Applying machine learning to automated information graphics generation

by M. X. Zhou
S. Ma
Y. Feng

Information graphics, which include graphs, charts, and diagrams, are visual illustrations that facilitate human comprehension of information. In this paper, we present our work on applying machine learning to the automated generation of information graphics. Our approach is embodied in a hybrid graphics generation system, IMPROVISE*, which uses both rule-based and example-based generation engines. We discuss the use of machine learning to support such systems from three aspects. First, we introduce an object-oriented, integrated hierarchical feature representation for annotating information graphics. Second, we describe how to use decision-tree learning to automatically extract design rules from a set of annotated graphic examples. Our results demonstrate that we can acquire, with quantitative confidence, concise rules that are difficult to obtain in handcrafted rules. Third, we present a case-based learning method to retrieve matched graphic examples based on user requests. Specifically, we use a semantics-based, quantitative visual similarity measuring algorithm to retrieve the top-k matched examples from a visual database. To help users browse a graphics database and understand the inherent relations among different examples, we combine our similarity measuring model with a hierarchical clustering algorithm to dynamically organize our graphic examples based on user interests.

Information graphics, which include graphs, charts, and diagrams, are visual illustrations that facilitate human comprehension of information. To help users understand information and achieve their information-seeking goals (e.g., understanding the trend of the real-estate market for a particular area), a number of systems have been built to design information graphics based on user's information needs (e.g., visualizing a file system using a cone tree[1]). However, information graphics created by these systems are carefully designed and preprogrammed by hand. Handcrafting information graphics is a difficult and time-consuming task, and relatively few information system developers have had training in graphic design. Moreover, handcrafting information graphics in advance is inadequate for an interactive environment, where different users may have different tasks in mind (e.g., comparing two data items vs understanding a trend) or may change their tasks in the course of information exploration. To support dynamic design and customization, researchers have been investigating approaches to the automated generation of information graphics. There are two main approaches: rule-based and example-based graphics generation.

For the past decade, researchers have been developing systems that can automatically create information graphics using a set of design rules. Given a set of data entities, a presentation intent (e.g., *sum-*

0018-8670/02/$5.00 © 2002 IBM

marize a set of information), and a presentation context (e.g., a specific type of users and presentation devices), an automated graphics generation system automatically constructs a customized information graphic that communicates the input data to the intended users in the specified context. For example, systems like APT,[2] SAGE,[3] and IMPROVISE[4] employ handcrafted design rules to map data onto proper information graphics. Nevertheless, the rule-based approaches present two major problems. First, acquiring design rules manually is difficult. Handcrafting design rules can be laborious, as experts may need to process large amounts of evidence before extracting rules. Second, maintaining and extending a large rule base is difficult. As the rule base grows, it is difficult to integrate new rules with existing rules, and to discover and rectify inconsistencies among the rules.

Alternatively, researchers have started to explore example-based design approaches.[5,6] Instead of using rules, an example-based approach starts with a set of existing examples. Upon a user's request (e.g., find presentations that are suitable for my data), this approach searches through a graphics database and retrieves the most relevant examples. The retrieved examples can then be directly reused for, or adapted to, the new situation (e.g., new data or new visual preference). However, an example-based approach alone is usually inadequate, because it is difficult to learn all low-level design details directly from examples (e.g., the exact scale and position of every graphic element).

To overcome these problems, we are exploring a hybrid approach, which makes use of both rule-based and example-based generation. Our approach is embodied in a prototype system, IMPROVISE*, which is an extension of our previous rule-based generation system, IMPROVISE.[4] On the one hand, IMPROVISE* employs an example-based approach to improve extensibility, allowing new graphic examples to be easily added and reused. On the other hand, IMPROVISE* utilizes rules to determine various graphic details efficiently, complementing example-based generation.

Our focus here is on applying machine learning to support systems like IMPROVISE*. To the best of our knowledge, our work is the first to apply machine learning to automated information graphics generation. Our contributions are threefold. First, we introduce a unique feature selection/extraction method for information graphics design. In particular, we define an object-oriented, integrated hierarchical feature representation for annotating information graphics. As described later, our annotation captures cohesively both data and visual characteristics. Our annotation formalism provides the needed foundation for applying machine learning to automated graphics generation.

Second, we address how to systematically acquire design rules from existing graphic examples. Specifically, we employ a decision-tree learning algorithm to induce design rules automatically from a set of annotated graphic examples. Our results demonstrate that machine learning helps the system to acquire useful design rules automatically and to improve the overall quality of a rule base (e.g., by simplifying expert-derived rules and removing redundancies).

Third, we investigate how to create new information graphics directly from existing graphics to augment our previous rule-based generation approach.[4] In particular, we develop a case-based learning method. We use a semantics-based, quantitative visual similarity measuring model to retrieve top-matched graphic examples, which can then be directly reused for, or be adapted to, the new situation.

The paper is organized as follows: first we discuss related work, then provide an overview of IMPROVISE*. We then introduce an object-oriented, integrated hierarchical feature representation for annotating graphic examples. Using our annotation formalism, we present the systematic application of decision-tree learning to extract design rules from annotated graphic examples. We then describe a case-based learning method that uses a semantics-based quantitative visual similarity measuring model to retrieve relevant graphic examples. Finally we conclude and indicate future research directions.

Related work

Before presenting our machine-learning approaches to information graphics generation, we discuss related work from two aspects: machine learning in rule acquisition and example-based graphics generation.

Automatic rule acquisition. Machine-learning techniques, especially supervised learning, have been widely applied to rule acquisition in other domains (e.g., speech synthesis[7]). However, applying these techniques to the acquisition of graphic design rules has hardly been addressed. One of the causes lies

in the difficulty of expressing information graphics accurately and comprehensively.

To identify features for machine learning, we have adopted previous research results on data and graphics characterization. On the data side, researchers establish data characterization taxonomies to abstract what and how presentation-related data properties influence visual encoding strategies.[8–10] To describe visual patterns systematically, on the other hand, researchers characterize different visual formats,[11] formulate a set of graphics languages,[2] and define the formal syntax or semantics of a particular visual representation.[12] However, we have extended these efforts to characterize both the data and visual patterns more comprehensively. In particular, we developed an object-oriented hierarchical feature representation to uniformly express the semantic, meta-level, and structural properties of a graphic at multiple levels of abstraction. This is also different from the conventional feature representation used in many machine-learning applications, where a flat structure of simple features is used.

Example-based graphics generation. Programming by demonstration is perhaps the earliest technique that supports example-based graphics generation.[13] Through user demonstration, this approach attempts to generalize the behavior to a design principle, which can then be applied to an entire class of tasks in the future. However, coming up with the right set of examples requires design knowledge on the part of the user.[14]

Instead of asking users to supply the right examples, a recent example-based generation approach allows a user to select an example to be automated from the user's past requests, or operations.[6] The user can modify the selected example or refine it for performing future tasks. This approach also uses a hierarchical representation to express the semantics of stored examples and new user operations. However, it matches a new user operation with a single selected example based solely on the similarity of lowest level components, without considering compositional structures. In contrast, IMPROVISE* matches a user request against multiple examples, and it takes into account both content and structural similarities and differences.

Another closely related example-based generation system is Sage/SageBook.[5] Upon a user's request, Sage searches its stored visual presentations and retrieves those relevant for reuse in the new situation. It uses both data features and visual properties to describe the stored presentations and user requests. Moreover, Sage matches a user request with a presentation example by qualitatively comparing lowest level data and visual contents (e.g., data elements and graphemes). IMPROVISE* differs significantly from Sage in its use and representation of examples and in its method for matching requests to examples. First, Sage uses only examples that it creates,[3] whereas IMPROVISE* exploits graphic examples from a wide variety of sources, which may or may not be generated by our system. Second, Sage employs a flat data characterization, separated from its hierarchical visual feature description. In contrast, we express both visual and data features hierarchically and integrate visual features with their corresponding data features at every level of the abstraction. Third, Sage uses a qualitative matching method to retrieve desired examples, whereas we develop a quantitative similarity measuring method to facilitate a more accurate comparison between examples and user requests. In addition, we allow users to dynamically adjust various weights in our similarity model and to submit partial requests with different matching criteria.

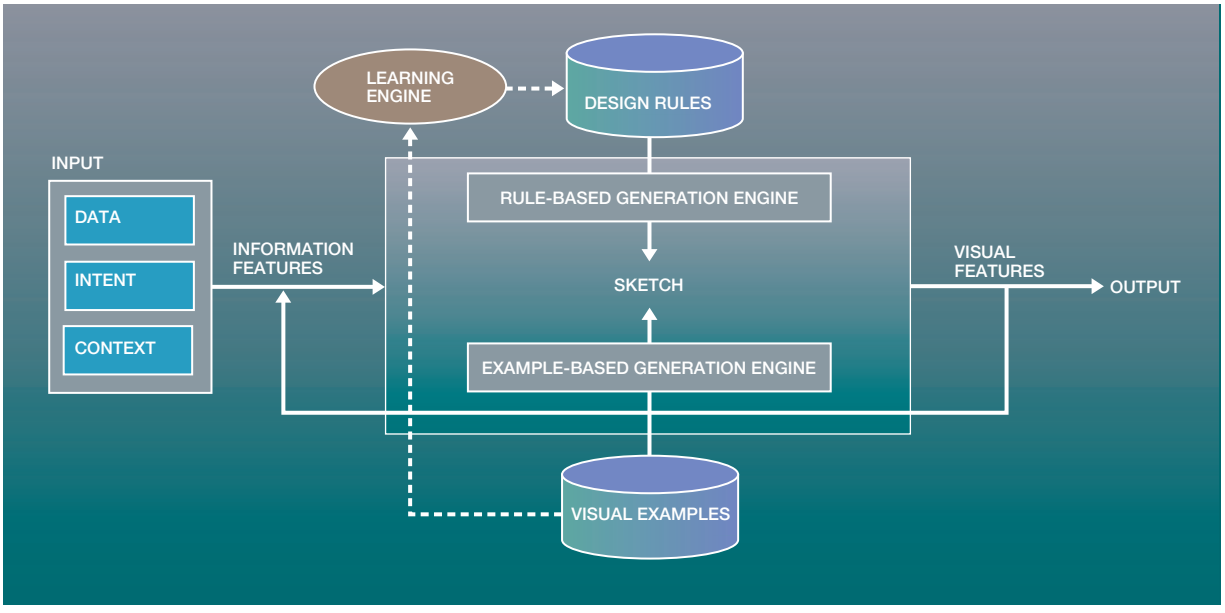
IMPROVISE* overview

Figure 1 shows the high-level components of IMPROVISE*. The initial input is a set of user requests, including the data to be conveyed, presentation intent, and presentation context. The output is an information graphic that conveys the input data. A typical generation process includes three main steps.

Step 1. Starting with a set of inputs, IMPROVISE* always attempts to find the matched examples first, because the examples imply the most specific rules. To search for a set of relevant examples, IMPROVISE* calculates the similarity distances between the user request and the existing graphics. Consequently, it retrieves the top-k matched examples (e.g., $k = 3$). If there are no matched examples (e.g., the similarity difference exceeds a certain threshold), IMPROVISE* will attempt to generate a sketch using a rule-based approach.[15]

Step 2. Using the matched examples, IMPROVISE* first creates a *sketch*, which is an intermediate representation of an information graphic. A sketch outlines the basic visual structure and visual elements without specifying low-level visual details. For example, a sketch of a bar chart specifies the number of axes and bar elements to be created, but not the

Figure 1 High-level components of IMPROVISE*



specifics of the axes and bars, such as their exact scales and positions.

Step 3. Once a sketch is created, IMPROVISE* uses layout and perceptual rules to refine the sketch. These rules usually appear in the form of constraints. As described earlier, we use these constraints to determine the detailed visual characteristics (e.g., the exact scale or position of a graphical element) instead of learning from examples. IMPROVISE* may repeat these steps until a complete information graphic is built.

To support the automated graphics generation process, we apply machine learning to address two subproblems. First, we use decision-tree learning to automatically acquire design rules from existing graphic designs. As shown in Figure 1, IMPROVISE* has an off-line rule learning engine. This engine can extract relevant rules with quantitative confidence factors according to specific learning goals (e.g., learning the usage pattern of a visual technique such as highlight).

Second, we use case-based learning to retrieve and organize similar graphic designs based on user input. To search for relevant examples, we develop a semantics-based, quantitative visual similarity measuring model that supports partial matching. In particular, IMPROVISE* calculates the similarity distances between the user request and the existing graphic examples. As a result, it retrieves the top-k matched examples. To help users navigate through graphic examples in a meaningful way, we also combine our case-based learning method with a hierarchical clustering algorithm. Using this approach, IMPROVISE* can automatically categorize graphic examples based on their subtle semantic and visual properties.

Object-oriented, integrated hierarchical annotation of information graphics

To apply machine learning to graphics generation as just described, we express both input and output uniformly as a collection of features.[16,17]

In traditional machine-learning applications, learning data are simply expressed using a vector of features, which in turn are expressed using flat, categorical values or numbers. However, we have found this simple, flat feature representation inadequate for expressing information graphics. From a computational standpoint, an information graphic expresses a complex mapping between a set of data patterns and visual patterns.[2,4] For example, Figure 2A maps the data correlation (a data pattern) between two data sets, car countries and names, onto

Figure 2 A scatter plot adopted from MacKinlay and its visual composition

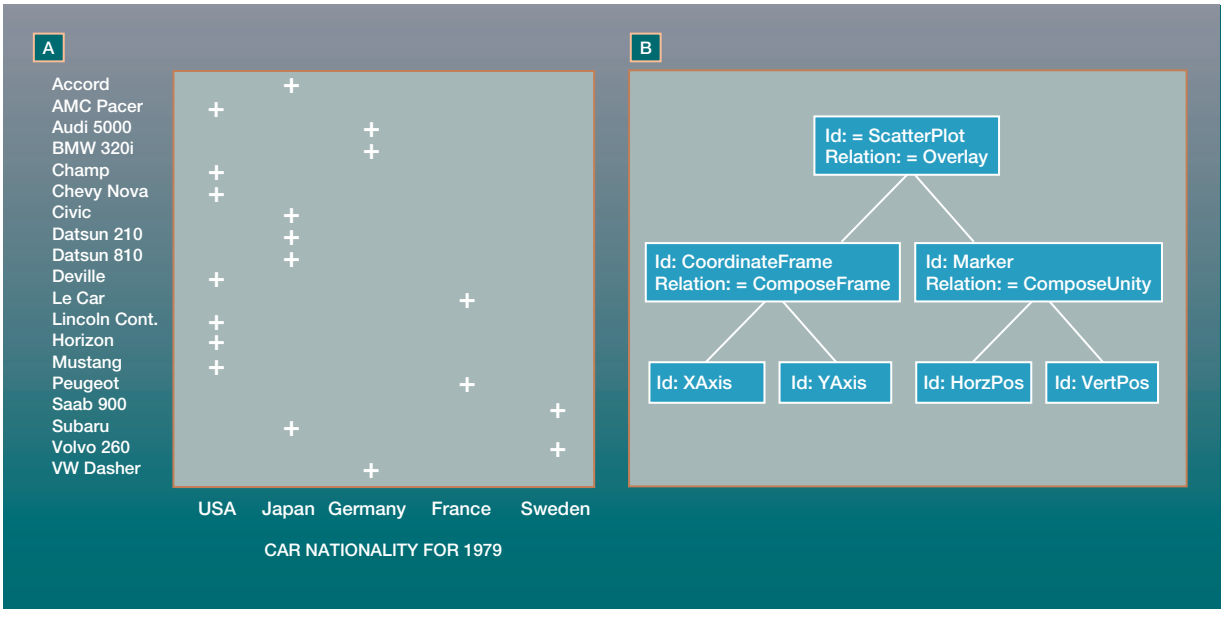


Figure 2A from J. MacKinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Transactions on Graphics*, Vol. 5, No. 2 (1986). Copyright 1986 ACM. Reprinted by permission.

a scatter plot (a visual pattern). To express such a data-to-visual mapping, we must capture the rich characteristics of both patterns. Thus we develop an object-oriented, integrated hierarchical feature representation for describing information graphics.

Visual objects and visual features. A visual depiction may be considered a hierarchical composition of different visual patterns.[2–4] To describe all visual patterns uniformly, we introduce the concept of *visual object*, which is a meaningful visual pattern with a unique syntax and semantics.[4] Moreover, a visual object can be recursively made up of a hierarchical composition of other visual objects.[18] Figure 2B illustrates the visual composition of Figure 2A. In this case, the top-level visual object ScatterPlot is made up of two lower-level visual objects: a coordinate frame (CoordinateFrame) and a set of markers (Marker). A coordinate frame consists of two visual objects (XAxis and YAxis), and a marker contains two other visual objects: a horizontal position (HorzPos) and a vertical position (VertPos). Here we also capture the relationships among visual objects. For example, a scatter plot is composed by overlaying (a visual composition relation) the markers on top of the coordinate frame.

Building on previous work,[16,17] we use seven features (Table 1) to describe a visual object.[19] Four features describe the syntactic aspects of a visual object, such as its visual dimensions (e.g., two or three) and encoding technique (e.g., highlight or cutaway). Two features capture the semantic properties of a visual object (i.e., function and purpose). One feature (structure) expresses a visual object's composition, including its components and the composition relations. The structure feature characterizes an information graphic as a hierarchical composition of visual objects.

It is worth noting that our visual features are different from those low-level visual features (e.g., color histogram) used in content-based image and video retrieval applications (e.g., Aslandogan[20]). In particular, we capture the multilayered visual syntactic, semantic, and structural properties of graphics.

Data objects and data features. Similarly, we use *data objects* and their compositions to represent the data encoded in an information graphic. For example, each marker in Figure 2A encodes a data object, which contains two other data objects: car name and nation.

Table 1 Seven visual features

Feature	Value	Definition
Category (syntactic)	Path	Syntactic category of a visual element, a path in our visual hierarchy
Technique (syntactic)	String	One of the 21 techniques for visualizing data
Parameter (syntactic)	Pair	Parameters of visual techniques, in the form of attribute-value pairs
Dimension (syntactic)	Integer	Dimension of Euclidean space (1, 2, or 3), where the visual element is rendered
Intent (semantic)	Path	The purpose of a visual object, a path in our visual task ontology
Role (semantic)	String	One of the 10 roles that a visual element can play in a visual composition
Structure (composition)	Graph	Represents a visual composition, including the subcomponents and the relations among them

Table 2 Sixteen data features

Feature	Value	Definition
Domain (semantic)	Path	The data application/domain—a path in our ontology
Category (semantic)	Path	The semantic category to which a data element belongs—a path in our ontology
Type (semantic)	String	One of the four meta-data types (entity, relation, ellipses, compound)
Form (meta)	String	One of the three meta-data formations (singleton, array, collection)
Scale (meta)	String	One of the four ways a data element is ordered and measured (nominal, ordinal, interval, ratio)
Unit (meta)	String	The unit of measure used
Continuity (meta)	String	Whether data are continuous or discrete
Resolution (meta)	Integer	The number of distinct values of a data set
Volume (meta)	Integer	The total number of children in a data element
Cardinality (meta)	Integer	The number of elements in a data set
Arity (meta)	Integer	The number of elements in a data relation
Source (meta)	String	Whether the data are originally in the database or are derived from other information
Role (presentation-related)	String	One of ten presentation-related roles that a data element plays in a data composition
Intent (presentation-related)	Path	The presentation goal for visualizing a data element—a path in our user task ontology
Importance (presentation-related)	String	One of four presentation priorities of data: (primary, secondary, extraordinary, background)
Structure (composition)	Graph	Expresses the data composition

We use 16 features to describe a data object comprehensively (Table 2). Three features express the semantic properties of a data object, such as its semantic domain (e.g., business domain) and semantic category (e.g., price). Nine features describe various meta-data properties (e.g., the cardinality of a data set), and two features specify presentation-related data properties (e.g., presentation importance). The last feature, structure, captures the composition of a data object, including the composition relations and components.

Data-visual mapping. Using the concepts just introduced, we can now express an information graphic as a set of visual/data objects, which are expressed by a collection of features. We also bind the corresponding visual and data objects together to explicitly express the data-visual mapping. Specifically, our representation scheme allows a visual object to be linked to its corresponding data object and *vice versa*. It is worth noting that such a binding is not always one-to-one. For example, a visual object (a spherical icon) can be used repeatedly to represent mul-

tiple data objects of the same kind (all towns). On the other hand, a data object (price) can be visualized by multiple visual objects (a textual label or a rectangular bar) in the same graphic. The fact that this binding is not one-to-one complicates the similarity measure between two information graphics, as we discuss later.

Using the features defined in Tables 1 and 2, we have defined an XML (Extensible Markup Language) annotation schema to describe each graphic example. Based on our schema, we have annotated by hand over 3000 visual and data objects for about 100 graphic examples. Each example now in our database has two parts: an XML annotation and the image itself in JPEG (Joint Photographic Experts Group) format. Although we have carefully designed our current feature set to cover a wide variety of graphic examples, by no means is this a complete set of features. Nonetheless, it is easy to extend our current representation framework by adding a new feature or modifying an existing feature.

Automatic graphic-design rule acquisition

In graphics generation, graphic-design rules help map a set of data objects onto a set of visual objects. To acquire these rules automatically, we employ a decision-tree learning technique. Moreover, we identify and formulate a set of specific learning goals that systematically define the types of rules to be learned. We also address how to extract a subset of relevant features from our entire feature set to describe observational data for achieving specific learning goals.

Decision-tree learning. Every design rule can be considered a mapping between a set of rule antecedents (conditions) and consequents (actions). Our goal is to discover the mapping between the n conditions and m actions:

$$G\colon \mathrm{C}_1, \ \ldots\ , \mathrm{C}_i \ldots, \mathrm{C}_n \Rightarrow \mathrm{A}_1, \ldots, \mathrm{A}_j, \ldots, \mathrm{A}_m$$

where C_i denotes the ith condition and A_j is the jth action; G is the mapping function that states the relations among the conditions (in our case, a Boolean function). Moreover, each condition can be considered to be computed based on a feature f.

Thus, our goal of inducing such a mapping G is a special case of supervised learning:[21] $G(f) \Rightarrow t$, where G maps the training samples' feature space f to its target space t. Among various rule-learning algorithms,[21,22] a decision-tree-based approach meets our needs best because of its predictable training time and comprehensive output.[23,24] We thus use this approach to induce a set of classification rules from training samples.

In our application, each sample is an existing graphic; f is a set of design facts and t is a set of design actions. Thus classification rules categorize the mappings between a set of design facts (features) and design actions (targets). For example, one classification rule may look like this:

IF $f[1]$ = VAL
THEN $t[2]$ = T [80.0%]

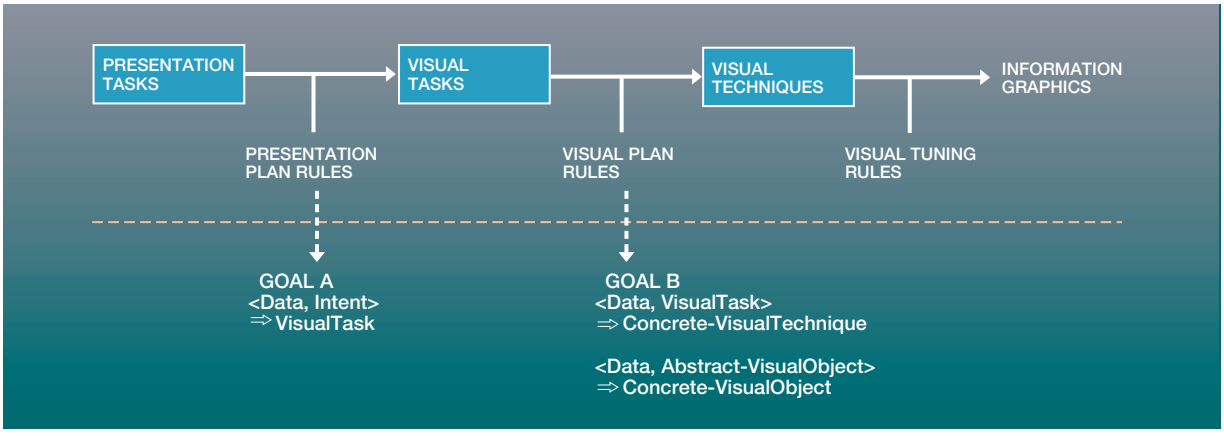
This rule states that if the first feature's value is VAL, we should then use the second design action T. Here 80 percent is a confidence factor (CF), the assessed probability of the occurrence of the mapping. Using the confidence factor, we can solve potential design conflicts. That is, a rule with a higher CF is preferred over a rule with a lower CF, when the conditions of both rules are met.

Learning goals. Specific learning goals help us to identify relevant features and proper feature assignments. Suppose that our goal is to learn how data objects are depicted using different *types* of visual techniques. We would label samples using the category of the techniques (e.g., highlight or zoom), but not the parameters (e.g., the particular color used for highlighting). To formulate learning goals systematically, we follow our design process.

As shown in Figure 3, our process starts with a set of presentation tasks, which specify the general presentation intents and the data to be presented (e.g., summarize school information). Using *presentation plan rules*, IMPROVISE* maps the presentation tasks onto a set of visual tasks (e.g., emphasize school cost). These visual tasks describe *what* desired visual effects are to be achieved.[4] *Visual plan rules* then refine the visual tasks to a set of concrete visual techniques, such as highlight and enlarge. These visual techniques specify *how* to achieve the desired effects. To complete a graphic design, *visual tuning rules* help define the parameters of a visual technique (e.g., the color of a highlight technique).

Since visual tuning rules determine visual details, which could vary a great deal (e.g., the exact scales of visual objects), it is difficult to generalize these rules from examples. Thus we focus on learning only

Figure 3 Learning goals formulated systematically, based on a design process



presentation plan rules and visual plan rules. Accordingly, we formulate two sets of learning goals. The first set of goals (Goal A in Figure 3) is to establish presentation plan rules, which map presentation data and intents onto visual tasks. For example, a particular learning goal may be to learn how to map a "summarize" intent to different visual tasks (e.g., cluster or emphasize).

Our second set of goals (Goal B) is to learn visual plan rules, which capture the correlations among the data, visual techniques, and visual objects. Since visual planning includes refining both visual techniques and visual objects,[15] we have two subsets of learning goals. One is to learn the relations between abstract and concrete visual techniques so that abstract techniques can be refined (visual tasks are considered to be abstract techniques). For example, to refine the abstract technique "emphasize," we need to learn how it is related to concrete visual techniques, such as highlight and enlarge. The other subset is to learn the relations between a higher-level visual object and its components (e.g., the semantic relation between a bar chart and its header). The second subset of learning goals helps us to acquire visual object construction rules (e.g., how to compose a bar chart).

Learning from features with complex structures. In traditional learning applications, features are usually described by a flat structure of numerical or nominal values. However, many of our features are specified using a complex structure. For example, our visual feature "structure" describes the composition of a visual object using a complex expression. Moreover, simply flattening the complex features may produce an overwhelmingly large number of possible feature assignments due to the complexity of our feature domains. For example, flattening the visual feature "category" would produce a whole set of values that describe a visual object at different levels of abstraction.

To systematically select relevant features and feature assignments, currently we use a trial-and-test approach. Initially, we select all relevant features based on a learning goal. We then construct training samples by using the most general values of each feature. Because of the general description, observational data may not be well distinguished and the learned rules may appear trivial. In this case, we gradually refine each feature, starting with those that have the fewest possible values (the lowest information gain, according to Mitchell[21]). This trial-and-test approach is not optimal, but it does help us to monitor the learning process, while preventing us from missing out on useful features or feature assignments.

Experiment: Learning visual technique correlation. Using a decision-tree approach, here we demonstrate how to learn rules for refining a visual technique.

Learning goal. One of the advantages of automated graphics generation systems is that users can specify their information-seeking goals at a high level without worrying about low-level visual details.[3,4] Upon receiving an abstract user goal, IMPROVISE* must determine what specific visual techniques should be employed to achieve the goal.[4] Specifically, in our ex-

Figure 4 Data samples used in learning

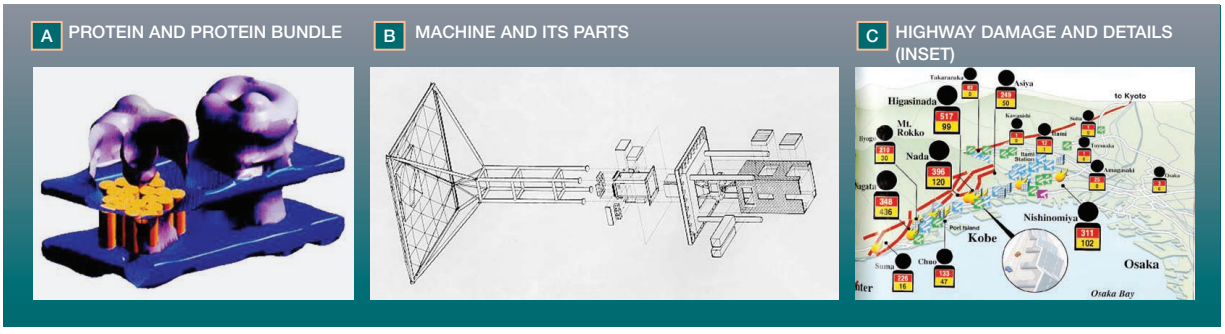


Figure 4A from P. Keller and M. Keller, *Visual Cues: Practical Data Visualization,* IEEE Press, Piscataway, NJ (1993). Copyright 1993 IEEE. Reprinted with permission. Figure 4B from *Information Graphics,* Thames & Hudson, London (1998). Copyright 1998 Foster and Partners, London. Reprinted with permission. Figure 4C from *Information Graphics,* Thames & Hudson, London (1998). Copyright 1998 TUBE Graphics, Tokyo. Reprinted with permission.

Table 3 Possible feature values for relation

Relation Value	Value Definition
Constituent	o_2 is a part of o_1
Attributive	o_2 is an attribute of o_1
Enumeration	o_2 enumerates o_1
—Specify	o_2 is a specification (e.g. detail) of o_1
—Instance	o_2 is an instance of o_1
—Subconcept	o_2 is a subconcept (subclass) of o_1

periment we want to learn the correlations between a visual goal, reveal, and three specific visual techniques, expose, separate, and overlay. Here visual goal reveal $\langle v_1, v_2, o_1, o_2 \rangle$ aims to display a visual object v_2 (encoding data object o_2) that is usually "hidden by" visual object v_1 (encoding data object o_1). The three techniques describe the specific ways of bringing out the hidden object v_2. Expose uses standard graphic techniques, such as cutaway, open, or set transparency, to expose the hidden object (Figure 4A). Separate reveals the hidden object by splitting and slicing v_1 (Figure 4B). Unlike expose and separate, which modify v_1, overlay places v_2 directly on top of the existing display (Figure 4C). Thus, our learning goal can be expressed as:

$$G: f \Rightarrow \{\text{Expose, Separate, Overlay}\}$$

where G is a Boolean function, and f is a set of features describing v_1, v_2, o_1, and o_2.

Training data and tool. Our training data are collected from three books.[25–27] Because of the diversity of the samples, we hope to avoid biased conclusions that might be meant only for a specific application. Among more than 500 illustrations, 60 are related to visual revealing.

Based on our learning goal, we first determine our learning input. We initially extract five features: relation, DCat_1, DCat_2, VCat_1, and VCat_2. The relation feature is extracted from the structure data feature to describe the semantic relation between data objects o_1 and o_2. DCat_1 and DCat_2 are the semantic categories of o_1 and o_2. VCat_1 and VCat_2 are the visual categories of v_1 and v_2. During our feature selection, we have eliminated indistinguishable and irrelevant features. For example, the intention feature for o_2 is always a path: Lookup $\Rightarrow$ Individual $\Rightarrow$ Detail, and the cardinality feature is irrelevant to our goal. We define our target to be one of the three reveal techniques: expose, separate, and overlay.

We can describe all five features at multiple levels of abstraction. For example, Table 3 lists three top-level values of feature relation and three lower-level values of enumeration. Assigning top-level values to all five features, we obtain the most general feature assignment. Table 4 shows the three samples in Figure 4, using this assignment. Next we discuss our experimental results using Quinlan's decision-tree learning algorithm, C4.5.[24]

Experimental results. We train the C4.5 algorithm on the 60 selected illustrations, using sixfold cross validation, a standard procedure when the amount of

Table 4 Values assigned for features in Figure 4

	Relation	$DCat_1$	$DCat_2$	$VCat_1$	$VCat_2$	Target
4A	Constituent	Tangible	Tangible	VisUnity	VisUnity	Expose
4B	Constituent	Tangible	Tangible	VisUnity	VisUnity	Separate
4C	Enumeration	Intangible	Intangible	VisUnity	VisUnity	Overlay

Table 5 The design rules for reveal, and the salient feature

$VCat_1$ Values and Definitions	Rules (Overall Error Rate 21.7 Percent)
Icon: An iconic representation	(A) $VCat_1$ = Icon $\Rightarrow$ Overlay (85.7 percent)
Label: A textual representation and its variations (e.g., buttons)	(B) $VCat_1$ = Label $\Rightarrow$ Overlay (82.0 percent)
Symbol: A three-dimensional representation	(C) $VCat_1$ = Symbol $\Rightarrow$ Expose (60.0 percent)
VisStructure: A visual schematic drawing (e.g., map)	

data is limited. Using the most general feature assignment, we learn five classification rules with an overall classification error of 30 percent:

1. $DCat_1$ = Intangible $\wedge$ $VCat_2$ = VisStructure $\Rightarrow$ Overlay [87.1%]
2. Relation = Enumeration $\wedge$ $VCat_2$ = VisStructure $\Rightarrow$ Overlay [84.1%]
3. Relation = Enumeration $\wedge$ $DCat_1$ = Intangible $\Rightarrow$ Overlay [84.1%]
4. $DCat_1$ = Tangible $\wedge$ $VCat_2$ = VisUnity $\Rightarrow$ Expose [56.7%]
5. Relation = Constituent $\wedge$ $DCat_1$ = Intangible $\wedge$ $VCat_2$ = VisUnity $\Rightarrow$ Separate [50.0%]

Although these rules indicate a strong correlation between the visual object v_2 ($VCat_2$) and reveal techniques, they may not be useful in predicating a reveal technique. This is because $VCat_2$ might be unknown when the technique needs to be decided. After dropping $VCat_2$, we get four new rules, but with an overall error of 35.7 percent:

6. Relation = Enumeration $\Rightarrow$ Overlay [74.0%]
7. $DCat_1$ = Intangible $\Rightarrow$ Overlay [70.2%]
8. $DCat_1$ = Tangible $\wedge$ $DCat_2$ = Intangible $\Rightarrow$ Expose [58.7%]
9. Relation = Constituent $\wedge$ $DCat_1$ = Tangible $\Rightarrow$ Expose [48.8%]

To reduce the classification error, we systematically vary feature assignments from the most general to more specific. Consequently, an assignment of feature $VCat_1$ produces a set of useful rules, which predict reveal techniques based on v_1 with a relatively low error rate (Table 5).

Compared to the handcrafted rules used in IMPROVISE,[15] the learned rules give similar design suggestions. But the learned rules are more concise (with fewer, but adequate conditions), and have a quantitative confidence factor, which is missing in our handcrafted rules.

Case-based graphic example retrieval

This section addresses our second application of machine learning in automated graphics generation. A new graphic design can be acquired by transforming existing graphic examples that address situations bearing a strong similarity to the new situation. Assume that an existing graphic illustrates city information on a map, including the city location and population. We wish to transform this graphic to display school information on a map, including the school geographical boundary and student data.

Because this process requires that analogous cases be retrieved from memory, transformed, and applied to the new situation, this type of learning is known as *case-based learning*.[21] In this example, the analogy is drawn upon a set of parameters, such as the presentation goal (locate objects on a map) and the inherent relations among the data elements involved (both city and school are entities, and other data are their attributes). To determine relevant parameters that help us to draw a meaningful analogy, we use

our feature representation to express the properties of cases (existing graphics and user requests).

To draw an analogy, a critical step is to develop a similarity measuring model that can precisely assess how well a stored case (graphic example) matches a new case (e.g., a user request). Before presenting our similarity measuring model, we first describe our internal representation of a case (information graphics).

Picture objects and picture graphs. As described earlier, we annotate each information graphic using three parts: a hierarchical composition of visual objects, a hierarchical composition of data objects, and the mappings between these two compositions. To allow an accurate visual similarity measuring, we introduce *picture objects*, which merge the two hierarchies together while preserving all their features.

Specifically, each picture object encapsulates two components: a visual object and a corresponding data object. Moreover, picture objects must be linked together to reflect the structures of their visual objects and data objects. However, linking picture objects together is not straightforward, because the data-visual mapping is not always one-to-one. This implies that data objects can be connected in one way and visual objects in another. Thus, we *cannot* simply link picture objects together by following only the data composition or only the visual composition structure.

Figure 5A shows two hierarchical structures: a data composition (left) and a visual composition (right). In the data composition, data object D0 is composed of four data objects, which in turn are made up of other data objects. Similarly, in the visual composition, visual object V0 consists of three visual objects, which may or may not be made up of other visual objects. We use dotted, double-ended arrows to indicate the data-visual mapping between corresponding data and visual objects. For example, there is a mapping between D0 and V0, and between D1 and V1. Note that both data objects D1 and D2 are mapped to the same visual object V1, and D41 is mapped onto two visual objects, V31 and V32. Data object D3 (not directly visualized) and visual object V3 (not encoding any data content) have no corresponding mappings.

To capture both data and visual compositions, we organize picture objects using two graph structures. First we construct a picture-visual graph (PVGraph), which organizes picture objects based on the relations among the visual objects. Figure 5B depicts a PVGraph, created based on the visual structure in Figure 5A. Here, each picture object contains two parts; for example, P0 is a picture object, containing a visual object V0 and a data object D0. The special symbol "/" represents a null object. Because a visual object (e.g., V1) may be mapped to multiple data objects (D1 and D2) and *vice versa*, a picture object may contain more than one visual or data object (e.g., picture object P1 in Figure 5B). Likewise, we can construct a picture-data graph (PDGraph), which links picture objects together based on data relations. Figure 5C shows a PDGraph, built on the data compositional structure in Figure 5A. Note the differences between the generated PVGraph (Figure 5B) and PDGraph (Figure 5C) when the data-visual mapping is not one-to-one (Figure 5A).

Using the PVGraph and PDGraph together, we are now able to obtain an accurate comparison between two graphics by comparing their visual objects, data objects, and the mappings between the visual and data objects. This is different from previous systems, which usually separate the data comparison from the visual comparison (e.g., Chuah,[5] Derthick[6]).

Similarity measure. When comparing two annotated graphics, IMPROVISE* first creates a PVGraph and a PDGraph for each graphic (currently in the form of an XML document). The similarity between the two graphics is then measured by calculating the distances between the four corresponding graphs:

$$D(E_1, E_2) = w_v \times D(PV_1, PV_2) + w_d \times D(PD_1, PD_2)$$

Here E_1 and E_2 are two graphic examples. PV_1 and PV_2 are their PVGraphs, and PD_1 and PD_2 are their PDGraphs. Weights w_v and w_d determine the contributions of the visual and data parts, respectively. Similarly, in the rest of the section we use different weights to indicate the contributions of different features or feature sets in our distance measure. Currently, since all of our distances are normalized to lie between 0.0 and 1.0, the summation of all weights at the same level is always 1.0. As described later, users can adjust these weights to express what features and feature sets they are interested in. For example, if a user is interested only in comparing the graphic examples by their visual appearance, w_d would be 0.0, and w_v would be 1.0. We have prepared a default weight file, which weighs all parts equally, for specifying the weights if users do not have specific interests.

Since both PVGraphs and PDGraphs are ordered graphs with roots (starting nodes), the distance between two graphs is the distance between the two corresponding roots, which are hierarchical compositions of other picture objects. Here we use *composite* and *atomic* picture objects to distinguish picture objects with children from those without children. Thus, a composite picture object has four parts: a visual object,[28] a data object, a set of children, and the composition relation among its children.[29] Conversely, an atomic picture object has only two parts: a visual object and a data object.

Each root is a composite picture object. Thus, the distance between two roots is defined by four parts:

1. Distance between their visual objects
2. Distance between their data objects
3. Overall distance between their children
4. Distance between the composition relations

Here (1) and (2) can be measured by computing the distance between a set of visual/data features,[30] and (3) can be recursively defined by the distance between two picture objects (see below). Finally (4) requires only a simple feature comparison (see the "Distance between features" subsection). Next, we define the distance between two picture objects.

Distance between picture objects. We define the distance between two picture objects P and P' using three formulas:

1. If both P and P' are atomic picture objects:

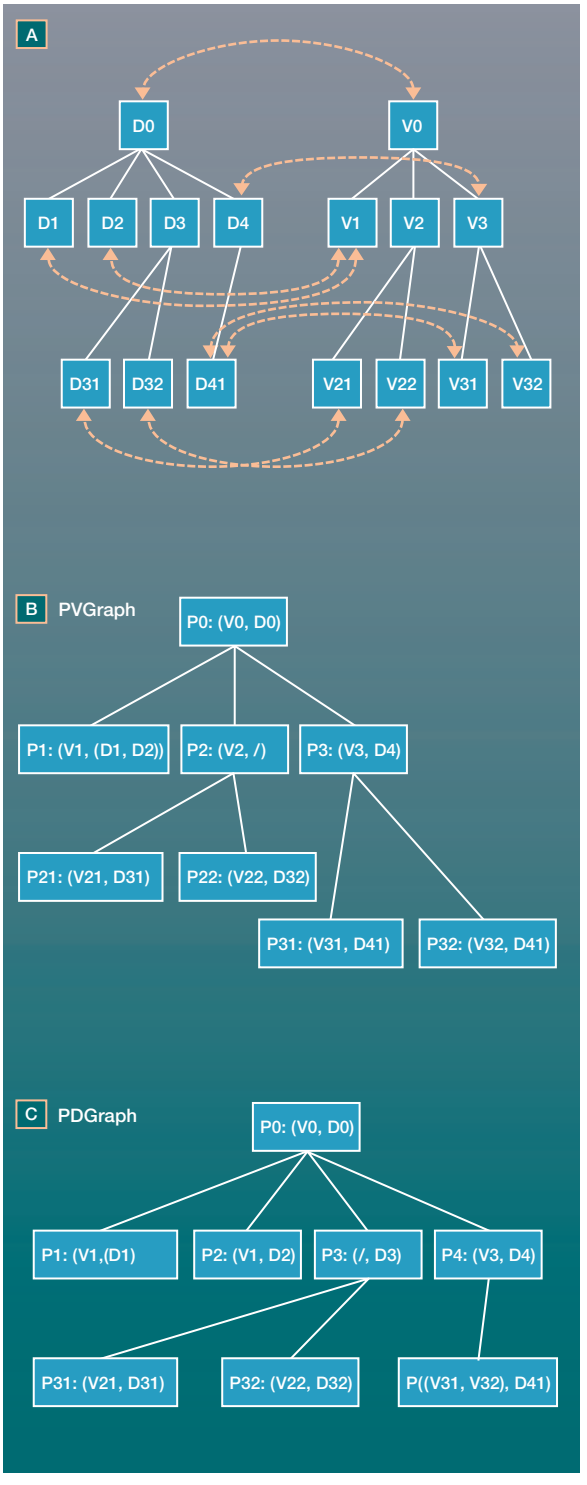
$$D(P, P') = w_{v1} \times D(VO, VO') + w_{d1} \times D(DO, DO') = D_{vd}(P, P')$$

   Here VO and VO' are the visual objects in P and P', respectively; and DO and DO' are the corresponding data objects. This formula states that the distance between two atomic picture objects is the weighted sum of the distances between their corresponding visual objects and data objects. This case is the "stop" condition in a recursive process for computing the distance between two picture objects. We record it using $D_{vd}(\,)$ for later use.

2. If P is atomic and P' is composite:

$$D(P, P') = w_{vd} \times D_{vd}(P, P') + w_s \times \text{avg}[D(P, P'_j)]_{1 \le j \le Q}$$

Figure 5 A visual-data mapping and the corresponding PVGraph and PDGraph



Here Q is the total number of children of P' and P'_j is the jth child. The distance between an atomic and a composite object is a weighted sum of two distances: (1) the distance between their visual/data objects, and (2) the average distance between the atomic object and the children of the composite object. The combined distance ensures that we compare all visual/data features, including their structures.

3. If P and P' both are composite:

$$D(P, P') = w_{vd} \times D_{vd}(P, P') + w_s \times D_s(P_s, P'_s)$$

where

$$D_s(P_s, P'_s) = w_c \times D_c(P_c, P'_c) + w_r \times D(r, r')$$

Here P_s and P'_s are the visual/data composition structures, including their children P_c and P'_c and the composition relations r and r' among the children. This formula indicates that the distance between two composites is a weighted sum of two distances: (1) the distance between their visual/data objects ($D_{vd}(\,)$), and (2) the distance between their structures ($D_s(\,)$). The distance between their structures is computed by the distance between their children ($D_c(\,)$), and the distance between the relations.

Now we define the overall distance between two sets of children $D_c(\,)$. The complication here is that P and P' may have different numbers of children. To make a systematic comparison, we first insert "dummy" objects to make up the difference. In addition, we define that the distance between a regular picture object and a dummy is always 1.0 (the largest possible distance in our model). We then seek a mapping between the two sets of children so that their overall distance is minimized. The rationale is that a similarity measure should stress the *similar* properties (e.g., matched components) but not the differences. Moreover, function min( ) guarantees that the distance is zero between two graphics that are exactly the same. We formulate the distance between two sets of children P_c and P'_c as follows:

$$D_c(P_c, P'_c) = \min_{\sigma} \left(\sum_{i=0}^{K} D(P_i, P'_{\sigma_i}) \right), \text{ where } K = \max(M, N)$$

Here M and N are the total number of elements in P_c and P'_c, respectively. σ is a permutation of P'_c, P_i is the ith element of P_c, and P'_{σ_i} is the ith element of the permutation σ. Theoretically, finding such a mapping requires an optimization algorithm, which is normally NP-complete. Currently we use a greedy algorithm to find the mapping. However, we have a special case if both relations are similar and noncommutative (e.g., the distance between the relations is below a certain threshold). In this case, the mapping between the two sets of children is determined by the ordering of the children. For example, when comparing Overlay(a, b) with Overlay(a', b'), we only map a to a' and b to b', since Overlay is noncommutative.

Distance between visual/data objects. As just described, computing the distance between two picture objects requires calculating the distance between visual/data objects. Because both types of objects are described by a set of features (Figure 6), we can define the distances between their relevant features.

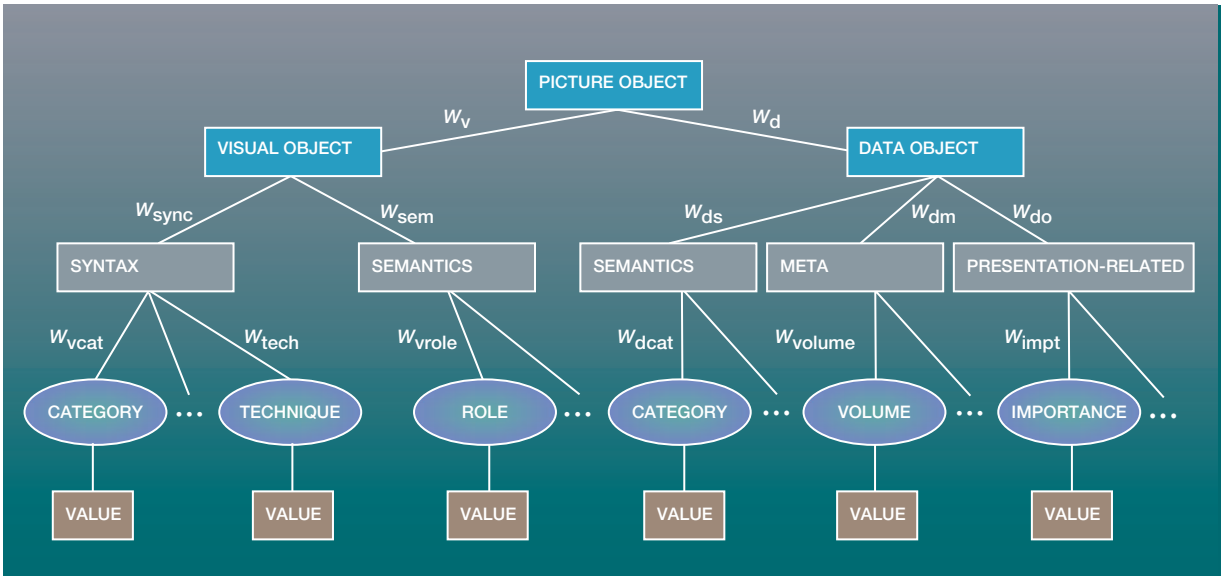
Features can be grouped into feature sets. Figure 6 shows two visual feature sets and three data feature sets. Each feature set or feature is assigned a weight to promote or demote its contribution in the overall distance measure. Figure 6 indicates that the distance between two visual/data objects is a weighted sum of distances between all feature sets. Similarly, the distance between two feature sets is a weighted sum of distances between all features in the set. Thus we can express the distance between two visual/data objects O and O':

$$D(O, O') = \sum_{i=1}^{N} \sum_{j=1}^{K_i} w_{ij} \times D(F_{ij}, F'_{ij})$$

Here N is the total number of feature sets and K_i is the total number of features in the ith feature set. F_{ij} and F'_{ij} are the jth features in the ith feature set of O and O', respectively, and w_{ij} is the weight of the jth feature in the ith feature set (e.g., w_{vcat} is the weight of the category feature in the visual syntactic feature set shown in Figure 6).

Distance between features. We compute the distance between two features F and F' by comparing their feature values V and V'. If both values are simple categorical values, the process is trivial (e.g., using strcmp( ) to compare strings). However, we have three special cases, because some of our features

Figure 6 A typical picture object and its composition



carry rich semantics and their distances are not simple Euclidean distances.

The first case compares feature values that are paths (e.g., the value of the visual/data category feature). In this case, we define the distance between two paths V and V' as:

$$D(V, V') = \sum_{i=1}^{\max(M,N)} w_i \times D(v_i, v_i')$$

where

$$V = \{v_1, \ldots, v_N\} \text{ and } V' = \{v_1', \ldots, v_M'\}$$

Note that if $N \neq M$, the shorter path would be padded with "/" values. The distance between "/" and any string is always the longest possible distance (1.0). The weights in this formula gradually decrease ($w_i > w_j$ if $i < j$) to ensure that a higher-level match is preferred to a lower-level match.

The second case compares feature values that are strings, but have special connotations, for example, two visual relations ComposeFrame and ComposeUnity. We have prepared several relation distance matrices to specify the distance between any two relations. For example, the distance between HorzAlign and VertAlign is 0.5, whereas the distance between HorzAlign and Overlay is 1.0.

The third case compares numeric feature values for which the absolute numeric distance is inadequate to return a meaningful result. For example, we may consider that data volume is relatively *small* if the value is less than 100. Thus, a graphic that has a data volume of 100 is not very different from one with a data volume of 30. To deal with this situation, we use the following distance measure between two numeric values v and v':

$$D(v, v') = \text{avg}(D(\text{scale}(v), \text{scale}(v')), \text{normalize}(|v - v'|))$$

Here function scale( ) maps an actual number onto an ordinal scale. By default, we use four ordinal scales: small (0–100 data objects), medium (100–1000), large (1000–10000), and huge (>10000). Using this setting, the scaled distance between values 60 and 180 is the distance between the two ordinal scales small and medium.

To handle boundary values (e.g., 100 and 101 are mapped onto two ordinal scales, and 1000 and 9999 are mapped to the same scale), we also compute a normalized absolute distance between v and v'. Currently, we use a normalization function to map an absolute distance onto a value between [0, 1].

Dynamic clustering. The most common use of case-based learning is to classify a new case (e.g., a newly generated graphic or a new sample provided by others) based on a set of similar existing cases.[21] In addition, a user may want to browse the example database to understand the general structure and patterns of existing presentation examples. Along with our similarity model, we use a hierarchical clustering algorithm[31] to organize existing graphic examples and classify new information graphics. The clustering algorithm ensures that the intracluster similarity is maximized and the intercluster similarity is minimized.[31] Instead of using a static classification method (e.g., Chuah[5]), IMPROVISE* creates clusters dynamically, based on user requests. In particular, users can dynamically adjust various weight assignments to view different organizations of graphic examples by interests (e.g., organizing all examples by visual appearance). Users may also ask that a new sample be classified (e.g., find all existing examples similar to the new one).

There are two reasons for supporting dynamic clustering for an automated graphics generation system such as IMPROVISE*. First, new graphics are generated dynamically and need to be stored back in the visual database in a useful way (e.g., for efficient future retrieval). Second, as we continue collecting examples from other sources (e.g., professional designers or other generation systems) to enrich our database, we need an evaluation tool to tell us how the new samples are related to our existing cases. As the number of examples grows, our dynamic clustering also provides a level of organization by which users can easily identify the overall trend among all examples, and the local patterns within a subset.

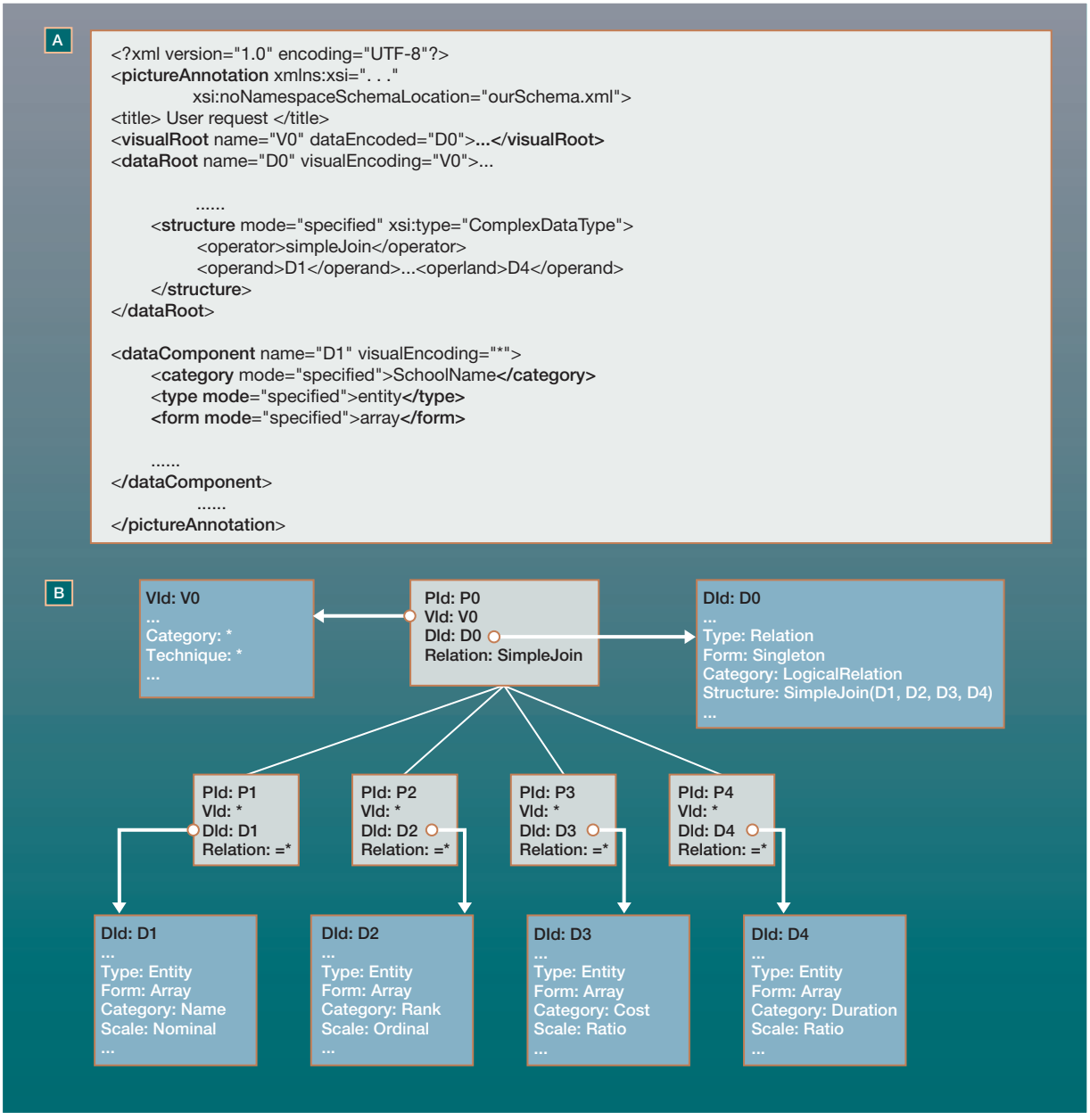
Partial matching. In our approach, we allow a user to specify partial requests. This means that the user does not need to know every aspect of the data and desired visual characteristics. Suppose that a user wants to discover the correlation among a set of school data: school names, academic ranks, the cost of education per year, and the average time spent on completing a degree. Our users can easily enter such a request using a form-based graphical user interface provided by the XML Spy Integrated Development Environment[32] based on our defined schema. Figure 7A shows a fragment of this request created by XML Spy. According to our schema, each request may include (1) a top-level visual/data object (visualRoot/dataRoot) and (2) a set of lower-level visual and data objects (visualComponent/dataComponent). Each visual/data object (including its root) is specified in terms of the features defined earlier. For example, data object D1 is described as an array (form) of school names (category). In this case, since little visual preference is provided (no visual component is specified), IMPROVISE* automatically creates only a PDGraph (Figure 7B) based on this input. In the generated PDGraph, each picture object (e.g., P0) contains a data object identifier (DId), which points to the actual data object, and an unspecified visual object. The symbol "*" represents all unknown objects or feature values.

In addition to specifying a partial request, a user may provide certain matching criteria. IMPROVISE* allows two types of matching criteria: soft match and hard match. A *soft-match* criterion does not guarantee an exact match, whereas a *hard-match* criterion requires an exact match. By default, all user specifications are soft-match criteria. Users can update the criteria by adjusting various weights defined in our model. Conversely, users must explicitly specify a hard-match criterion for a specific feature of a visual/data object (see Experiment 2, following).

Experiments. In this section, we illustrate the application of our case-based learning through two concrete experiments. In the first, we demonstrate that we can distinguish different graphic examples at a fine granularity. In the second, we show how IMPROVISE* handles a partially specified user request.

Experiment 1. Suppose that a user wants to browse all graphic examples without a specific emphasis on any of the features. This request is translated into a neutral weight assignment, in which all independent features are weighted evenly. But if a feature (e.g., scale) dominates another (e.g., measurement), the two are weighted differently. Using these weights, IMPROVISE* produces a hierarchical cluster for 20 visual examples, which are collected from different sources.[2–5,12,33] Figure 8A shows all the clusters, which are hierarchically formed by recursively merging two clusters together until all examples are under one cluster. Here the *x*-axis represents the example identifier, and the *y*-axis represents the distance between two clusters. By setting different distance thresholds, the user can have a different number of clusters. For example, if the threshold is set to 0.25, there will be six clusters (the number of dots in Figure 8A). Continuing this experiment, we modify the feature weights to stress only the visual features. This new weight assignment produces a new clustering result (Figure 8B).

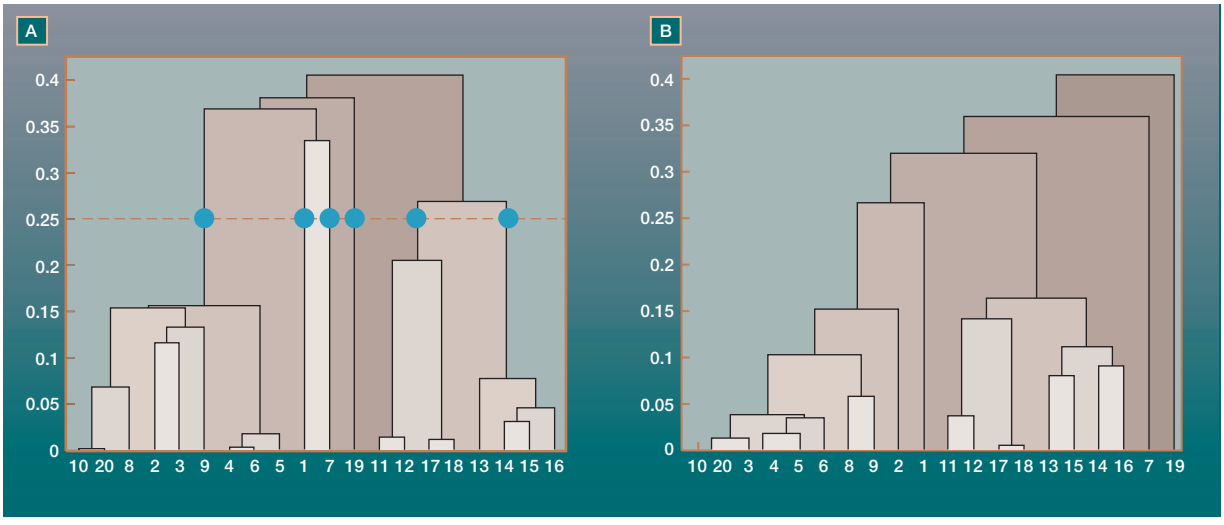
Figure 7 The representation of a partial user request



Experiment 2. In most cases, users submit a partial request that describes the data to be conveyed, with few visual preferences. Figure 7 illustrates such a request. Taking this request as input, IMPROVISE* returns the top-three matched examples: 8 and 9 (see Figure 9) and 5 (see Figure 10). Here example 5 is a bar chart that encodes seven cars with their name, price, and nation (represented by color).

Not surprisingly, all data objects expressed in the request can be mapped to at least one of the data objects encoded in the three examples. The cost in the

Figure 8 Clusters produced by different weights



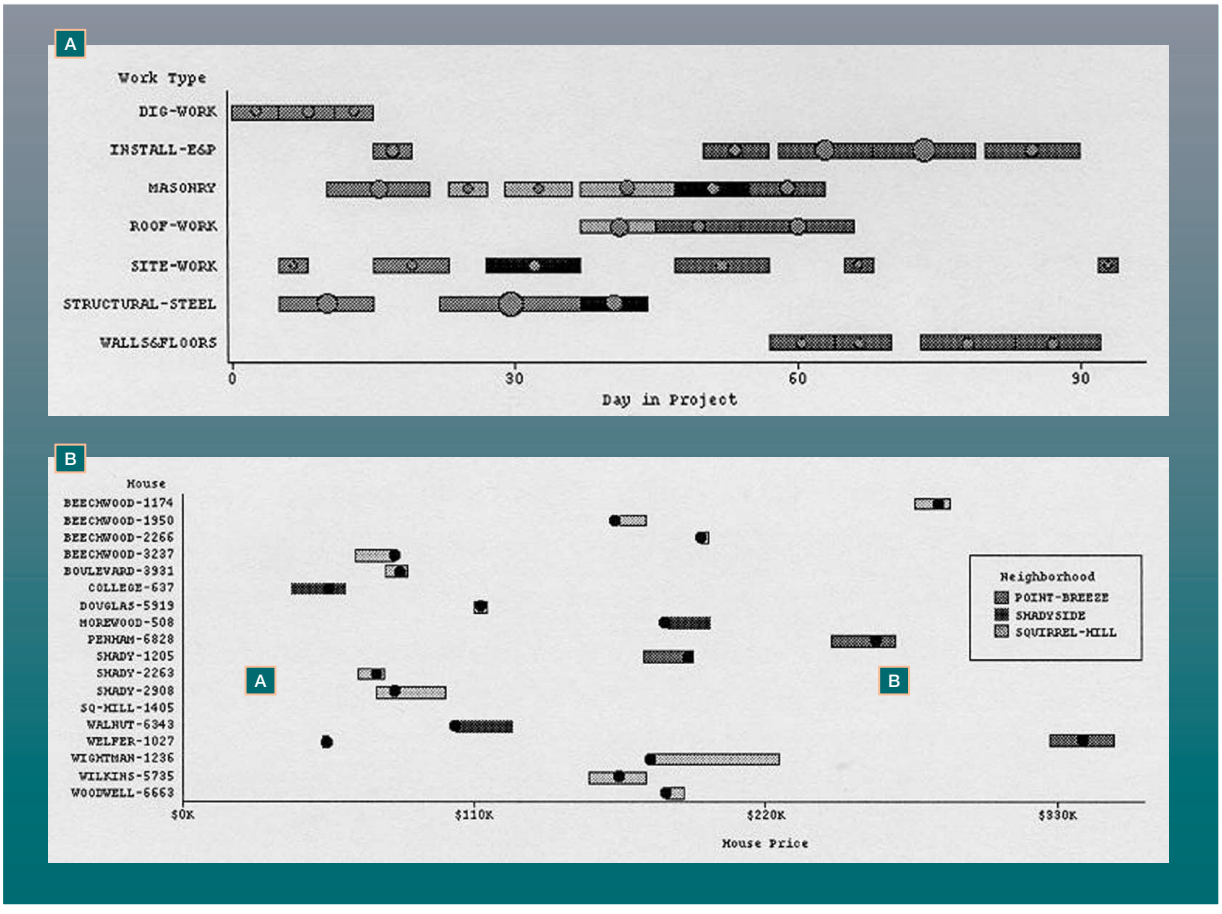
request can be mapped to the cost in example 8, and to the price in examples 9 and 5. Note that examples 8 and 9 encode *more* data dimensions (seven) and example 5 encodes *fewer* data dimensions (three) than the user requested (four). Thus IMPROVISE* performs subset matching against examples 8 and 9,[5] and superset matching against example 5 (i.e., one element of the data in the request cannot be mapped to any element in the example). In example 5, we need to find other means to encode the data left unmatched. Note that our current algorithm does not allow overload matching (e.g., mapping both the time spent in school and the cost in the request onto the price in example 5). This is because we cannot use the same visual cue to encode two different types of data.[34]

In addition to providing data descriptions, an experienced user may supply specific visual preferences. Continuing our experiment, suppose that our user specifies that the category of V0 in Figure 7 *must be* a bar chart. This hard-match criterion requires that all retrieved examples be bar charts. Consequently, IMPROVISE* retrieves example 5 (Figure 10) as the top match. Here example 5 replaces the top matches in the previous experiment, examples 8 and 9, which are not bar charts.

It is interesting to note the change of relationships between examples 8 and 9, which are both created by SAGE (Figure 9). When considering all features equally, IMPROVISE* finds that examples 8 and 9 are not that similar in Experiment 1. But it finds them to be quite close in Experiment 2, when considering only visual features. After examining the examples, we are able to produce an explanation. First, examples 8 and 9 are different in their data composition characteristics although they appear very similar. Particularly, the vertical axis in example 8 encodes the work type, which is not a unique identifier for the projects (a one-to-many mapping exists between the work types and projects). In contrast, the vertical axis in example 9 encodes the house name, which can uniquely identify a house (a one-to-one mapping exists between the names and the houses). As indicated in Chuah,[5] examples 8 and 9 also differ in their data-encoding format. The circles (encoding the project cost) in example 8 have no relationship to the bars along the horizontal axis (time duration), whereas the circles (agent-estimated price) and bars (asking and selling prices) in example 9 are related. The subtle data differences widen the distance between examples 8 and 9 in Experiment 1, but their strong visual structural similarity brings them close in Experiment 2.

Although we describe the dynamic clustering and partial matching in separate experiments, in reality they work together to provide the foundation of our example-based graphics generation. Specifically, IMPROVISE* presents users with the top-matched examples that are marked on a clustering hierarchy.

Figure 9 Examples 8 (top) and 9 (bottom), from Chuah, Roth, and Kerpedjiev



From M. Chuah, S. Roth, and S. Kerpedjiev, "Sketching, Searching, and Customizing Visualizations," *Intelligent Multimedia Information Retrieval*, M. Maybury, Editor, American Association for Artificial Intelligence, Menlo Park, CA (1997), p. 97. Copyright 1997 American Association for Artificial Intelligence. Reprinted by permission.

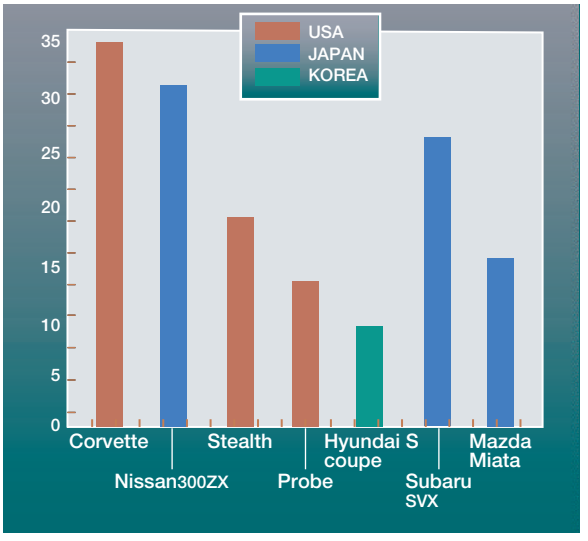
Users can then review these examples and observe the overall relations among them to facilitate further example browsing and refinement.

Conclusions and future work

Automated graphics generation promises to simplify information-system developers' tasks by automatically creating visual depictions that can effectively communicate the desired information to users. Previous research efforts focus mainly on using rule-based generation approaches, which employ handcrafted design rules. To overcome the shortcomings of rule-based approaches, we have developed a hybrid approach that combines rule-based with example-based generation approaches. Currently, our approach is embodied in the prototype system IMPROVISE*. Under this hybrid framework, our current work is on how to apply various machine-learning techniques to facilitate graphics generation. In particular, in this paper we focus on addressing the use of machine learning in graphics generation from three aspects.

First, we introduce an object-oriented, integrated hierarchical feature representation that we use to annotate existing information graphics and user requests. Not only do we capture various characteristics of data and visual components; we also stress the data-visual bindings. Second, we describe how to use

Figure 10 Example adopted from Myers, Goldstein, and Goldberg



From B. Myers, J. Goldstein, and M. Goldberg, "Creating Charts by Demonstration," *Proceedings, ACM Conference on Human Factors in Computing Systems*, Boston, MA (April 24–28, 1994). Copyright 1994 ACM. Reprinted with permission.

decision-tree learning to automatically extract design rules from a set of annotated graphic examples. Our experimental results have demonstrated that we can learn concise and useful design rules. Third, we combine case-based learning with dynamic clustering to retrieve matched graphic examples for reuse based on user requests. Specifically, we use a semantics-based, quantitative visual similarity measuring algorithm to retrieve the top-k matched examples. To demonstrate our approach, we have conducted a series of experiments and obtained satisfactory results.

There are several things on our agenda to improve our current use of machine learning in automated graphics generation. To facilitate fine-grained graphic design analysis using machine learning, we are collecting a large graphics corpus, which will contain several thousand design patterns. In addition, we are developing rule-acquisition learning algorithms that can take advantage of our complex structured features; existing tools focus only on numerical or nominal features. Dealing with a potentially large set of features in graphics synthesis, we would like to explore how to automatically extract salient features.

Our current similarity measure emphasizes the overall similarity between two cases. We are improving the current model to calculate the similarity between meaningful design fragments. In graphics generation, it is crucial to retrieve existing matched design fragments. This is because a new graphic design may need to be pieced together by using multiple design fragments from different existing graphic examples.

To facilitate the use of our system, we are investigating how to better support user interaction by incorporating direct manipulation and multimodal input techniques that allow users to specify their information visualization requests more easily and naturally. We are also exploring how to apply advanced computer vision and graphics technologies to automatically extract various visual characteristics from examples to aid our current hand-annotation process.

Acknowledgments

We would like to thank Jun Hu for implementing the initial version of our similarity measuring algorithm and Bill Yoshimi for reviewing our paper. We are also very grateful that Dr. Steven Roth, Carnegie Mellon University, has kindly let us use figures generated by SAGE.

Cited references and notes

1. G. Robertson, S. Card, and J. MacKinlay, "Information Visualization Using 3D Interactive Animation," *Communications of the ACM* **36**, No. 4, 56–71 (1993).
2. J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Transactions on Graphics* **5**, No. 2, 110–141 (1986).
3. S. F. Roth and J. Mattis, "Automating the Presentation of Information," *Proceedings*, *7th IEEE Conference on AI Applications*, Miami Beach, FL (February 26–28, 1991), pp. 90–97.
4. M. Zhou and S. Feiner, "Efficiently Planning Coherent Visual Discourse," *Journal of Knowledge-Based Systems* **10**, No. 5, 275–286 (March 1998).
5. M. Chuah, S. Roth, and S. Kerpedjiev, "Sketching, Searching, and Customizing Visualizations: A Content-Based Approach to Design Retrieval," *Intelligent Multimedia Information Retrieval*, M. Maybury, Editor, AAAI Press/The MIT Press, Cambridge, MA (1997), pp. 83–111.
6. M. Derthick and S. Roth, "Example-Based Generation of Custom Data Analysis Appliances," *Proceedings*, *International Conference on Intelligent User Interfaces*, Santa Fe, NM (January 14–17, 2001).
7. S. Pan and K. McKeown, "Learning Intonation Rules for Concept to Speech Generation," *Proceedings*, *17th International Conference on Computational Linguistics and 36th Annual Meeting of the American Association for Computational Linguistics*, Volume 2, Montreal, Canada (August 10–14, 1998).
8. S. F. Roth and J. Mattis, "Data Characterization for Intel-

ligent Graphics Presentation," *Proceedings, ACM Conference on Human Factors in Computing Systems*, Seattle, WA (April 1–5, 1990), pp. 193–200.
9. M. Zhou and S. Feiner, "Automated Visual Presentation: From Heterogeneous Information to Coherent Visual Discourse," *Journal of Intelligent Information Systems* **11**, 205–234 (1998).
10. S. Casner, "A Task-Analytic Approach to the Automated Design of Graphic Presentations," *ACM Transactions on Graphics* **10**, No. 2, 111–151 (1991).
11. G. Lohse, K. Biolsi, and H. Rueter, "A Classification of Visual Representations," *Communications of the ACM* **37**, No. 12, 36–49 (1994).
12. J. Marks, "A Formal Specification Scheme for Network Diagrams that Facilitates Automated Design," *Journal of Visual Languages and Computing* **2**, No. 4, 395–414 (1991).
13. B. Myers, S. Hudson, and R. Pausch, "Past, Present, and Future of User Interface Software Tools, *ACM Transactions on Computer-Human Interactions* **7**, No. 1, 3–28 (2000).
14. B. Myers, J. Goldstein, and M. Goldberg, "Creating Charts by Demonstration," *Proceedings, ACM Conference on Human Factors in Computing Systems*, Boston, MA (April 24–28, 1994), pp. 106–111.
15. M. Zhou, "Visual Planning: A Practical Approach to Automated Visual Presentation," *Proceedings*, *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden (July 31–August 6, 1999), pp. 634–641.
16. M. Zhou and S. Ma, "Toward Applying Machine Learning to Design Rule Acquisition for Automated Graphics Generation," *Proceedings*, *AAAI Spring Symposium Series on Smart Graphics*, Stanford, CA (March 20–22, 2000).
17. M. Zhou and S. Ma, "Representing and Retrieving Visual Presentations for Example-Based Graphics Generation," *Proceedings*, *1st International Symposium on Smart Graphics*, Hawthorne, NY (March 21–23, 2001), pp. 87–94.
18. A visual composition is a graph, rather than a tree, because the children might be related to each other, and one child may have multiple parents.
19. See more at http://www.research.ibm.com/RIA/research/Improvise/annotation.htm.
20. Y. Aslandogan, C. Their, C. Yu, and N. Rishe, "Using Semantic Contents and Wordnet in Image Retrieval," *Proceedings*, *20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, PA (July 27–31, 1997), pp. 286–295.
21. T. Mitchell, *Machine Learning*, McGraw-Hill, New York (1997).
22. S. Ma and C. Ji, "Performance and Efficiency: Recent Advances in Supervised Learning," *Proceedings of the IEEE* **87**, No. 9, 1519–1536 (1999).
23. L. Breiman, J. H. Friedman, and C. J. Stone, *Classification and Regression Trees*, Wadsworth Publishing Company, Belmont, CA (1984).
24. J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman Publishers, San Francisco, CA (1993).
25. P. R. Keller and M. M. Keller, *Visual Cues: Practical Data Visualization*, IEEE Computer Society Press, Los Alamitos, CA (1993).
26. R. Wurman, *Information Architects*, Graphics Press, New York (1996).
27. P. Wildbur and M. Burke, *Information Graphics: Innovative Solutions in Contemporary Design*, Thames & Hudson, London (1998).
28. For simplicity, if a picture object contains more than one visual/data object, we treat it as one complex visual/data object.
29. The composition relation within a picture object is the visual/data composition relation between the visual/data object and its children.
30. The visual/data structure feature is excluded here, since it is already implied by the structure of the PVGraph and the PDGraph.
31. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc., New York (1973).
32. See http://www.xmlspy.com.
33. R. Harris, *Information Graphics: A Comprehensive Illustrated Reference*, Management Graphics, Minneapolis, MN (1996).
34. In fact it is common to use the same visual cue to encode the *same type* of data for different data instances to achieve visual consistency. But it causes visual confusion if the same cue is used for two *different types* of data.

Accepted for publication April 29, 2002.

Michelle X. Zhou *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: mzhou@us.ibm.com).* Dr. Zhou is a research staff member at the T. J. Watson Research Center and currently manages the department of intelligent multimedia interaction. She received a Ph.D. degree in computer science from Columbia University. Her research interests include three-dimensional (3D) graphical user interfaces, information visualization, intelligent multimodal and multimedia interfaces, and automated authoring of animated 3D graphical and multimedia presentations.

Sheng Ma *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: shengma@us.ibm.com).* Dr. Ma received a B.S. degree in electrical engineering from Tsinghua University in 1992. He received M.S. and Ph.D. degrees in electrical engineering from Rensselaer Polytechnic Institute in 1995 and 1998, respectively. He joined the T. J. Watson Research Center as a research staff member in 1998, where he is now a manager of the Machine Learning for Systems department. His current research interests are machine learning, data mining, network and computer system management, and network traffic modeling and control.

Ying Feng *Department of Computer Science, Indiana University, Bloomington, Indiana 47405 (electronic mail: yingfeng@cs.indiana.edu).* Ms. Feng is a Ph.D. candidate in the Department of Computer Science at Indiana University. She received her B.S. degree in 1994 in computer science and technology at Tsinghua University in Beijing, China, and her M.S. degree in 1999 from Indiana University. Her research interests include data analysis and visualization, intelligent graphical user interfaces, and the application of 3D graphics and artificial intelligence technology to information visualization. The research in this paper was performed while she was a summer researcher at the T. J. Watson Research Center.