Evolutionary Art Revisited: Making the Process Fully Automated

ANDRÉS GÓMEZ DE SILVA GARZA and ARÁM ZAMORA LORES
Departamento Académico de Computación
Instituto Tecnológico Autónomo de México (ITAM)
Río Hondo #1, Colonia Tizapán-San Ángel, 01000—México, D.F.
MÉXICO

Abstract: - Many recent systems for computer artwork generation with evolutionary algorithms have been interactive, relegating the task of evaluating new genotypes to their user(s). In contrast, we are interested in fully automating the entire evolutionary cycle applied to artwork generation. In this paper we present an evolutionary computer system that, without human intervention, generates artwork in the style of the Dutch painter Piet Mondrian. Several implementation-related decisions that have to be made in order to program such a system are then discussed. The most important issue that has to be considered when implementing this type of system is the subroutine for evaluating the multiple potential artworks generated by the evolutionary algorithm, and our method is discussed in detail. We also discuss our results in relation to other research into the computer generation of artwork that fits particular styles existing in the real world with evolutionary algorithms.

Key-Words: - Evolutionary art, case-based reasoning, computational representations of style

1 Introduction

Ever since computer use began to spread, people started using computers to generate artwork. At first printers and monitors could not display anything other than text, so rough figures depicting cartoon characters or everyday objects such as houses were "painted" by displaying sequences of ASCII characters that were designed to create an outline of the desired figure (sometimes even incorporating shading, when viewed from a distance, by using different ASCII characters for different parts of the figure to be depicted). Eventually colors and high-resolution monitors and printers became available, so computer art evolved into what it is today, able to display a multitude of abstract or realistic figures and textures in such detail that the results can now be used in movies to create animation that is virtually indistinguishable from the results of real cinematography. Another approach to computer art, rather than to try to reproduce the real world virtually, is to try to get the computer to create its own sense of style (or rather, its programmer's) and produce artwork in that style that people consider to be aesthetically pleasing (usually an abstract geometric style). Yet another approach is to try to get the computer to create artwork that resembles (or is indistinguishable from) the creations of human artists.

It is this last approach that motivated the research presented in this paper. Specifically, we set out to produce a program that would generate artwork in the style of the Dutch painter Piet Mondrian, who was active mainly in the first half of the 20th century. Like many other modern painters, Mondrian started his career painting landscapes, human figures, and other realistic subjects, but eventually developed his own distinctive and abstract style (called simply *de stijl*, which is Dutch for "the style"). Paintings in Mondrian's style typically include vertical and horizontal black lines over a white background, with some or all of the primary colors (blue, red, and yellow), plus black, filling in some of the square or rectangular regions (or parts of the regions) separated out from the background by the black lines. It is this style that our system tries to emulate. Fig. 1 shows a typical Mondrian painting in his distinctive style.

The method we have used for getting the computer to generate different paintings is through an evolutionary algorithm [1]. This type of algorithm represents a generate-and-test, trial-and-error, brainstorming-like approach [2]: many possible paintings (populations of them) are generated quickly, by using mainly random decisions. Probably most of these paintings are of quite low quality, but after being generated they are then evaluated to determine how much they make sense. In the context of our research, "making sense" would imply being as close as possible to the style of Mondrian. The best paintings (according to the evaluation subroutine of the evolutionary

algorithm) are kept for future evolutionary generations (a process known as elitism), and the others are discarded (so as to keep the size of the population of the algorithm constant across generations). This process ensures a monotonic increase in the average quality of the paintings in the population between generations. Depending on what is desired, either when this average quality or when the quality of just one individual painting is good enough according to the evaluation subroutine, the process is terminated. The evaluation subroutine of the evolutionary algorithm, therefore, is of critical importance to the success of the approach.

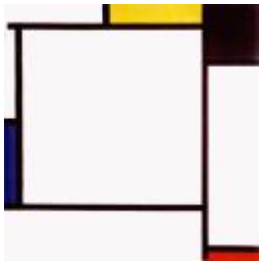

Fig. 1. A typical Mondrian painting

Section 2 of this paper provides a discussion of related work in computer artwork generation and representation of style which serves to further frame and provide motivation for our work. Section 3 briefly talks about our evolutionary algorithm, and discusses some decisions we had to make in order to implement it for the domain of Mondrian-style artwork generation in particular. This implementation has resulted in a system named MONICA (MONdrian-Imitating Computer Artist). Finally, Section 4 presents some of the results we have obtained from running MONICA and provides a general discussion of these results and possible future work on the project.

2 Related Work

Just as generating artwork by computer is not a new idea, neither is using evolutionary algorithms for this task. Two recent surveys, published in book form, of the use of evolutionary algorithms for design tasks in general, are included for instance in [3] and [4], and these include discussions of seven systems for artwork generation. All of the systems described in these books that create artwork ([5], [6], [7], [8], [9], [10], and [11]) do so by using the evolutionary operators of crossover and mutation. However, all of these systems leave it to the user(s) to decide which of the new paintings (or which of their features) to keep for future evolutionary cycles, and/or how to rank the new paintings according to their subjective (and probably unconscious) aesthetic criteria. Thus, the decisions on what is aesthetic or interesting, or what paintings fit a particular style, are not made by the systems. In contrast, MONICA is designed to be a fully autonomous system requiring no user feedback as its evolutionary algorithm proceeds.

In order for MONICA to be fully autonomous, an evaluation procedure that captures and recognizes the stylistic characteristics of Mondrian paintings had to be programmed into its evolutionary algorithm. Neither the relationship between evolutionary algorithms and Mondrian, nor the attempt to capture and automate the generation of new creations in the style of given artists or designers, is new either. However, the approach we followed in implementing MONICA is different from those that have been used in other projects that have explored these issues.

A Mondrian Evolver (as well as an Escher Evolver) is mentioned in [11]. However, the Internet web-page cited does not seem to exist anymore, so we have not been able to view the program. If it works the same way as the Escher Evolver described in the book chapter cited, then anyone who accesses the Mondrian Evolver web-page can provide feedback to the program in order to influence the results of the next evolutionary cycles. Thus, the system is again non-autonomous, unlike MONICA.
A Mondrian Applet [12] and a Mondrian Machine [13] can be found on the Internet. The first one of these two systems is completely autonomous, and the second one semi-autonomous (because the user's clicks determine the positions of black lines in the paintings which are then automatically generated, with colors being randomly selected to fill some of the spaces between the resulting lines). Both systems have been programmed with certain rules about how to generate artwork that looks like Mondrian's according to their programmer's understanding of *de stijl*. The difference with our work is that generating new paintings in MONICA is done completely at random through the evolutionary operators of crossover and mutation, not by following any pre-programmed generative rules. It is only in the evaluation subroutine in MONICA's evolutionary algorithm that any knowledge of Mondrian's style (according to our understanding of it) is programmed, and this knowledge in MONICA is used for style recognition rather than generation or emulation.

Returning to evolutionary algorithms, one of these has in the past been applied to generating Mondrian-like artwork, as reported by Schnier and Gero in [14] and [15]. The genotype representation used in these two publications is hierarchical (much like that used in genetic programming [16]), and

relies on the observation that a large subset of Mondrian's paintings can be described by successive recursive divisions of a "canvas" into rectangular areas. It is the recursive subdivisions that are embodied in the hierarchical aspects of the representation. The fitness of new paintings generated by this system is determined by measuring the distance between them and the exemplars (cases) of real Mondrian paintings used to initiate the evolutionary process. The non-hierarchical genotype representation used in MONICA provides much less guidance to the system as to the structure that a Mondrian-like painting should have in comparison to the un-named system described in the two papers by Schnier and Gero. Another difference is that MONICA's evaluation subroutine explicitly captures stylistic characteristics used by Mondrian and observed by the authors, whereas the evaluation module in the system described by Schnier and Gero does not. On the other hand, the use of real cases of Mondrian paintings and the overall evolutionary framework and attempt to obtain Mondrian-like results create a close similarity between the two projects.

Work into capturing the style of particular artists or designers in the computer has often focused on shape grammars [17] or semantic networks [18]. However, some work has also used evolutionary algorithms to explore style. A system is described in [19] that generates traditional Chinese architectural facades after it infers a representation of their style. The inference and subsequent learning is done with an evolutionary algorithm which in the end obtains a hierarchical genotype which represents a particular style (by generalizing from the exemplars which have been shown to it). As with the project described by Schnier and Gero, here we have exemplars and we have hierarchical genotypes. However, in this project the genotype does not describe a particular design, but rather an entire design style. This is because the genes that compose the hierarchical genotype embody syntactic and semantic primitives that combine to describe a particular style. The task of the system is to learn this hierarchical genotype given the exemplars, and the learning task is done through an evolutionary algorithm. Recognizing whether a new design matches a particular style involves matching its features with the style representation embodied in the hierarchical genotype. In contrast, in MONICA style recognition is pre-programmed, not learned, and forms part of the evaluation module of the evolutionary algorithm that generates possible Mondrian-like paintings, not part of the knowledge captured in the genotype representation.

3 Evolutionary Algorithm Implementation

Fig. 2 shows the flow of subtasks performed by our evolutionary algorithm.

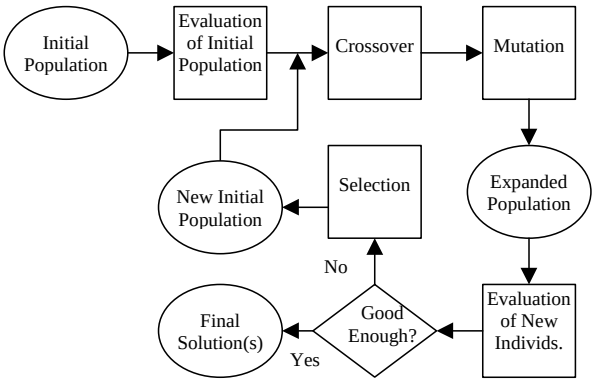

Fig. 2. Our evolutionary algorithm's subtasks

Briefly, a population of potential solutions (which in this case represent paintings) is kept throughout the process. The makeup of the population changes due to the evolutionary process. New potential solutions are generated by the genetic operators of crossover and mutation, which at random combine and modify the features of old potential solutions that are already in the population, respectively. A temporarily expanded population is created by adding these new potential solutions to the original population. Each new potential solution is evaluated and a fitness value assigned to it. The fitness value in our system is a measure of how close or how far the painting is from Mondrian's style. If one or more (as desired) of the new potential solutions is already perfect (i.e., already fits the desired style) according to the evaluation procedure, the evolutionary process stops (or, alternatively, if the search has gone on for too long without successfully producing any Mondrian-like works of art). Otherwise, a selection procedure sorts the potential solutions in the temporarily expanded population according to their fitness value, keeps the best of them, and discards the rest. The individuals that are kept become the initial population for the next evolutionary cycle (which has been trimmed down to be of the same size as the initial population of the previous evolutionary cycles). This new population may include both old and new potential solutions (i.e., some carried over from previous generations and some newly-generated ones), and the process repeats itself.

The system we have programmed, MONICA, implements an evolutionary algorithm like the one

described above on a Windows platform in C++ (for the reasoning engine) with OpenGL (for the graphical interface), and applies the algorithm to the domain of Mondrian-style artwork generation. The following subsections discuss some issues that arose during the implementation of MONICA.

3.1 Makeup of the Initial Population

As can be seen from Fig. 2, the population of the first cycle of the evolutionary algorithm must be seeded in some way before starting the process. We have considered several options in order to produce this initial population. One option for this seeding process is to use cases (in this situation, actual Mondrian paintings) as the individuals in the initial population. A process model for this approach, used in the domain of residential floor plan design, is presented and evaluated in [20] and [21]. The cases used to seed the initial population include examples of different things that Mondrian has done in his paintings, and thus, when combined and modified repeatedly, provide material for the evolutionary algorithm to quickly produce new potential solutions that contain Mondrian-like features. On the other hand, the use of cases can bias the algorithm too much towards creating paintings that only differ from Mondrian's in small details. We have gathered 55 cases of Mondrian paintings (pertaining to *de stijl* and not including pieces representing stylistic transitions or his rhomboidal "lozenges," as they are known) from several Internet sites, and from the books [22] and [23]. Apart from having these cases in JPG format, we have made them available to MONICA by hand-coding them into our genotype representation scheme (described below).

A different option for the seeding process is to produce an initial population entirely at random. This approach has the advantage that little prior bias can be said to exist in the initial population, and thus it is easier to claim that the computer itself really evolved Mondrian-like paintings from nothing, without any help. On the other hand, it may take too long to start producing Mondrian-like artwork, despite the bias that the evaluation procedure of the evolutionary algorithm provides in that direction, and thus may not really be a feasible approach. This method of seeding the initial population is also available to MONICA. The algorithm used in MONICA in order to produce a random initial population is based on randomly deciding the sizes, coordinates, and colors of different rectangular regions to be placed in a painting. Some slight bias that considers aspects of Mondrian's style, such as limiting the total number of colored regions in each individual to Mondrian-type numbers, was still present in the "random" generation of the initial population, but not much. In our algorithm, colored regions that end up getting assigned very small widths or heights end up being lines, but are not treated any differently from square or rectangular colored regions. There is nothing in the random seeding algorithm to force colored regions not to overlap, not to exceed the limits of the frame used for the painting, or any other things that will eventually have to happen, at the same time, for a painting to be accepted as Mondrian-like by the evaluation procedure.

A final option is to combine both cases and random individuals in the initial population. The question then becomes what the relative proportion of cases and random individuals should be in the initial population. For the purposes of being able to produce Mondrian-like results, it doesn't seem to matter much, though there may be an effect on convergence time. We are planning to perform some experiments with MONICA in the future to measure any possible effects. In addition, [24] uses a similar approach, using cases combined with evolutionary algorithms for electronic circuit design, and has performed experiments having to do with seeding the population with different proportions of cases and random "solutions." These experiments have measured the effect of changing the relative proportions of cases and random solutions on the amount of exploration and exploitation present in an evolutionary algorithm, and have come to the conclusion that around 10% of cases is a good figure. It will be interesting to see if our future experiments validate these results or if the results are specific to the domain and representation used in [24].

3.2 Genotype Representation

Fig. 3 shows how we have represented in MONICA the individuals in the population of our evolutionary algorithm as genotypes at the highest level. The genotype is split into twenty regions, each of them corresponding to one of twenty possible colored regions permitted in a painting.

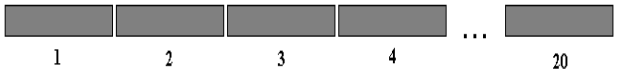

Fig. 3. Genotype representation used

Fig. 4 shows at an intermediate level how each of the twenty genotype regions is split into five sections in order to represent the color, width, height, and x- and y-coordinates (of the center, as per OpenGL standards) of each colored region. These

last four measurements are all limited to the same range of values.

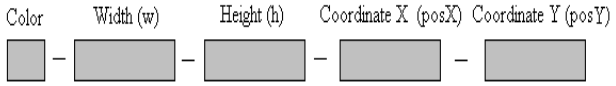

Fig. 4. Representation of each colored region

Fig. 5 shows, at the bit level, the internal details of the representation of the color and one of the four measurements shown in Fig. 4.

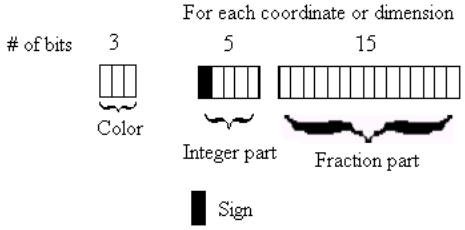

Fig. 5. Bit-encoding of the color and any one measurement in the representation

3.3 Crossover and Mutation Operators

The genetic operators of crossover and mutation, in their pure form, should make all their decisions at random. For instance, crossover combines the makeup of two original individuals by randomly choosing which two individuals to combine and randomly choosing the position within their genotypes in which to cut and splice them. However, this completely "blind" process can produce a lot of senseless and time-wasting results, including offspring genotypes that have the exact genetic makeup as their parents. In MONICA, if a particular painting has only four colored regions, for example, and given the fact that all genotypes are of the same length, the genes in the genotype corresponding to colored regions 5-20 will just be filled with zeroes. If the crossover point is randomly chosen to fall in this region of the genotype for one parent, and the same occurs with the other parent, each one of the two resulting offspring genotypes will be identical to one of their parents. Thus, we would not have produced any new, and therefore potentially good, results in the new generation. In order to avoid this problem we have added some "intelligence" to our implementation of both the crossover and mutation operators in order to ensure that any work done during evolution results in something new.

We could also add some intelligence to MONICA in order to avoid any useless results. For instance, we only accept five possible colors in a Mondrian-like painting, yet have three bits in which to represent the value of the color gene within an individual's genotype (leading to eight possible values). Because of this, the value of the left-most bit position in the color gene will, most of the time, be 0 in a good genotype, unless the two right-most bits are 0's, in which case the left-most one can be a 1. We could program the crossover and mutation operators to take into account these factors when making their "random" decisions about where to cross or mutate genotypes, in order for all the genes of offspring genotypes to get assigned values that make sense. However, we feel that this would be tampering too much with the way evolution is supposed to happen, so we have not gone so far. Again, any biasing of the population according to how much the individuals in it resemble Mondrian's style we have left to the MONICA's evaluation procedure, and we did not want to distribute its effects to other modules.

3.4 Evaluation of New Genotypes

The evaluation of new genotypes produced during the course of evolution is the most important aspect of an evolutionary algorithm, since it is where domain knowledge is used to indirectly guide the evolutionary search towards a particular objective. In MONICA we assign a fitness value between 0 and 1 to each individual in the population, where a 0 would mean that the individual doesn't satisfy any of the evaluation rules used to determine how close it is to fitting Mondrian's style, and a 1 would represent a perfect fit. We have implemented eight evaluation rules, each of which assigns a "local" fitness value between 0 and 1 (interpreted as above), and calculate the total fitness of an individual by adding the eight local fitness values and dividing the total by eight, thus giving each rule an equal importance or weight. The eight rules were articulated (and then programmed) by the authors after examining the 55 cases of Mondrian paintings that we had access to, and discussing the patterns that we seemed to observe in these cases. We have shown in [25] that most people agree with the results given by our evaluation rules. The eight rules are the following:

1. EvaluateColor: Each colored region that is contained in a genotype must have one of the five valid colors.
2. EvaluateCoordinates: The height, width, x-coordinate, and y-coordinate of each colored region in a genotype must all fall between 0 and 3.9999.
3. EvaluateLineThickness: Up to two black colored regions are allowed in a genotype that are not thin, but all other black regions must be either vertically

or horizontally thin (and thus represent a line rather than a rectangular region).

4. EvaluateNumberOfVerticalLines: A minimum of two and a maximum of ten vertical lines must be present in a genotype.

5. EvaluateNumberOfHorizontalLines: A minimum of two and a maximum of ten horizontal lines must be present in a genotype.

6. EvaluateLimits: Each colored region in a genotype must be adjacent either vertically (both above and below) or horizontally (both to the left and to the right), or both, to another colored region or to the edge of the frame (with some small tolerance).

7. EvaluateFrame: All other colored regions in a genotype must fall within the coordinates of the frame, whose color is white by definition and whose coordinates are represented just as any other colored region's are.

8. EvaluateNumberOfColoredRegions: There must be at least one colored region represented in a genotype, and at most 13, not counting lines. At most one of them can be white (and represents the frame).

4 Discussion and Results

Fig. 6 shows three Mondrian-like paintings which were created by our system (which assigned a fitness value of 1 to them) at different times. The fact that MONICA has shown the capacity to create Mondrian-like paintings on several occasions validates the set of evaluation rules we programmed into the system and also confirms the capacity to create artwork by computer autonomously, without the need for user feedback or intervention.

This kind of project helps contribute to several interrelated research fields, notably design computing, case-based reasoning, and evolutionary algorithms. Most importantly, the notion of "style" is still not well understood in a formal manner, even though most people probably have an intuitive feel for its meaning and can recognize different styles. Computational models of style generation and/or style recognition like the one used in MONICA help to further our formal understanding of style in a way that the artwork-generating systems described in [3] and [4] can't.

Future work on MONICA will focus on not having to pre-program any style recognition knowledge at all in order to create the evaluation module of the evolutionary algorithm. Perhaps a similar approach to the one presented in [19] can be used, employing first a learning evolutionary algorithm to evolve a representation of Mondrian's style, and then the current evolutionary algorithm to create possible Mondrian-like paintings, but to evaluate their fitness based on the learned representation rather than using hand-coded rules for this evaluation. Or perhaps a different method (e.g., we are considering neural networks) will be able to capture Mondrian's style in such a way that it can be used by MONICA's evolutionary algorithm, but without having to program any of its evaluation or recognition knowledge explicitly. For the moment, pre-programming a particular style into the system was not too difficult, given Mondrian's geometric and relatively simple style. However, applying the same ideas used in MONICA in order to imitate other painters' styles in the computer (e.g., da Vinci) might require a different approach!

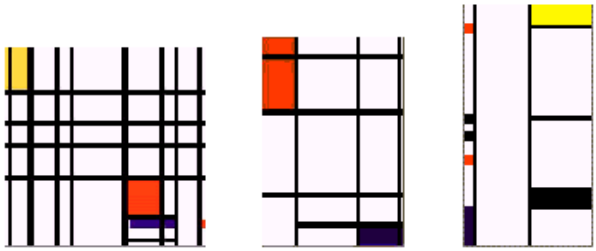

Fig. 6. Three Mondrian-like paintings generated by our system

References:

[1] M. Mitchell, *An Introduction to Genetic Algorithms (Complex Adaptive Systems Series)*, MIT Press, Cambridge, Massachusetts, 1998

[2] C.H. Clark, *Brainstorming: The Dynamic Way to Create Successful Ideas*, Doubleday, Garden City, New York, 1958

[3] P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, California, 1999

[4] P. Bentley and D.W. Corne (eds.), *Creative Evolutionary Systems*, Morgan Kaufmann Publishers, San Francisco, California, 2002

[5] S. Todd and W. Latham, The Mutation and Growth of Art by Computers, in P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, California, pp. 221-250, 1999

[6] T. Witbrock and S. Neil-Reilly, Evolving Genetic Art, in P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, California, pp. 251-259, 1999

[7] A. Rowbottom, Evolutionary Art and Form, in P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, California, pp. 261-277, 1999

[8] S. Rooke, Eons of Genetically Evolved Algorithmic Images, in P. Bentley and D.W.

Corne (eds.), *Creative Evolutionary Systems*, Morgan Kaufmann Publishers, San Francisco, California, pp. 339-365, 2002

[9] L. Pagliarini and H.H. Lund, Art, Robots, and Evolution as a Tool for Creativity, in P. Bentley and D.W. Corne (eds.), *Creative Evolutionary Systems*, Morgan Kaufmann Publishers, San Francisco, California, pp. 367-385, 2002

[10] P.J.B. Hancock and C.D. Frowd, Evolutionary Generation of Faces, in P. Bentley and D.W. Corne (eds.), *Creative Evolutionary Systems*, Morgan Kaufmann Publishers, San Francisco, California, pp. 410-423, 2002

[11] A.E. Eiben, R. Nabuurs, and I. Booij, The Escher Evolver: Evolution to the People, in P. Bentley and D.W. Corne (eds.), *Creative Evolutionary Systems*, Morgan Kaufmann Publishers, San Francisco, California, pp.425-439, 2002

[12] C. Lendon, *Mondrian Applet*, http://www4.vc-net.ne.jp/~klivo/soft/mondrian.htm, 1999

[13] M. Lewis, *Mondrian Machine*, http://desires.com/2.1/Toys/Mondrian/mond-fr.html and http://www.ptank.com/mondrian, 1996

[14] T. Schnier and J.S. Gero, J.S., Dominant and Recessive Genes in Evolutionary Systems Applied to Spatial Reasoning, in A. Sattar (ed.), *Advanced Topics in Artificial Intelligence (10th Australian Joint Conference on Artificial Intelligence AI-97 Proceedings)*, Springer-Verlag, Heidelberg, Germany, pp. 127-136, 1997

[15] T. Schnier and J.S. Gero, From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations, in I. Parmee (ed.), *Adaptive Computing in Design and Manufacture III*, Springer-Verlag, London, pp. 207-219, 1998

[16] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachussets, 1992

[17] M.-Y. Cha and J.S. Gero, Style Learning: Inductive Generalisation of Architectural Shape Patterns, in A. Brown, M. Knight, and P. Berridge (eds.), *Architectural Computing from Turing to 2000, eCAADe*, University of Liverpool, England, pp. 629-644, 1999

[18] J.R. Jupp and J.S. Gero, Feature Based Qualitative Representation of Architectural Plans, *Proceedings of the Computer-Aided Architectural Design Research in Asia Conference (CAADRIA-03)*, Rangsit University, Patumtani, Thailand, 2003

[19] L. Ding and J.S. Gero, The Emergence of the Representation of Style in Design, *Environment and Planning B: Planning and Design*, 28(5), pp. 707-731, 2001

[20] A. Gómez de Silva Garza and M.L. Maher, A Process Model for Evolutionary Design Case Adaptation, in J.S. Gero (ed.), *Artificial Intelligence in Design '00*, Kluwer Academic Publishers, Worcester, Massachusetts, pp. 393-412, 2000

[21] A. Gómez de Silva Garza and M.L. Maher, GENCAD: A Hybrid Analogical/Evolutionary Model of Creative Design, in J. Gero and M.L. Maher (eds.), *Computational and Cognitive Models of Creative Design V*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 141-171, 2001

[22] S. Deicher, *Mondrian*, Benedikt Taschen Verlag GmbH, Cologne, Germany, 1999

[23] M. Bax, *Complete Mondrian*, Lund Humphries (Ashgate Publishing), Aldershot, United Kingdom, 2001

[24] S.J. Louis, Learning from Experience: Case Injected Genetic Algorithm Design of Combinatorial Logic Circuits, in I.C. Parmee (ed.), *Adaptive Computing in Design and Manufacture V*, Springer-Verlag, Berlin, Germany, pp. 295-306, 2002

[25] A. Gómez de Silva Garza and A.Zamora Lores, A Cognitive Evaluation of a Computer System for Generating Mondrian-Like Artwork, in J.S. Gero (ed.), *Design Computing and Cognition '04*, Kluwer Academic Publishers, Dordrecht, the Netherlands, pp. 79-96, 2004