Computational Aesthetics:
On the Complexity of Computer-Generated Paintings

Kang Zhang, Stuart Harrell and Xin Ji

Many argue that with the rapid advances of modern technology, particularly digital display and computer graphics, digital art is more expressive than traditional visual art. This increased expressive power, compounded with highly advanced artificial intelligence, has created tremendous opportunities for the realization of computational aesthetics and even simulated creativity [1]. If the potential of computational aesthetics is achieved, we would see a profound impact on various application domains where computer technology has traditionally played only an assistive role, such as graphic and industrial design. This is not to advocate the possibility of replacing human creativity with computational creativity; rather, the advancement of computational aesthetics would further extend human creativity by providing inspiration to artists and graphic/industrial designers.

I. EMERGING NEW DISCIPLINES IN ARTS AND COMPUTING

When combining visual art with digital technology, two lines of questioning tend to emerge. On the one hand are questions such as: “How can the computer automatically generate various forms of visually aesthetic expressions?” On the other hand, some ask: “How can the theory and techniques of traditional visual art help beautify modern technology outputs and products and enhance their usability?” Addressing such questions, two interdisciplinary areas have emerged in recent years: computational aesthetics and aesthetic computing [2]. Both computational aesthetics and aesthetic computing aim at bridging computer science, philosophy, cognitive science and the fine arts through analytic and synthetic investigation.

Figure 1 illustrates a conceptual map of the relationships between various computer science areas (in rectangular nodes) and arts (in oval nodes), moving from theories to applications. The dotted arrows in Fig. 1 help to conceptualize that computational aesthetics aims at answering the first question above, that is, “How can the computer automatically generate various forms of visually aesthetic expressions?” In other words, computational aesthetics investigates how modern technology helps the arts. This technology in fact serves to create tools that can enhance the expressive power of visual art and heighten human understanding of aesthetic evaluation, perception and meaning. In the reverse direction, illustrated by the dashed arrows in Fig. 1, aesthetic computing addresses the second question—“How can the theory and techniques in the traditional visual arts help beautify modern technology outputs and products and enhance their usability?” This issue includes the aesthetic design of computer algorithms, simulation, visualization, human-machine interfaces and high-tech products, so that users are highly engaged and thus usability is enhanced. An interesting example of aesthetic computing is the application of Kandinsky’s aesthetics to Java programming [3]. In 2006, Malina [4] highlighted the aesthetic computing activities published in Leonardo over the previous 40 years.

We have previously examined aesthetic computing, studying how information visualization using modern computer technology could benefit from the theory and practice of ab-

Fig. 1. A conceptual map of computational aesthetics and aesthetic computing. (© Kang Zhang)
Fractal images, such as Fig. 3, are representatively generated visual outputs. A computer program can then automatically generate various attributes and styles, and drawn and saved in advance.

At Level 2, the user needs only to provide various attributes and styles, and possibly mathematical formulas, as inputs. A computer program can then automatically generate desired visual outputs. Fractal images, such as Fig. 3, are representative of works at this level. Research in fractals and fractal arts was extremely prolific from the late 1980s to the early 2000s [7,8]. One of the best-known researchers on this topic is Taylor [9,10]. Most other works at this level are freestyle fractal images (e.g. Fig. 3) automatically generated using adapted and randomized iterative formulas.

Two general approaches are used to characterize Level-3 computerized abstract paintings: generative and transformational. With the generative approach, artists’ styles are encoded into computational rules and algorithms, so that it is possible to generate paintings of a particular style that mimic an original artist’s paintings. One example at this level can be seen in automatically generated Mondrian-style abstract paintings, as shown in Fig. 4. We can trace these efforts back to the pioneering work of Noll [11].

The Level-3 transformational approach includes two-dimensional nonphotorealistic rendering (NPR) that is used to transform digital images into technical illustrations, cartoons, watercolor paintings and sketches, as well as abstract paintings. This approach applies image-processing techniques to an input image (usually a photo) to mimic brush strokes and texture patterns, such that the image is transformed into an abstract painting [12,13].

The main differences between levels 2 and 3 are that Level 2 is usually generated based on mathematical formulas augmented with certain degrees of randomness, while Level 3 uses knowledge-based machine intelligence and is typically heuristics-based. Level-3 research and practice has been most active in the last 10 years. The section below titled “An Attempt to Generate Kandinsky-Style Paintings” will further demonstrate a Level-3 effort. This is our recent attempt to generate Kandinsky-style paintings.

Level 4 is the ultimate aim of computational intelligence that is creative enough to generate highly aesthetic visual forms, such as abstract paintings or graphic designs. An abbreviated list of computational intelligence capabilities includes:

- automatic detection of specific styles from existing painting images
- objective aesthetic evaluation and measurements [14,15]
- adaptation to the social, emotional and cultural backgrounds of the audience
- automatic conceptualization of complex information into abstracted visual formats.

In Section IV below, “Toward More Systematic and Creative Approaches,” we will explore a possible systematic approach as a Level-4 attempt and discuss future research directions in reaching the ultimate aim of maximal computational intelligence in enabling creativity. The four levels of complexity and their characteristics are summarized in Table 1. Human involvement in the Level 4 processes enables the combined and maximized creativity of human artists and computational generative power.

II. COMPUTERIZED ABSTRACT PAINTING

When considering computer-generated abstract paintings, we may consider four levels of sophistication based on the computational power utilized. Here we measure the sophistication of computer-generated abstract paintings by their computational complexity—that is, to what extent machine intelligence is utilized in generating the paintings—rather than by their visual complexity, as in Taylor’s analysis [6]. At Level 1, one could use an existing painting software to draw paintings manually, as illustrated by the example in Fig. 2. At this level, the user may also select various visual components from a database of either manually generated or automatically generated components. With these works one can change visual attributes as needed. The computer provides digital brushes, a variable-sized canvas, a palette of colors and possibly a repository of commonly used visual elements that were manually drawn and saved in advance.

At Level 2, the user needs only to provide various attributes and styles, and possibly mathematical formulas, as inputs. A computer program can then automatically generate desired visual outputs. Fractal images, such as Fig. 3, are representative of works at this level. Research
Several secondary elements that are derived from the primitive elements are also given prominence, such as the checker design and the intermingling of half-circles with straight lines. These are further combinations of the three primitive elements and have interesting properties. The checker design is a derivative of the equal squares of a checkerboard; the lines separating the equal squares are skewed and disfigured. The combination of the skewed and missing lines promotes discord, while the semblance to the unity of the checkerboard contributes to the aesthetic appeal. The generated paintings embody symbolists’ aesthetic-eclectic, ideational, “dematerializing” properties, which are characteristic of abstractionists.

The algorithm for composing elements on the screen follows our original blueprint of constructing an arrangement of primitive shapes and then disrupting it with random perturbations, as depicted in Fig. 5. We treat the line primitive as if it were an adhesive, that is, it associates the various elements with one another. We begin by spawning a single line. Key points are picked along this line, including both ends and random points on the line. Furthermore, these key points, which act as connection junctures for primitives to attach to, are defined for every element. Next, for a specified number of iterations, a random element is placed at the randomly chosen key point of the previous element. The more iterations that are performed in this stage, the more harmoniously the elements appear together. The entire process, starting with the randomly drawn line, is then repeated for a pre-specified number of iterations. The more iterations that are performed in this stage, the more randomization or discord is induced. This enabled an easy method by which to manipulate the ratio of the combination of aesthetic properties with dissonance to find the optimum balance.

The composition approach outlined here reveals a number of similar properties between our computerized paintings and those of Kandinsky. They both “grow outwards from many points to envelop the whole (in a manner similar to the way new skin grows over a wound)” [23] rather than being developed from a single center as paintings of other genres are usually composed. The same forked approach was taken for color. The colors for each component of the painting are chosen randomly from a subset of either light or dark colors depending on what stage of the algorithm is being performed. If the stage of initial aesthetic construction is being performed, then the colors will be chosen to be complementary; otherwise, if we are in the final dissonance stage, random colors are used. Our simple contrast model could be improved by introducing color schemes, but this is not in play with the concept of a simple approach. Gradients were also introduced; for example, circles radiate from the light subset of color to the dark subset. This creates an interesting effect; it serves to distinguish the greater shape—the circle—from the rest of the greater shapes without any change in either dimension or composition. The balance and simplicity present throughout the specification and application of rules are what make this approach successful. Two more example drawings automatically generated are shown in Color Plate C No. 2 (a,b).

In summary, mimicking Kandinsky’s style using a computational approach, as shown above, is extremely challenging. A deep understanding of the painter’s psychological and emotional expressions and much more sophisticated analysis are needed to properly encode his true

<table>
<thead>
<tr>
<th>Level of Complexity</th>
<th>Human Participation</th>
<th>Means of Computer Support</th>
<th>Applicability</th>
<th>Example in This Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Full</td>
<td>Digital canvas and brushes, color palette</td>
<td>Painting and graphic design tools</td>
<td>Paint program (Fig. 2)</td>
</tr>
<tr>
<td>2</td>
<td>None</td>
<td>Mathematic forms, randomized</td>
<td>Fractal art, limited styles of paintings</td>
<td>Fractal art (Fig. 3)</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
<td>Knowledge-based heuristic rules or patterns encoding given styles or domain knowledge</td>
<td>Domain-specific, or style-specific paintings and design</td>
<td>Program-generated Piet Mondrian (Fig. 4) and Section III</td>
</tr>
<tr>
<td>4</td>
<td>Minimal</td>
<td>Heuristic encoding aesthetic rules and multidisciplinary domain knowledge, customizable</td>
<td>Automated abstract painting and graphic design</td>
<td>Future direction</td>
</tr>
</tbody>
</table>
style into the generative machinery of rules and algorithms. Kandinsky once said “that one can feel the multi-sensory consonances and dissonances in simultaneously performed color movements, musical movements and dance movements” [24]. The aesthetic properties displayed in his paintings were perhaps profoundly influenced by his synesthesia [25]. Advanced artificial intelligence and social computing techniques at Level-4 complexity are required in order to generate any digital arts that may parallel Kandinsky’s style.

IV. Toward More Systematic and Creative Approaches

An abstract painting or graphic design is often considered an aesthetic composition of various shapes, each with specific spatial and color assignment. The previous sections discussed the use of computational rules for encoding aesthetic principles and specific styles. Such rules, when expressed in graphs, shapes and spatial positions, are best represented as grammars that are programmable and executable by computers. A rule consists of two parts separated by an arrow pointing from left to right. The part to the left of the arrow is termed the Left-Hand Side (LHS). The part to the right of the arrow is termed the Right-Hand Side (RHS). The process of applying the rules to an existing graph to verify that the graph obeys the defined grammar is called parsing. The reverse process of applying the rules to generate all legal graphs is called generation. Each step during parsing or generation is called a transformation. The generation process is in the same spirit as Leyton’s definition of aesthetics, based on the principles of maximization of transfer and maximization of reusability [26].

Graph grammars with their well-established theoretical background have long been used as a natural and powerful syntax-definition formalism [27] for graphical programming languages [28], which model structures and concepts in a 2-dimensional fashion. The parsing algorithm, based on a graph grammar, may be used to check the syntactical correctness and to interpret the language’s semantics.

Different from all other existing graph grammar formalisms, the spatial graph grammar (SGG) [29] introduces spatial notions into the abstract syntax and is context-sensitive. In the SGG, nodes and edges together with spatial relations construct the precondition (as LHS) of a rule application. Figure 6 is a screenshot of the SGG tool called VEGGIE [30] during the process of constructing a grammar rule. Using spatial information to directly model relationships in the abstract syntax is consistent with the concrete representation, avoiding converting spatial information to edges. The distinct spatial capability in the spatial graph grammar makes the SGG an ideal specifying formalism to interpret abstract painting styles and graphic designs. SGG has been applied to mobile interface adaptation [31], UML diagram interpretation [32], reverse engineering [33] and Web interface interpretation [34].

Shape grammars [35], on the other hand, are a class of rule-based systems that generate geometric shapes. A shape grammar consists of shape rules and a generation engine that selects and processes rules. A shape rule defines how an existing part of a shape can be transformed. It depicts a condition in terms of a shape and a marker. The RHS depicts how the LHS shape should be transformed and where the marker is positioned. The marker helps to locate and orient the new shape [36].

Shape grammars can be used to form a basis for visual computation. The primitives in shape grammars are shapes, rather than symbols as in the SGG. The relationships and operations are all spatial (e.g. similarity, rotation) rather than symbolic, and thus complement spatial graph grammars. Shape grammars have been studied and applied in designs of mechanical parts [37], row-houses [38], floor layouts [39], decorative patterns [40] and interior layouts of buildings [41], as well as generating painting styles [42]. A shape grammar interpreter (SGI) developed by Trescak is available online [43,44]. It supports real-time sub-shape detection and labeled rules, as illustrated by an SGI screenshot in Fig. 7.

An interesting application of the spatial graph grammar formalism combined...
with shape grammar formalism is to generate shape-based abstract painting styles like that of Miró. Kirsch and Kirsch first proposed to model Miró’s style by storing typical Miró shapes in a database and then manually analyzing the target composition using the stored shapes [45]. They did not encode the analysis into a shape grammar but offered some preliminary discussion. In our proposed approach, the shape grammar is responsible for generating singleton and composite shapes derived from a user-provided set of basic shapes. The spatial graph grammar is used to position these shapes in the provided 2-dimensional space. A combined parser for both grammars is used to position these shapes in the provided 2-dimensional space. A combined parser for both grammars is used to generate an almost infinite number of visual images with different shape compositions and positioning. Creativity is thus demonstrated by the generation of many potential solutions toward specific designs [46]. From this point, two approaches to complementing computational intelligence with human creativity may be used to refine the solutions.

Using the first approach, the human designer would manually refine (possibly adding/deleting/modifying) the grammar rules to introduce more constraints in order to filter out less-desirable solutions. Then the grammatical system would be executed again with the refined grammars to generate better solutions. This iterative process could continue as many times as necessary until one or more satisfactory solutions are generated.

The second approach uses more computational intelligence. The human designer selects a subset of the solutions, based on which the grammatical system automatically refines the grammars itself by parsing the selected and unselected solutions and revising the grammar rules. Thus, new solutions are generated as many times as necessary until one or more satisfactory solutions are generated.

Level-4 computational aesthetics poses challenging questions to artificial intelligence researchers, such as how to complement a human’s creativity using computational creativity and whether machine intelligence could eventually become more creative than human beings. Aesthetic principles such as those of Gestalt theory [47,48] could be programmed into production rules.

In the domain of chess competition, a computerized chess player, IBM’s Deep Blue, has already been able to beat all the world chess champions, including the Russian Grandmaster Gary Kasparov. Can the computer be an artist? [49] We believe that, given a constrained domain, such as graphic design for a specific application, computer-generated aesthetics in a 2-dimensional or 3-dimensional art form will soon be useful.

Acknowledgments
We would like to thank the anonymous Leonardo reviewers for their insightful and invaluable comments, which have helped us to improve the paper’s presentation significantly. Stuart Harrell was partially supported by a UT-Dallas Undergraduate Research Grant.

References and Notes
Unedited references as provided by the authors.

24. Ione and Tyler [20].
33. Zhao and Zhu [13].
36. Sting [35].
45. Kirsch and Kirsch [42].
49. J. Lee, Goodman’s Aesthetics and the Languages of Computing, in Fishwick [2], 29–42.
50. Trescak et al. [44].

Manuscript received 13 August 2010.

Kang Zhang is a Professor of Computer Science at the University of Texas at Dallas, U.S.A. His research interests include information visualization, visual languages and aesthetic computing.

Stuart Harrell is an undergraduate student in the Mathematics program at the University of Texas at Dallas, U.S.A.

Xin Ji is a lecturer in Aesthetics at Southeast University, China. Her research interests include aestheticization of everyday life, art history and computational aesthetics.
ANNOUNCEMENT

Leonardo Book Series

Editor in Chief: Sean Cubitt
Editorial Advisory Board: Annick Bureaud, Laura U. Marks, Anna Munster, Michael Punt, Sundar Sarukkai, Eugene Thacker
Editorial Consultant: Joel Slayton

The arts, sciences and technology are experiencing a period of profound change. Explosive challenges to the institutions and practices of engineering, art-making and scientific research raise urgent questions of ethics, craft and care for the planet and its inhabitants. Unforeseen forms of beauty and understanding are possible, but so too are unexpected risks and threats. A newly global connectivity creates new arenas for interaction between science, art and technology, but also creates the preconditions for global crises. The Leonardo Book Series, published by The MIT Press, aims to consider these opportunities, changes and challenges in books that are both timely and of enduring value.

Leonardo Books provide a public forum for research and debate; they contribute to the archive of art-science-technology interactions; they contribute to understandings of emergent historical processes; and they point toward future practices in creativity, research, scholarship and enterprise.

Proposals that address these challenges in terms of theory, research and practice, education, historical scholarship, discipline summaries and experimental texts will be considered. Single-authored books are particularly encouraged.

When submitting a proposal, bear in mind that we need to know as much as possible about the scope of your book, its intended audience and how best to bring the book to the attention of that audience. We need to be convinced that the material is important and that you can communicate clearly and precisely in ways your audience will appreciate.

Proposals should include (1) a prospectus describing the book, (2) a detailed table of contents, (3) two to four sample chapters, and (4) an up-to-date résumé/curriculum vitae for the author.

Full submission guidelines: <leonardo.info/isast/leobooks/guidelines.html>.

Inquiries and proposals should be submitted to both:

Leonardo Book Series and Doug Sery
c/o Leonardo
MIT Press Books
211 Sutter Street, Ste. 501
55 Hayward Street
San Francisco, CA 94108
Cambridge, MA 02142
U.S.A.

E-mail: <leonardobooks@mitpress.mit.edu>.

RECENT TITLES:

STEPHEN JONES: Synthetics: Aspects of Art and Technology in Australia, 1956–1975
NORIE NEUMARK, ROSS GIBSON AND THEO VAN LEEUWEN, editors: VOICE: Vocal Aesthetics in Digital Arts and Media
LAURA U. MARKS: Enfoldment and Infinity: An Islamic Genealogy of New Media Art
GEORGE GESSERT: Green Light: Toward an Art of Evolution
SARAH COOK AND BERYL GRAHAM: Rethinking Curating: Art after New Media
MATTHEW FULLER: Software Studies: A Lexicon
BEATRIZ DA COSTA AND KAVITA PHILIP, editors: Tactical Biopolitics: Activism and Technoscience

To order Leonardo Books, visit <leonardo.info>.