Identifying CAD Objects and Behaviors as a Means of Adapting to Designers

Doris S. Shaw and DerShung Yang, U.S. Army Construction Engineering Laboratory
P.O. Box 9005, Champaign, IL 61826-9005

James H. Garrett, Jr. and Suresh Bhavnani , Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract: Human beings learn to communicate through mutual efforts to develop shared understandings within a given context. With neural network technology, computers may learn to identify objects and behaviors in the CAD context by which they may "understand" the needs of designers. This capability can enable the computer to respond adaptively to assist individual users.

Introduction

The possibility we would like to propose in this paper is, *If we can identify objects and the way they are constructed in a CAD system, then the computer can use the information to adaptively support individual designers.* CAD systems have become more and more complex in order to appeal to wider and wider ranges of customers. CAD is used for everything from construction engineering to jewelry design and is capable of representing drawing scales varying from microscopic organisms to the solar system. The systems include thousands of commands and the general interface permits users to access all of them. In addition, working methods and add-on tools for particular tasks are specialized for different domains. Techniques that are necessary for one user may unnecessarily complicate the use of the system for someone else.

CAD designers have tried to manage the interface in different ways. The long menus are often presented in scrolling pages or tree structures. Menus are provided with other menus that pull-down, pop-up, or display a bewildering array of icons that pull, pop, branch, and scroll. Prompts may ask for key-ins; dialogue boxes require settings, toggles, or selections. Hypertext buttons open windows that may be sized, closed, or lowered. The job of making a selection from the many options is confusing, even if the user is aware of all the choices and knows which one is appropriate for the task at hand. Many professionals who use CAD have complained that it takes so much time to learn the system, which is updated (sometimes with major changes) at least once a year, that they either have little time to practice their profession after mastering the software or else they give up CAD along with the time and cost savings that automation might offer them.

1993 ACM 0-89791-558-5/93/0200/0278

Communication and Learning

The answer involves better communication between people and machines. Admonitions for systems designers suggested by Card, Moran and Newell include that the designer know about "The psychology of the user" and "specify the user population" [1], but as we have seen, the user population in the case of a CAD system is likely to be too broad for such directives. One solution is for the system to communicate adaptively with each individual user, which requires that the computer learn something about the user as the communication is taking place. How that might be accomplished will be the topic that follows.

Communication between humans is not exact or based on rules. It depends upon overlapping experiences by which people create shared meaning [2]. The effectiveness of the communication is measured by the response it receives. For example, it is quite possible for two people to have somewhat different meanings when they hear the word, "car," but to have enough commonality to produce effective behaviors. If someone shouts, "car" at a person crossing the street, almost everyone would respond by moving quickly to the nearest curb - probably an appropriate response. If, on the other hand, a person asks how long it takes to go from point A to point B by car, the answer might vary depending upon individual interpretations of the word, "car." The meaning is dependent upon the context and it evolves with greater understanding as the communication proceeds.

Human learning may be experiential, as Seymour Papert learned about gears when a child [3], or from abstract sources such as reading books and looking at pictures. In order to arrive at appropriate responses and working relationships, it is helpful if the communicating pair actively attempts to build understanding and make use of coexisting indicators of the intended meaning. When using images, which is the concern of this paper, we must remember always that the picture is not the object itself, "The picture is a model of reality" [4]. Reaction to a picture must be to the reality that it depicts, even though the method of learning that reality is different for different people.

It is unlikely that a computer can experience a picture in the same way a person does. To a computer, a graphic is a collection of coordinates, points, and rules organized in a particular sequence in an identified place in its memory. With these, the computer can produce an image that a

person can recognize as a model of reality. The person can draw an object which the computer converts to coordinates, points, and rules and stores them in sequence. If the person and the machine are to share the meaning of a graphic, the method of learning the graphic will no doubt be different, but there must be some overlap of experience if they are to communicate and we can expect the meaning to evolve over time. We know that the way a person draws a picture depends upon what he thinks the meaning of the picture is [5]. Is it possible for a computer to learn to predict meanings from the construction of an object and learn to differentiate between different images?

Image Understanding In A CAD System

Image understanding has always been a very interesting, very practical, and yet very difficult research topic in computer science. Image understanding, in a general sense, ranges from simple two dimensional (2D) pattern recognition to sophisticated three dimensional (3D) scene analysis, with the common goal of mapping the image to the reality it represents. The technical difficulty of image understanding in a CAD system lies between these two extremes, which suggests that there may be a potential for success in the perceivable future.

The difficulties in image understanding are largely due to the distortion of objects in the image. No matter what approach it employs, an image understanding system usually falls short when the input image is distorted in some way that the system does not expect. Unfortunately, the number of ways that an image might be distorted is infinite. Only few forms of these distortions, such as translations, rotations, or scalings, can be described mathematically and thus be handled beforehand. All other distortions, such as variations in handwritten characters, have to be handled purely by the system's own generalization capability. The quality of a generalization should be judged by the overlap between the system's prediction and human perception of reality, which is the fundamental challenge of image understanding.

Several special features of a CAD system lead us to believe that image understanding in a CAD system can be achieved. In a CAD system, distortions can be limited by using precision commands such as "GRID." Noise that might exist in a computer-scanned photograph is normally not present on a CAD window. The temporal information about how the user draws an object can be easily monitored. Raster images and symbolic descriptions of all drawing entities are both available. We believe that all these features can provide sufficient assistance to the task of image understanding within a CAD system. The following sections describe our prototype system that utilizes the spatial and temporal information to recognize "cells" in a CAD system.

Cell Recognition

Most CAD systems provide commands to manipulate a group of drawing entities as a whole, which is what we shall call "cells." Examples of cells in an architectural design can be doors, windows, or pieces of furniture. Cells can be copied, and all properties such as textual descriptions that are associated with the cells will be copied as well. Using cells can thus, on one hand, improve design quality by enforcing objects of the same type to be drawn in the same way, and on the other hand, speed up the design process by avoiding redrawing the same type of objects many times. The U.S. Army Corps of Engineers has compiled a set of standard cell libraries that include more than one thousand cells to be used for producing CAD drawings.

The benefits of using cells are compromised by the difficulties in managing cells. The above mentioned standard set of cells covers seven categories and is stored in 16 cell libraries. On average, each cell library contains nearly 100 cells. To utilize a cell, the CAD user must, first, be aware of the existence of that cell. Second, the user has to know which cell library that cell is in. Third, the user has to know the name of cell, which is usually an abbreviated name, e.g., 'SHNG' for 'single hinged door,' or numerically, e.g., 'MP0028' for 'gas piping.' In most cases, the CAD user cannot meet any of these three requirements and is thus unable to take advantage of using the library of cells. Therefore, we designed a prototype system that can recognize cells while the user is drawing on the screen.

Our Prototype System

Our prototype system consists of three parts: The first part determines where the user is working and captures the screen image surrounding the working area. The second part feeds the captured screen image to a neural network for recognition. The neural network then returns the indices to the two cells that are most similar to what the user is drawing. The third part finally uses these indices to bring the cell to the user's attention. The full description of this system is beyond the scope of this paper. Interested readers should refer to Yang, Garrett, and Shaw [6].

Capturing the Object while Drawing

Our program allows the user to draw freely in a design window. How to isolate the part of the drawing where the user is currently working is one of the problems that has to be overcome. Otherwise, our program has to deal with everything in the design window, which significantly complicates the problem. The entity that the user is currently manipulating is obviously part of the object. The question remains, what the other parts of the object are. As mentioned earlier, two types of information that are available during a CAD design session can give us clues about how to approach this problem. First, cells (objects) tend to be small in scale. In other words, we should look at entities that are close to the entity which the user is manipulating. We call this characteristic "spatial proximity." Second, we have argued elsewhere that a CAD user tends to complete the drawing of one object before

starting on another object [7]. This "temporal proximity" suggests that we look at those entities which were drawn recently. Based on the spatial and the temporal proximities, we can maintain a window, both in time and in space, to monitor the user's drawing process and determine where the user is focusing his or her drawing activity.

Using the temporal proximity, we maintain a window W to cover the last n entities drawn consecutively. The window size (n) is determined by using the spatial proximity. When an entity is created or modified, a window w containing this entity is also created. The spatial relationship between W and w determines how W should be updated to accommodate the newly created or modified entity in w. For example, if w is far away from W, we assume the user is staring a new object and make w as the new W.

Once W is updated, the screen image of W is captured. The captured image is then converted into a 48X48 pixel bitmap image. Pixels with low density such as the grid points are discarded. This image is then normalized in size and placed in the center of the input area before being presented as input to a neural network. Using both the temporal and the spatial information, we are able to isolate the entities that are possibly part of the object that the user is drawing and simplify the problem of recognizing many cells on a large design window to recognizing a single cell on a small 48X48 pixel window.

Neural Network Cell Recognition

Many neural networks have been proposed in recent literature to recognize objects regardless of the presence of translations, rotations, and/or scalings. We have studied five different neural network approaches to the problem of transformation-invariant pattern recognition [8]. We chose Le Cun, et al.'s Zipcode Net [9] as the classifier for our system because of its good generalizability in both rotations and scalings. The Zipcode Net's performance on translation is poor, but can be remedied by adding an appropriate normalization process, such as Yuccer and Oflazer's RST-blocks [10]. Also, the Zipcode Net is very robust with respect to random noise.

Before we describe the Zipcode Net, we need to first describe the preprocessing process, i.e., the RST-blocks. The RST-blocks can normalize a raw image in size, location, and orientation. The T block can normalize the location by aligning the center of the gravity to the center of the input area. The S block normalizes the image size by forcing all images to have the same average radius. The R block rotates the image such that the axis of maximal variance become the x-axis. Our experiments [8] showed that the RST-blocks solve the translation problem for the Zipcode Net. However, the RST blocks were vulnerable to random noise. We found the R block was to be blamed for this disadvantage due to the calculation of the axis of maximal variance. Therefore, we chose to use only S and T blocks for preprocessing and used four versions of the cells, each rotated by 90 degrees, to train the network in order to solve the rotation problem.

The Zipcode Net is a variation of the well-known backpropagation network and is used to recognize handwritten zipcode digits. The Zipcode Net imposes more topological constraints than the traditional backpropagation network such that not all units in adjacent layers are connected. Figure 1 shows a typical Zipcode Net topology.

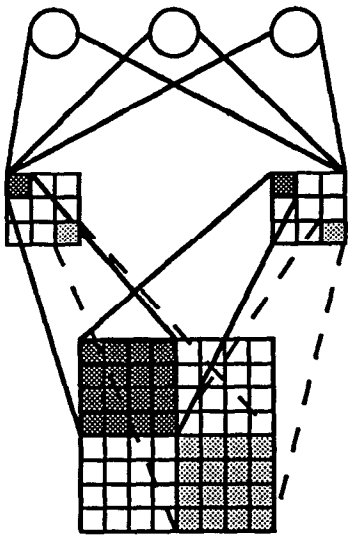

Figure 1: A typical Zipcode Net

In Figure 1, an 8X8 pixel area is used as the input layer. Three output units are used in the output layer. The second layers consists of two "feature maps," each of size 3X3 pixels. In each feature map, each unit accepts inputs from different areas from the input layer. For example, the upper-left unit in both of the feature maps receives inputs only from the upper-left heavily-shaded 4X4 pixel area in the input layer. Similarly, the lower-right unit in the two feature maps receives inputs from the lower-right lightly-shaded 4X4 pixel area in the input layer. This type of connection is called "local connectivity." In traditional backpropagation topologies, all the units in the second layer are connected to all the units in the input layer. Obviously, local connectivity greatly reduces the complexity of the network. This reduction in complexity improves not only the efficiency, but also the generalizability of the network.

Another special feature of the Zipcode Net is "weight sharing." Each feature map functions as a feature detector. The "features" that are important for distinguishing between different objects are determined solely by the network itself. In other words, as the training proceeds, the feature maps can adjust themselves to find the features that are critical for recognizing objects. Weight sharing refers to the fact that all of the units in a feature map share the same set of incoming weights. The purpose of weight sharing is to force the detection of a particular feature regardless of where the feature appears in the input area. A by-product of weight sharing is to reduce the complexity of the network.

Limitations

Although our network has been trained to recognize 23 cells that are commonly seen in architectural floor plans, it inevitably has its limitations. First, it is unclear how the network scales when more and more cells are added to the network. Eventually, we have to handle more than one thousand cells. We are not sure how large the network will be then, and how long it will take to train the network. Second, the training process is not incremental. Adding or deleting cells requires that the training process be re-run from scratch, which seems unnecessary. Third, the network's generalization is not always consistent with human expectation. Occasionally, the network might respond with cells that are not similar to the target cell from the user's viewpoint. Finally, the network cannot determine the best configuration itself. Training the network thus involves lots of trial-and-error effort, which is very time-consuming.

Despite all of these limitations, we feel that image understanding in a CAD system can be realized in the near future for two reasons. The first reason is that we have not fully utilized the temporal and the spatial information available in a CAD design session. Further investigation on how to utilize the information might lead to a significant improvement of our system. The other reason is the tremendous effort that is currently being spent on advancing neural network technology and other related artificial intelligence techniques.

Skill Level and Behavior Characteristics

Neural networks may also assist computer learning in another important area. A study of indicators of user skill level has revealed some preliminary results to guide future research. We have found that experts used a wider range of commands to achieve a higher quality drawing. Experts used more setup commands and less creation commands than novice users. Users with less experience produced a high number of create-erase pairs and produced a lower quality drawing. Such behavior signatures can easily be detected during a CAD design session. A neural network approach to recognizing these behavior patterns in users could take several approaches.

One approach might be to have the units of the input layer represent each of the possible commands. The activation values of these units would represent the frequency of the command usage by the user, normalized over the total number of commands used by the user. This type of network would learn, for example, that experts use more setup commands compared to inexperienced users.

Another approach might be to have a discrete set of sequential events form the input for several different neural nets. So, for example, the first neural net would accept as input the first 20 events performed by a user and the next neural network would accept as input the next 20 events. Each event would be represented by a set of input units representing each of the possible classes of commands that can be invoked within an event. As this approach represents the sequence of events rather than only the frequency, the network could learn that in the first 20 events, inexperienced users have many create-erase cycles.

In detection mode, a controller could pass, in real time, the appropriate number of events to each neural net. The activated neural net would then compute the experience category of the user generating these events.

However, more important than classifying the skill level of a user is the ability to map from the sequence of user commands to a set of higher-level behavior characteristics that are indicative of the presence or absence of usage problems (i.e., the "features" upon which a diagnosis of usage problems can be logically inferred using an AI-based diagnostic approach). For example, recognizing the presence of "create-erase" loops would be useful in that they are symptoms of various CAD usage problems for which specific advice can be rendered. Current research efforts are aimed at using machine learning (specifically neural networks) to cluster patterns of user commands into sets of common, recognizable CAD behavior characteristics (e.g., create-erase loops) that can then be recognized. Given a set of these CAD usage characteristics, a diagnostic system can then be used to infer the presence of problems. Given some identification of the problems, a remediation system can then be used to infer the appropriate advice to render to address the identified problems. Thus, the mapping from user commands into these CAD usage characteristics is the first step to providing adaptive assistance during a CAD session.

Let us look at our main theme of this paper again: *If we can identify objects and the way they are constructed in a CAD system, then the computer can use the information to adaptively support individual designers.* The above sections try to validate the plausibility of the "if" part of our claim, i.e., computers can identify objects and behavior as users draw in a CAD system. Now, we proceed with the "then" part of our claim.

The Adaptive WorkStation

Making the CAD system easier to use calls for an adaptive workstation that can tailor information to the needs of the user. The selection of information should be optimal for the task at hand, unobtrusive, and appropriate in language and approach. As Arno Penzais [11], Nobel Prize researcher, has stated, "Ideas flow better with needed facts at our fingertips — fewer interruptions for look-ups, and fewer detours down blind alleys." Adapting the computer response to individual designers depends upon linking two different types of information.

One body of information consists of the system capabilities that are accessible to users through the CAD system. Design drawings connect with databases containing catalog information and detailed descriptions. Various software

tools such as scheduling expert systems, analysis packages for energy consumption, and object oriented design systems can be accessed by users.

The other class of data relates to the person using the workstation. This includes the information we hope to learn from the objects that are being constructed, the user characteristics of the user and other factors which may contribute to understanding the designer's needs.

Modelling

The key to automating the liaison between users and information lies in representing mental processes and knowledge structures in a way that can link the two. One methodology that has been successful in diagnostic / prescriptive systems has been the construction of mathematical models which associate misconceptions or incomplete knowledge with appropriate assistance. In one study, observation and classification of wrong answers revealed that certain conceptual error patterns mapped into specific areas in a two dimensional graph. If the coordinates of a student's responses were located in certain areas of the graph, particular deficiencies could be diagnosed and the task of providing remedial information was simplified and made more efficient [12]. Such an approach may be useful in managing automated architectural and engineering design knowledge systems to make assistance available to the designer as it is needed.

CAD File Information

Specific parallels to the diagnostic / remedial model can be suggested in the architectural and engineering context. In the above cited study of children's arithmetic, when it was discovered that a student made certain types of errors in subtracting mixed fractions that occurred from not understanding how to convert fractions to different forms, instruction about the identity 1 = x/x was offered to assist the student. The response patterns were observed in a computer file which classified them and mapped them into the "Rule Space" of the graph. Similarly, a designer's use of symbols can be identified with specific disciplines, such as design, construction or building maintenance. CAD usage characteristics fall into recognizable patterns. It has been suggested that behaviors characteristic to the design stage may also be recognized from CAD file information. These patterns can be classified and mapped in order to manipulate the system. CAD files appear to be an appropriate site for observations leading to understanding the designer (at least to the point of being able to provide useful advice).

A Knowledge Model

Once a profile of the user is determined, the computer can facilitate communication between the user and knowledge, but the knowledge must first be structured. The knowledge model must represent the information in such a way that the system can add new knowledge as it becomes available. A useful model might reflect the relationships between skills and usage characteristics, stages of design, and specific disciplines.

One structure for the knowledge system that might be used by the adaptive workstation could be thought of as a three dimensional knowledge model linked through nodes with common centers. At each node, the system could be rotated to other planes in which related information is located. The slope of the plane would be determined by its x, y, and z coordinates where, for example, x, y, and z represent intent or discipline, stage of design, and characteristic usage pattern of the user. When the system discovers a specific design intent during a session, a cluster of knowledge located on a calculated plane would be brought to the attention of the user. As the user moves from node to node, the different planes would be related to the problem at hand.

Figures 2 and 3. show how the information in such a system might be modelled. The particular information plane would be determined by its discipline, stage, and usage characteristic coordinates. If a user is in early concept design he or she would need a different set of tools from the analysis phase of design and would see a cluster on a different plane. The nodes associated with building design stages are all in the highlighted plane as shown in Figure 2. Selecting Specifications shifts the plane to highlight the

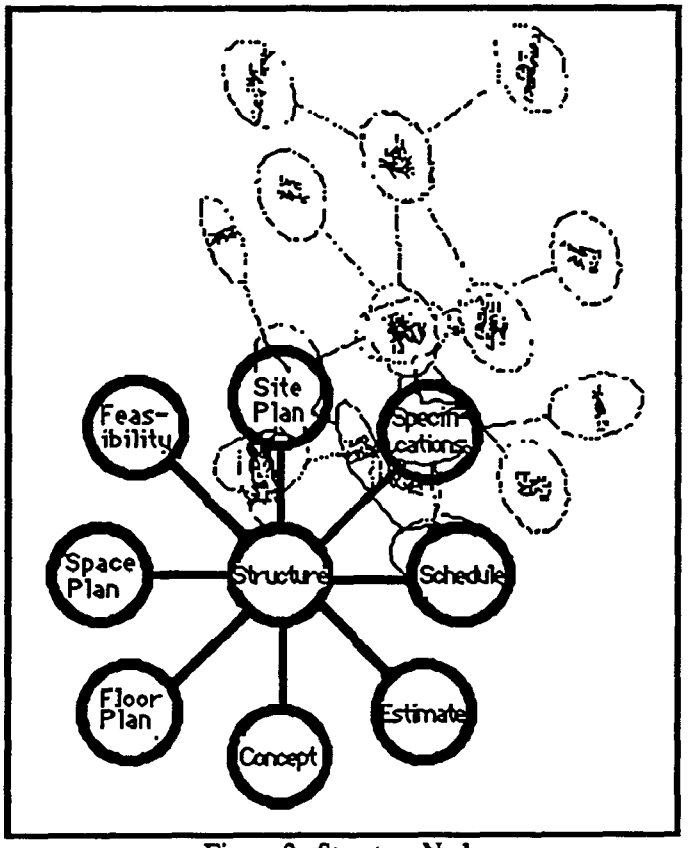

Figure 2. Structure Node
Building Design Phase Considerations

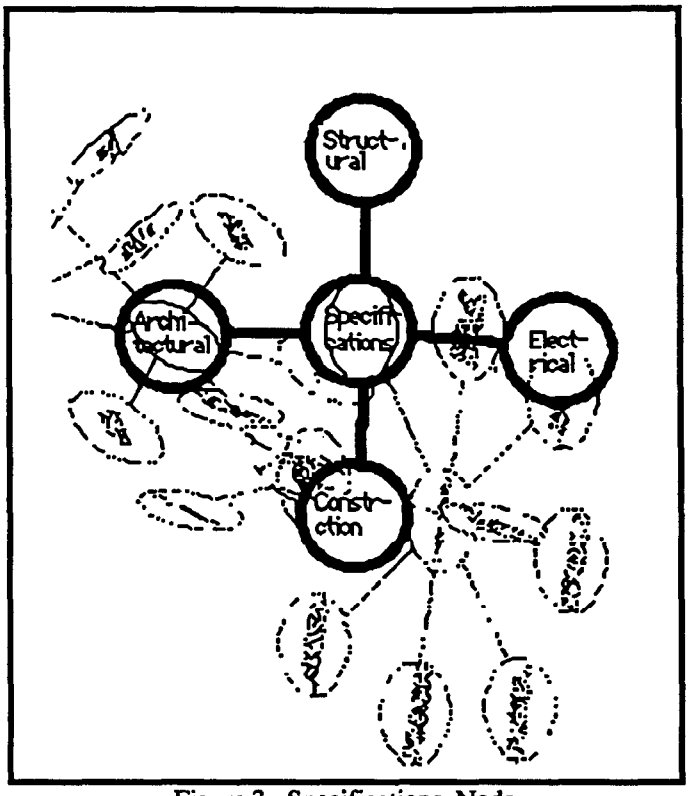

Figure 3. Specifications Node
Discipline Specific Intents

nodes associated with the specifications of different disciplines as in Figure 3. and to dim all other planes. Planes may also be brought into focus by the computer when a need for the information is diagnosed. The three dimensional model shown here is not the user interface, but it represents the informational resources which the interface would make available to the user, a much simpler set of options to display than the usual design environment. New information would require that its discipline, stage, and behavior characteristic approach be specified in order to calculate its position in the model.

Information Relationships

Different types of relationships within the model should be considered. Mathematical links may exist to compare the user data with architectural and engineering requirements. For example, design stages could be represented in sequential or linear order. Tree structured hierarchies, such as discipline specific intents, may have abstraction levels as well as numbered sets of members that may be sequenced. Distance between related nodes may be significant as well as angles between two pieces of information. These and other mathematical relationships must be studied and quantified if possible in order to construct a model that would make relationships between user needs and design knowledge understandable to a computer.

Semantic relationships between data are also being codified. If the system could use the relationships implicit in the knowledge structure, it should be able to place new information in the model and extract user relationships from it later. The workstation would then be able to adapt to the needs of the user.

Conclusions

In our thesis, *If we can identify objects and the way they are constructed in a CAD system, then the computer can use the information to adaptively support individual designers,* we have made the most progress with object recognition. Still there are many unanswered questions. If the network's generalization can be made more consistent with human expectation... If user behavior patterns can be shown to indicate needs for performance support... If a knowledge model can be described into which tools, instruction, and information can be inserted and accessed... If...

References

1.Card, Stuart K., Moran, Thomas P. and Newell, Allen (1983). The Psychology of Human-Computer Interaction. Hillsdale, N.J.: Lawrence Erlbaum Associates, Inc.

2.Delia, Jesse G. and Swanson, David L. (1976). The Nature of Numan Communication. Chicago: Science Research Associates, Inc.

3.Papert, Seymour (1980). Mindstorms. New York: Basic Books, Inc.

4.Wittgenstein, Ludwig (1922). Tractatus Logico-Philosophicus. New York: Harcourt, Brace & Co., Inc.

5.Van Sommers, P. (1984). Drawing and cognition. Cambridge: Cambridge University Press.

6.Yang, DerShung, Garrett, James H. and Shaw, Doris S.(1992). "Cell management using neural network approaches," In Proc. of the 3rd Government Neural Network Applications Workshop.

7.Yang, DerShung and Shaw, Doris S. (1992). "Object recognition in the design environment: A neural network approach." In Proc. of the 6th Int'l Conf. on Systems Research and Cybernetics, Vol. 1, Baden-Baden, Germany.

8.Yang, DerShung, Garrett, James H. and Shaw, Doris S.(1992). "A comparative study of neural networks that can perform transformation-invariant pattern recognition," To appear in Proc. of Int'l Joint Conf. on Neural Network, Bejing, China.

9.Le Cun, *et al.* [1989]. "Backpropagation applied to handwritten zip code recognition," *Neural Computation,* 1:541-551.

10.Yuceer, C. and Oflazer, K. [1991]. "A rotation, scaling, translation invariant pattern classification system," In Proc. Sixth Int'l Symp. on Computer and Information Science, 859-869.

11.Penzais, A. (1989). Ideas and information. New York: W.W. Norton & Co. 170.

12.Shaw, D. (1986). "Cognitive error diagnosis and prescription for fraction arithmetic," In Proc. of the Int'l Association for the Development of Computer-based Instructional Systems Conference. Bellingham, WA: ADCIS. pp. 324-331.