a final layout (Figure 6.4 (a), (b)).

**Clustered arranger:** An additional partitioning step computes the clustering of the segments within the two-dimensional feature space of the initial layout (Figure 6.4 (c), (d)). For finding the clusters in the feature space we apply again the mean-shift method. To implement the different layouts for the clusters and to ensure their compactness, additional compression forces are integrated. These forces do not take possible intersections of the segments into consideration and decrease linearly over ten iterations. Similar to the force-directed arranger, each cluster is simultaneously refined to resolve overlaps between segments with a correction force.

In the **radial layout** the compression force pushes the segments in the direction of their cluster center. The rotations of the segments are fixed to align to their cluster radii. The same strategy positions whole clusters in a radial layout.

The **pile layout** arranges all segments horizontally in the initial layout and the compression force pushes the segments to their cluster center in parallel to the $x$-axis.
In order to position the clusters, the clusters are sorted decreasingly by their area. The pile with the largest area is placed at the bottom left corner of the canvas, with each following cluster placed with an offset in $x$ to the right.

Summary

The reconfiguration model deconstructs a given image into subparts and rearranges the parts based on visual features.

- Mean-shift segmentation deconstructs the image.
- An order and a normed distance for 10 visual features in the spatial and in the color value domains enable sorting and grouping.
- The arranger maps the sorted segments onto the x- and y-axis of a canvas either with a force-directed or a clustered arrangement.

6.3 Parameter Space Exploration

As Section 6.2 illustrates, our pipeline offers a variety of control parameters that influence the visual appearance of the result. In order to support an understanding of the design space, we present several exemplary deconstructions and reconfiguration results in this section.

Segmentation: Settings are responsible for how recognizable the image parts remain. If we choose parameter that lead to a more chunky segmentation, image parts remain recognizable but also potentially heterogeneous. In turn, this will lead to a less tidied up impression of the reconfiguration. Also, a more rough segmentation might lead to false semantic assumptions about image parts. As shown in the third row of Figure 6.9, one of the segments appears to be a mouse when in fact it is just part of the bowls shadow. A finer segmentation, on the other hand, leading to more homogeneous segments overall, might produce eye-catching outliers but otherwise might appear too monotonous. Performance is also steered by the number of segments, increasing the time of the arrangement process. The refinement iterations of the segmentation and the number of segments are not know in advance. As practical performance trails, we compare in Figure 6.6 different segment counts and the resulting timings. We believe that performance is still improvable with some code adjustments.

Arrangement: The spatial organization of the segments enables fundamentally different reconfiguration designs. The force-directed arranger does not regroup the feature space and creates more global and less ‘tidy’ but abstract and possibly more artistic appearing designs. The clustered arranger makes it easier to comprehend the image data through its hierarchical approach, making boundaries in the feature
space clearly visible as clusters, as for example shown in Figure 6.4 (c) and (d). These boundaries are not necessarily directly sensible for a human observer or suited for deriving insights about the data of the original image. At this point we did not optimize our technique for its capabilities to visualize information.

**Layout:** Types constitute the overall large and space-filling structures within the result image (Figure 6.4). As an observer processes these structures as one of the first characteristics of an image, the layouts have a major visual impact. Accordingly, the results of the survey are strongly dependent on the layout type (Figure 6.8).

**Features:** The influence of the feature parameter are most difficult to predict. Even though all visual characteristics are easy to understand, the importance of each feature is hard to judge for an observer on basis of the original image, as the differences in Figure 6.7 show. Also, even if it is obvious which features are dominant in an image, it is not possible to anticipate how the structuring of an arranger will combine the two selected features on the canvas, especially in regard to the identified clusters of the clustered arranger. For navigating the space of feature combinations the user can explore possible reconfigurations with the help of the systems interface. Alternatively, we also offer a batch process, which generates for a segmentation setup all possible arrangers, layouts and feature combinations.
Figure 6.7: All feature combinations within a force-directed arrangement. The arrangements in the bottom row, left to right, are controlled in x-y by *size-yellow blue, size-angle, size-spatial sd* as visual features. Figure adapted from Klingel [2015].
Figure 6.8: Boxplot for the average rankings for the different clustered layouts from the user survey. Participants selected from all options their six favorite layouts and ranked them from 0 to 1 (best) and their six least favorites with a ranking from 0 to -1 (worst). The blue line indicates the median and the red dot the mean of which the exact value is listed in the top row for each box.

Summary

**Segmentation parameters** control the recognizability of the original and the homogeneity of the created pattern. A more homogeneous pattern increases the risk of outliers.

**Arrangement and layout parameters** create fundamentally different designs, with the force-directed arranger being less ordered, while the clustered arranger makes boundaries in the feature space visible.

**Feature parameters** have a explorative nature as their influence is difficult to predict.

### 6.4 User Study

We have conducted a user survey to understand the artistic design space of our results better. We asked participants to identify their six most and six least favorite layouts and ranked them from 0 to 1 (best) and their six least favorites with a ranking from 0 to -1 (worst). The blue line indicates the median and the red dot the mean of which the exact value is listed in the top row for each box.
reconfiguration images from all unique combinations of the six features size, compactness, lightness, chroma, green red, yellow blue, with each feature combination arranged with the clustered arranger in both, radial and pile layouts. We show ten different original images and their reconfigurations to each participant. In total, we collected 28 completed surveys.

For a detailed description of the experiment design, the data and statistical analysis, please refer to the supplementals of the publication [Gieseke et al. 2015].

6.4.1 Results and Analysis

The survey confirmed our hypothesis that the presented layouts are differently ranked by the participants, meaning that some layouts, on an average, produce more visually pleasing reconfigurations than others.

The collected data is overall normal distributed, checked with the Shapiro Wilk test. A repeated-measures ANOVA shows a significant effect of the layout type, with $F(29, 810) = 6.26, p < 0.001, \eta^2 = 0.18$ (Figure 6.8). Post-hoc pairwise comparisons also document significant differences ($p < 0.055$, computed with the Tukey’s Honest Significance Test) between the high ranged and low ranked layouts.

The most prominent observation from the survey data is that the radial layout is strongly favored, as from the 13 layouts with a positive ranking (from 30 in total), 10 employ the radial layout. Reconfigurations with the features lightness, chroma, green red, yellow blue, therefore overall referring to color, also lead to a greater visual appeal, as all layouts with a positive ranking are controlled by at least one of the color features. The color feature red green seems to outperform the others slightly, nevertheless we believe that which color feature to chose is depended on the original image. Compactness performs significantly worse than all other features, indicating that human observers pay little attention to the quality of the shape of a segment. The poor performance of this feature must also be credited to the output size of the reconfiguration images, as small sizes make the shape of the segments hard to recognize and the arrangement appears therefore random.

Summary

We asked participants which reconfigurations they liked in terms of visual appeal.

- Radial clustering was significantly preferred over the pile layout.
- Overall, all color features were liked the best.
- Compactness performed significantly worse than all other features.
6.5 Discussion and Outlook

Our technique produces parameterized, algorithmic art and offers a change of perspective of the original image data. With a large number of parameters controlling the rearrangements, an artist can navigate a variety of possible results. Due to implementation specifics, the results are not necessarily predictable in every detail, and artists are invited to experiment and to explore. Such a more intuitive than goal-oriented exploration of a design space is typical for an artistic approach. It is also up to the artist to set the semantic goal for the reconfiguration. Moreover, optimal parameter combinations are dependent on the design of the input image.

We furthermore investigated a first step toward an automatic configuration by gaining insights about human preferences and their connection to the parameter setup with a survey. Nevertheless, we did not incorporate the results of the study into the pipeline. It may be worthwhile for future work to investigate an automatically predicted configuration based on the semantic goal and the type of input image. The pipeline has no means for a semantic understanding of the input image itself or to match reconfigurations accordingly. Nonetheless, some lucky results mimic the content of the original image, creating especially pleasing results (see, for example, the middle image that looks like a fish as a reconfiguration of a fish tank in Figure 6.1).

Figure 6.9: Exemplary reconfiguration images computed with our pipeline with the force-directed \((fd)\), clustered-pile \((cp)\) and clustered-radial \((cr)\) arrangers. Control feature from top left to bottom right in \(x\)-\(y\): \textit{compactness-red green fd, compactness-hue cr, lightness-red green cp, lightness-red green cp, lightness-red green cr, lightness-color sd fd, red green-hue cr, red green-yellow blue cp, red green-chroma fd, lightness-yellow blue cr, red green-colorSD fd}. Image sources: please see Section 7.4 Image References.
Some of the factors that influence the visual appearance of the reconfiguration should have individual control parameters. Currently, the background color of the reconfiguration is the mean color value of the largest segment, which in turn is not included in the arrangement. The background color has a great visual impact on the aesthetic appeal of the result because one of the first perceptual grouping steps of human vision is the figure-ground organization. Therefore, when trying to align the appearance of the reconfiguration to the original image, it is important to create a similar foreground-background impression, which cannot solely be based on the largest segment. For the variation on *Starry Night* in Figure 6.10, the background is clearly a sky, and the background color of the reconfiguration should be a shade of blue rather than its comparatively unappealing green color. The similarity of the results to the original can improve by offering the option to adjust the background color or to have the user identify a segment as background. It would also be worthwhile to investigate in-painting methods with the goal of applying a segment directly as background. Areas of that segment where other segments are cut out could be filled in a visually matching fashion.

The aspect-ratio of the images is also of importance for the visual appeal of a result and for its similarity to the original. At this point, a final ratio is chosen by fitting the frame to the spacing of the segments, and the new ratio does not necessarily match the original (Figures 6.9 and 6.10). It might be beneficial to determine the desired aspect ratio in advance and to give the user control over it. In order to further improve the global layout of the design, the “empty” spaces between segment clusters should also be controllable, leading to more balanced results.

Additional design options for the arrangements of the segments are almost limitless. For example, a user could predefine shapes in which the segments should be arranged. Also, the global structures of the original image could control the arranger.

The applicability of the pipeline in terms of gaining further understanding about the original input remains questionable in the current state of the reconfiguration. In order to control the output to a degree that would enable actual information visualization, an artist would need more direct control over the design. In this regard, the definite optimization results of Ufer et al. [2012] could be combined with a visual analytics system. Moreover, in order to assess the information content of a reconfiguration, a survey should pose actual tasks for the participants to solve with the help of the reconfigurations.

### 6.6 Conclusion

In this chapter, we present a parameterized model to deconstruct an input image and to put it back together in an artistic and visually pleasing fashion. We explore the design space of our pipeline both with examples and a survey. We show that a wide range of artistic goals, for which we selected some more examples in Figure 6.9, are expressible with our pipeline. Based on the presented model, now an investigation of possible artistic control mechanisms is called for. With our algorithmic art, which is grounded in an academic context, we hope to inspire the artistic community to document and share approaches, algorithms and result evaluations more often.
Figure 6.10: Some aspects of future work are demonstrated with these reconfigurations. Due to the radial layout within the original, the pipeline could pre-select the radial clustered design as most suitable for the left original of van Gogh’s *Starry Night* and the pile layout for the supermarket image on the right with its strong composition lines from the shelves. However, the automatically computed background color is problematic, which should be a shade of blue for the *Starry Night* variation so that the background could be easily identified as sky. Moreover, the aspect ratio of three of the reconfigurations is in no relation to the originals, leading to a different overall impression. Image sources top to bottom: [van Gogh 1889; Stenudd 2012].
Conclusion and Outlook

This thesis contributes artist-centered creation processes for the procedural generation of two-dimensional patterns. The work considers the whole spectrum of controllability, including goal-oriented control, creative control and artistic exploration. All presented techniques are accompanied by user studies, referring to visual features, usability and aesthetic appeal. Hence, the surveys include the perspective of an artist or observer into the discussion of the algorithms.

After a review of the specific contributions, the following shows how the work contributes to other contexts in a generalized manner. Thereafter, insights are given on promising future research directions, and final remarks conclude the thesis.

7.1 Summary of Contributions

For a better understanding of creative creation and its translation to digital tools and their controllability Chapter 2 Analysis Framework for Creative Control introduces a novel classification framework for creative control. The chapter identifies navigation, transparency, variation and stimulation as means for creativity. A taxonomy relates common control methodologies to the how, what, where and when of a creation process by analyzing exemplars, parameterization, handling, filling, guiding and placing as interactions.

For an application of the analysis framework to the state of the art in the context of creative procedural pattern generation, this thesis chooses the specific design space of ornamentation. Ornamentation uniquely combines the need for structured automation of design principles with individual design choices, depending on the artist and the space to fill. In working toward the goal of a creative creation pipeline for ornamentation, Chapter 3 State of the Art of Creative Control for Procedural Ornamentation analyzes the state of the art and classifies it in regard to its controllability, creative means and ornamental results. Due to the complexity of ornamentation and its creation requirements, the chapter includes the investigation
of relevant work from other representations and output domains. The chapter also identifies what is still missing with regard to achieving creative control of ornamentation and advocates a unifying approach to further combine procedural and data-driven techniques.

As a representative goal-oriented control example-based parameter matching is investigated. As a prerequisite, relevant visual features are abstracted in Chapter 4 Goal-Oriented Control for Interactive Parameter Retrieval to quantifiable texture descriptors with Fourier, Laplace pyramid and Gabor filter bank descriptors. The parameter spaces are searched with Nelder-Mead and Levenberg-Marquardt optimization techniques. Different descriptor-optimizer combinations are compared and numerically analyzed for realistic two-tone patterns. Furthermore, a user study evaluates how the different descriptors correspond to human perception in both natural and synthesized image spaces.

The achieved comparison results enable well informed choices and a reasonable setup for an example-based parameter matching pipeline. The second part of Chapter 4 derives a novel retrieval technique for this. With a perceptually-driven and highly compact Gabor filter descriptor vector and the separate handling of the color and structure of a texture, a database of descriptor/structure parameter vector pairs can be pre-computed. The database is then searched with interactive performance to identify the parameter set that renders the procedural texture most similar to the given exemplar.

Chapter 5 Creative Control for Organized Order in Ornamentation presents a solution to combining automation with manual creation for the abstracted pattern type of ornamentation. A uniform approach supports control on various abstraction levels – from a global order down to individual element placement. Design constraints such as symmetry are defined by the artist but then automatically optimized for by the system. The optimization considers further artist input such as manually placed frames, strokes or individual elements and complements all to a cohesive whole. Overall, Chapter 5 presents a framework that contributes uniquely and more completely than the state of the art to the means for creative creation for ornamentation identified in Chapter 2. A designers survey confirmed the appeal, efficiency and further potential of the methods.

For an artistic exploration as open and the least targeted control mechanism within a creation process, Chapter 6 Explorative Control for Artistic Expression contributes with a parameter-driven experimental model. The model automatically deconstructs an image into visually coherent constituents and rearranges those pieces in a surprising and aesthetically pleasing fashion. Because it is parameterized, the reconfiguration can be controlled by the artist to a degree while still leaving considerable opportunity for surprising results and exploration. The result images were analyzed for their visual appeal with a user study. The study identified certain aesthetics as being clearly preferred by the audience, giving valuable insights on how to proceed with this work.
7.2 Generalizability

The framework presented in Chapter 2 for the analysis of creative control is highly generalized and applicable for every digital creation tool and interaction scenario. Furthermore, it can easily be combined with a user study, either as formal analysis step on its own or as a supporting structure and taxonomy for user tasks and questions.

Even though the specific creation techniques in Chapters 4 to 6 individually create meaningful results, their methodologies and output designs would integrate well into a unified pipeline for general procedural pattern generation.

The different techniques complement each other. For example, the more realistic textures addressed in Chapter 4 are often a subpart of an ornamental design. Similarly, the application scenario of texture design for a 3D visual effects scene in Chapter 4 requires global design principles, layouts and visual hierarchies in the same way as they are presented in Chapter 5. The experimental image reconfiguration model in Chapter 6 could be adapted to rearrange some parts a rendering of an ornamental model as they are used in Chapter 5. This could create an interesting variation to the actual model. The presented radial layout of Chapter 6 is a suitable design for ornamentation.

None of the techniques are dependent on the specific pattern designs presented in this work and the procedural models can be exchanged. The distance metric in Chapter 4 computes a similarity between what ever designs are given, and the weighting of the different structure scales can even be configured. The only requirement for the procedural models in Chapter 5 is the iterative generation logic. The approaches of Chapters 5 and 6 are also straight-forward to transfer to three dimensions and can be applied to geometries or even particle systems.

The principle of the image reconfiguration model can furthermore be generalized to an information visualization technique and could give insights about a painting that are not obvious on first sight, such as element counts or the overall color scheme. In order to do so, the exploratory nature of the model would need to be restricted in favor of full controllability.

The results of the user survey about the alignment of human perception of procedural texture features to the similarity measures based on a Fourier, Laplace and Gabor texture descriptor, as well as the results of the survey in regard to the visual appeal of different reconfiguration results are generalizable as the applied inferential statistics prove. The qualitative results of the survey with designers in regard to the usability of our creative creation pipeline emphasizes the general need for additional quantitative evaluation. With the comparatively small number of participants some contradicting results appear (“I prefer less vs. more order in the results”) and these can only be decided by a larger number of survey results and statistical evaluation. However, some completely unified results to some aspects of the qualitative survey also allow for transferrable insights. All negative feedback is in regard to missing convenience feature implementations. For example an undo functionality seems to be an essential requirement for practitioners, it is however usually neglected by research projects, as it is also discussed in the 7.3 Outlook of this thesis.
7.3 Outlook

In addition to the already discussed future research specifically presented for each technique, this thesis opens up the following directions overall.

**Procedural Models**  Research into artist-centered control mechanisms for procedural representation requires a variety of suitable models. From the perspective of an artist, control mechanisms can only be as good as their underlying models. For this thesis, the models have been carefully crafted by the author. This is a time-consuming and tedious task. Research to ease the creation of procedural models themselves is called for. Promising existing work on inverse procedural modeling with L-systems [Štáva et al. 2010] is a good place to start. In a similar way to how the pipeline of Chapter 5 allows for interactive adjustments of the output of a procedure, it would be worthwhile to investigate whether the procedural models themselves could adapt to artist input. Furthermore, the possible design spaces of procedural models pose many research challenges, such as a procedural layering of different models, for example.

**Control Mechanisms**  Chapter 5 presents a procedural generation pipeline for ornamentation that is the most complete in comparison to the current state of the art. This approach should be taken further, and all aspects of all chapters should be integrated into one procedural pipeline, as already discussed in Section 7.2. The pipeline may need to be extended with regard to some missing functionality, such as a navigation history. Even though this seems to be primarily a development task, addressing it may likely also offer some unrecognized research questions.

**Creative Creation**  The taxonomy for creative means and control mechanisms presents well defined characteristics that can be identified by a detailed analysis of a technique. In order to offer more directly comparable features of different techniques, the characteristics could be further specified. This could potentially lead to quantifiable features and even an automatic analysis framework.

Currently the stimulation category of the creative means, including the aspect of immersion, is the least specific. In combination with research into psychology and cognitive science, concrete mechanisms should be investigated, leading in turn to a better understanding and novel determining terminology.

In addition to the considerations about stimulation, the already discussed (Section 3.5) aspect of collaboration should become a focus of future research. Many creative processes depend on working together, and collaboration can be a valuable source for one’s own creativity. In this respect, digital tools and cloud-based computing bring unique opportunities that are impossible in the analog world.

**Artistic Expression**  Even though there have been various algorithmic art projects, very few have operated in an academic context. The rise of artistic research as a discipline addresses the gap between art practice and academic evaluation and opens up new forms of interdisciplinary research. It may be worthwhile to investi-
gate in a structured manner how artists create and control algorithms for artistic expression.

At the same time, technology, such as video analysis, photogrammetry or motion capture, could be used to measure and analyze the creation processes of traditional artists working with analog media. These findings could in turn be used to develop more suitable digital tools.

7.4 Closing Remarks

The overall challenge addressed in this thesis is how to support artists in their work with meaningful control mechanisms.

The investigation of controllability is put into the context of procedural generation of two-dimensional pattern designs. Procedural models and the computation of designs offer novel approaches to create content and benefits over traditional manual manufacturing. However, to provide control mechanisms that are intuitive to use and allow for individual designs is an ongoing research challenge.

To tackle this challenge, this thesis provides first a better understanding of the creation process of an artist, means for creativity, and how to relate the identified characteristics to control mechanisms for digital tools.

Specific techniques for three scenarios are presented: goal-oriented control for realistic textures, creative control for ornamentation and experimental control for artistic expression. Each approach is evaluated with a survey: in regard to the visual features of a pattern that are relevant to a human observer, in regard to the usability of the creation pipeline and in regard to the aesthetic appeal of the result.

The presented techniques complement each other methodically and can be seen as building blocks for a cohesive pipeline. This thesis further opened up the direction of bringing different approaches together and to carefully analyze and emphasize an artist-centered perspective. This is the basis for developing innovative tools that further the ability of artists to create and to creatively express themselves.
Image References

Chapter 2

Figure 3.1

Left to right, top to bottom: Manuscripts and Archives Division, The New York Public Library. [1450 - 1475]; Jones [1867]; The Miriam and Ira D. Wallach Division of Art, Prints and Photographs, The New York Public Library. [1882]; Spencer Collection, The New York Public Library. [1910]; The Miriam and Ira D. Wallach Division of Art, Prints and Photographs, The New York Public Library. [1877]; Morris [Around 1880, 1883]; Jones et al. [1868]; Jones et al. [1868]; Murphy [2018]

Figure 3.2

Ornament: Morris and Dearle [1910]

Figure 3.3

Left to right: Free Patterns Area [2018]; Marcel’s Kid Crafts [2018]

Chapter 4

Figure 4.2

Figure 4.5

Target images left to right, top to bottom: Martin; Walsh; Bptakoma; Else; Bleasdale; Mr Thinktank [2016]; Erase99 [a]; Palacy; Boudreaux [2016]; Campbell; van Deelen; Virtually_supine; Villalobos [2016]; Bangsmoen; Hill [a]; Beam; Webtreats; Kimbrough [2016]; Tanakawho; WarOnTomato [g]; García; Mr Thinktank; WarOnTomato [c]; DncnH; Janus; Musta; Palacy [2016]; Caleb; Ro; Stewart

Figure 4.6

All target images: Gieseke and Koch [2013-2015], except 3.: Mr Thinktank [2016], 4.: Boudreaux [2016], 5.: Palacy [2016], 8.: Kimbrough [2016]

Figure 4.8

Target image: 3dtotal [2014]

Figure 4.10

Target images top and bottom: CG Textures [2013], target image center: Gieseke and Koch [2013-2015]

Figure 4.11

Target images first column, top to bottom: CG Textures [2013], target images second column, top to bottom: Gieseke and Koch [2013-2015], except 2. image: CG Textures [2013]

Figure 4.12

All target images: Gieseke and Koch [2013-2015], except target first column, first image: CG Textures [2013], target second column, second image: 3dtotal [2014]

Figure 4.13

All target images: 3dtotal [2014], except target images first row: Gieseke and Koch [2013-2015], target images second row: CG Textures [2013]

Figure 4.14

All target images: CG Textures [2013], except target images third row: Gieseke and Koch [2013-2015]
Figure 4.15
Target images first column, 1., 2., top to bottom: CG Textures [2013], target images second column, 1.–3., top to bottom: CG Textures [2013], target images first column, 3., top to bottom: 3dtotal [2014], target images second column, 4., top to bottom: Turbosquid [2014]

Figure 4.19
Target images first row: CG Textures [2013], target images second row, 2., left to right: Turbosquid [2014]

Figure 4.20
Target image left: 3dtotal [2014], target image right: CG Textures [2013]

Chapter 5
Figure 5.1
Left to right: Colourbox [2011, 2016, 2013a]; Rejke [2011]; Colourbox [2013b]

Chapter 6
Figure 6.1
Input image: Personal photograph by the author.

Figure 6.2, 6.4, 6.6, 6.7
Input images: Kandinsky [1921]; Wright [2012]; van Huysum [1715]; Malewitsch [1912/13]

Figure 6.10
Input image top to bottom: van Gogh [1889]; Stenudd [2012]

Figure 6.9
Input images top to bottom, left to right: Seurat [1888]; Botticelli [1485/86]; van Hulsdonck [1620]; Seurat [1888]; van Gogh [1890]; Freepik [2015]
Image Sources


Botticelli, S. La nascita di venere, 1485/86. In the pulic domain. 147


ElbtheProf. Waiting... Flickr. http://flic.kr/p/4u17GT. Accessed July 20, 2016. CC-BY-2.0 Creative Commons License. 145


Gieseke, L. and Koch, S. Personal photograph, 2013-2015. 85, 146


Jones, O. Examples of Chinese ornament selected from objects in the South Kensington museum and other collections. London : S. & T. Gilbert, 1867. 145

Image Sources


Kandinsky, W. Roter Fleck II, 1921. Blue Rider Collection, Lenbachaus, Munich. 127, 147


Malewitsch, K. Der Holzfäller, 1912/13. In the public domain. 147


Morris, W. Strawberry thief printed textile, 1883. Planet Art CD of royalty-free PD images - William Morris. 145

Morris, W. Floral wallpaper design, Around 1880. Stapleton Collection. 145


Seurat, G. Seascape at port-en-bessin (Cutout), 1888. The W. Averell Harriman Foundation. Open Access. 147


Stenudd, J. Veganz, a supermarket (Cutout). Flickr, 2012. https://www.flickr.com/photos/72616463@N00/8209737977. Accessed March 10, 2015. CC-BY-2.0 Creative Commons License. 139, 147


van Gogh, V. Starry night (Cutout), 1889. In the pulic domain. 139, 147

van Gogh, V. Green wheat fields, 1890. Collection of Mr. and Mrs. Paul Mellon. Open Access. 147

van Hulsdonck, J. Wild strawberries and a carnation in a wan-li bowl, 1620. The Lee and Juliet Folger Fund. Open Access. 147

van Huysum, J. Still life with flowers and fruit, 1715. Patrons’ Permanent Fund and Gift of Philip and Lizanne Cunningham. Open Access. 133, 147


Bibliography


Association for Computational Creativity. http://computationalcreativity.net, 2018, Viewed September 05, 2018. 6


Dresser, C. *Principles of decorative design*. Cassell Petter and Galpin, 1875. 24


Julesz, B. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981. 54


Moughtin, C., Oc, T., and Tiesdell, S. *Urban Design: Ornament and Decoration*. Architectural Press, 1999. 24, 26


Nelles, O. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001. 70, 71


Prusinkiewicz, P. *The algorithmic beauty of plants*. Springer, 1990. 29, 103


