SHIFT THE STYLE: Supporting Product Design through Evolving Styles

TING-YUAN CHANG  
*National Yunlin University of Science and Technology, Taiwan.*  
g9234714@yuntech.edu.tw

AND

SHENG-FEN CHIEN  
*National Taiwan University of Science and Technology, Taiwan.*  
schien@mail.ntust.edu.tw

**Abstract.** How can we provide computational supports for product designers to actively utilize existing styles as driving force to achieve new designs? For a new style that can take the advantage from existing styles, we surveyed Case-Based Design and Case-Based Reasoning systems for product design and found discouraging results. We employ Learning Classifier Systems, a machine-learning paradigm, to perform automatic style classification. These classified styles are used to evolve new styles of designs.

1. Introduction

Given the ever-shortening lifecycle of products and the ever-growing competitiveness of the market, developing news products that meet consumers’ needs and tastes is critical in product design. Furthermore, consumers are becoming more picky with the style of a product and less about the usage functions. What is a style? The explanation is usually fuzzy and abstract. In product design, it is the esteem and aesthetic functions that constitute a style. These functions include brand image, personal aesthetics, and current trends or fashion.

We have seen some long-lasting styles, and the rebirth of old styles. We have also encountered very short-lived styles. For a designer, all these styles are sources of design inspiration. How can we provide computational supports for product designers to actively utilize existing styles as driving force to achieve new designs? In particular, we are interested in providing a design aid that is able to learn from new inputs.

2. Related Works

For a new style that can take the advantage from existing styles, we surveyed Case-Based Design (CBD) and Case-Based Reasoning (CBR) systems for product design
and found discouraging results. In the domain of building design, Heylighen and Neuckermans (2001) reviewed several CBD/CBR systems and reported gloomy results. Gomez de Silva Garza and Maher (2001) began to energize CBR with evolutionary methods.

For the learning capability, we reviewed systems that support learning through data classifications. Here, again, CBR systems provide such support. However, most research works in CBR and classification are in the domain of medicine and stock/bond market prediction. Less than a handful of publications in building related domains, such as Fenves et al., (2000) and Flemming and Aygen (2001) are found. In addition to CBR, genetic algorithms are also popular in classification systems (for example: Holmes et al., 2000; Kuri, 1998). In the field of pattern recognition research, genetic algorithms are widely utilized.

We deliberately neglect the method of neural networks because we are interested in maintaining the reasoning process. Although related works employing neural networks have been successful, neural network systems operate in a black box through a process that is not understandable by designers. We believe a design aid should interact with designers in an open process so that designers can always verify the information provided by the design aid.

3. Style Classification

Style classification is a knowledge-intensive and time-consuming activity. The process usually begins with a rigorous analysis design artifacts to identify key features and follows by an expert evaluation of these artifacts against those key features (for example, see: Chan, 2000; Chen and Owen, 1997; Muller and Pasman, 1996). The resulting classification may suffer from personal biases (Chen and Owen, 1997). In addition, modern designs are geared toward consumers’ expectations (Tsuchiya et al., 1996) therefore styles classified by experts may not reflect that by consumers.

To circumvent human biases, to quickly adapt to consumer needs, and to obtain the reasoning process of style classification, we consider style classification as a knowledge discovery process. We find that Learning Classifier Systems (LCSs) are very suitable to extract compact descriptions of interesting phenomena (i.e. features of styles) described by multidimensional data (i.e. products). LCSs are a machine-learning paradigm. LCSs exploit evolutionary computation and reinforcement learning to develop a set of condition-action rules (Holmes et al., 2002). Holmes and his colleagues (2002) summarize, in general, LCS models consist of four main components:

1. A finite population of condition-action rules (classifiers) that represents the current knowledge of the system;
2. The performance component that interacts with the environment, which provides feedback in the form of numerical reward;
3. The reinforcement component that distributes the reward received from the environment to the classifiers; and
4. The discovery component that is responsible for discovering better rules through a genetic algorithm.

Holmes et al. (2000) and Sette and Boullart (2000) have provided excellent illustrations on using LCS for knowledge discovery. However, these efforts are limited to problems of a single class. Stefanowski (2004) summarizes methods, in particular wrapper model and bagging approach, to assign a decision class label to a set of unclassified objects described by a set of attributes.

To discover, potentially multiple, styles from a database of unclassified products, we refine the four-component LCS model into a five-component style classification system (Figure 1). In additional to the original four components, the consolidation component is added to prevent the excesses of styles (i.e. actions/decision classes).

![Figure 1. The style classification system.](image)

In the style classification system, classifiers are defined as a condition string and a style class integer.

$$\text{classifier} = \text{condition} + \text{style class} \quad (1)$$

$$\text{condition} = \{0, 1, \#\} \quad (2)$$

$$\text{style class} = \{n \mid n = 0, n \in \mathbb{N}\} \quad (3)$$

The “#” symbol represents a “wildcard”. In a condition string, a “#” matches to either 0 or 1. A style class identifies a classification action when its associated condition is met. A style class value 0 represents unclassified; 1 or above represents different styles.
For experimentation, we focus on the style of automobiles and select the front-end portion of the automobile as sampling data. The condition portion of the classifier encodes features of radiator shape, radiator pattern, headlight and hood (Table 1). The style class portion records an integer number.

TABLE 1. Classifier format for the automobile style classification.

<table>
<thead>
<tr>
<th>grill shape</th>
<th>grill pattern</th>
<th>headlight</th>
<th>hood</th>
<th>style class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Coding</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The discovery component begins by generating a classifier at random. In the case of our experimentation, the length of code is 12-digit. We control the appearance of wildcard (#) to no more than three. New classifiers are evolved through a mutation operator and a single-point crossover operator. New classifiers are initialized with style class zero, which is changed to a non-zero value when a match is found.

The classification system needs to be initialized through a training session. During the training session, the performance component receives all sample data from the environment. Each sample is matched against an active classifier. When a match is found and if the style class portion of this classifier is 0, the style class value is set to an integer that is different from that of all other classifiers. Matched samples are recorded in the match set. The reinforcement component computes a reward based on the number of matches and the total number of sample data. When an active classifier finishes matching against all samples, the consolidation component examines records in matching sets, if there exists two identical matching sets then the associated classifiers are considered to have the same style. In this case, the style class portion of these two classifiers is modified to one identical code. At the end of the training session, classifiers with no rewards are purged.

After the training session, the style classification system will contain a set of rules to identify different styles. A new sample is then matched directly against the whole classifier population to determine its style.

4. Style Evolution

Style classification is a means to reach the real aim of this research—to assist designers evolve new styles (similar to Liu et al., 2004). At the time of this writing, we have yet to implement the style evolution system. We shall, instead, report the design of this system.

The style evolution system is a design aid rather than an automatic style generator. It is design to interact with designers and to allow designers guiding the
style evolution process. The interaction is provided through two levels. At the design product level, a designer may use existing design cases (samples) as bases to evolve new designs. Alternatively at classifier level, a designer may use classifiers as style prototypes to evolve new styles and generate new designs.

At the design product level, the system assists designers to select product of different style to evolve and to transfer the style from one product to another. Designers can pick different parts of gene from the initial population (existing design cases) to be the fitness function’s weighted rules and make the mixed style product. To ensure the diversity, the system will increase the probability of mutation and join different operators such as Swap, Insert, and Delete. The system can quickly classify the newly created design product to examine the shifting of design styles.

At the classifier level, the system assists designers to select style prototypes (classifiers) to evolve new style prototypes. Designers examine these new style prototypes through potential design products that are generated by the system as soon as the new style is created. Designers may request selected items from the potential design products to be reclassified or instruct the system to reexamine all existing design cases against the newly evolved style prototypes. This level of interaction is the direct manipulation of shifting styles, while the design product level of interaction is an indirect manipulation of styles through operating on the end products.

Throughout these interactions, the system may provide explanations (through deducing from classifiers) on styles and reasoning (classification) processes. We envisage the explanation may facilitate the design exploration process.

5. Discussions

Existing design cases (design precedents) are foundations to new designs and in many ways also the source of creativity. We have attempted to employ evolutionary computation to achieve what CBD/CBR systems had tried to accomplish. Our work is still in progress. At the moment, we are able to perform style classification with a fairly limited set of design cases using the LCS approach. The LCS is an old concept but according to Holmes and his colleagues (2002), new LCS models with better adaptivity, generalization and scalability are good for knowledge discovery (data mining). It may supplement the learning and adaptation in CBR.

For the eventual product of this research, we hope with such design aid, designers may harness the computational power to extend their design experience from direct end-product manipulation (for example: drawing forms or making models by hand) to an indirect form manipulation (for example: operating on classifiers), to enlarge the design space (exploring more design alternatives), yet still can maintain the consistency of styles with ease.
References