6
Tools for Exploring Associative Reasoning in Design

Richard D. Coyne

Department of Architectural and Design Science
University of Sydney

Two tools for storing and recalling information in computer systems are discussed and demonstrated in relation to design. The tools are hypermedia and neural networks. Each provides a valuable model for reasoning by the association of ideas.

Introduction

The ability to form associations between ideas appears to be an important part of design thinking. As disused in this paper, association provides a means of navigating through information and recalling information. In this paper we, will investigate two important tools for exploring associative reasoning in computer-aided design. These are (i) the use of interactive networks consisting of units of information connected by links (so called 'hypermedia') and (ii) methods of organising and storing information in computer memory as neural networks. These tools (hypermedia and neural networks) serve to highlight two widely different approaches to *information* processing of interest to designers. They each furnish its with useful models that may assist in the creation of computer systems for designers.

Recalling information by association can be seen as a smaller part of reasoning by analogy as a means of sol 'ring design problems and producing design decisions. Research into analogical reasoning by computer generally involves getting the computer to find solutions to problems on the basis of similarities with problems that have already been solved. Solutions and explanations of solutions are stored in a computer system, and some reasoning mechanism is used to establish associations between problems and explanations at different levels of abstraction. The theory and application of analogical reasoning is explored by Schank (1982, 1986), Mitchell at al (1988) and Dejong and Mooney (1986), and some interesting design applications are described by

Navinchandra and Sriram (1987), and Dyer et al (1986).

A full discussion of analogical reasoning and design lies beyond the scope of this paper. We will restrict ourselves to the idea of exploring associations between ideas rather than the general issue of analogical reasoning, bearing in mind that the issue fits within the broader field of study.

Association and Recall

Design *relies heavily* on the ability to recall ideas. Externalising design ideas as sketches can lead to further recall of useful ideas. This process can be characterised as a database problem-searching through a catalogue of graphical ideas (as pictures), and navigating through those pictures by association. It is possible to set up a computer system to demonstrate a rudimentary form of associative reasoning using the Hypercard applications program on the Apple Macintosh computer.

In essence the idea is this: information is stored as a series of 'cards' (records in a database) containing pictures. A human operator moves from picture to picture by abstracting some aspect of the current picture and asking the system to call up another picture that manifests a similar idea. In practice, the abstractions through which associations can be made are generally very simple-just some pictorial element. So if a picture shows a house with a door, it is possible to call up a picture of another artefact with a door, such as a shop or a car. The wheels on the picture of the car may prompt an association with a picture of a bicycle, and so on. In Hypercard, the linkages must be set up in advance by means of invisible buttons positioned above key pictorial elements. The user of the system operates the screen cursor to select which element is to prompt the association. Figure 1 shows the button idea. The buttons are normally invisible, but are shown here as rectangles drawn over key pictorial elements. Figure 2 shows some of the linkages between key elements and pictures within a database (a Hypercard 'stack'). The pictures *were scanned from hand drawings made on site by architecture students* studying the construction of a traditional Fijian buré.

The idea of navigating through pictures in this way has wide appeal for designers. It provides an efficient means of browsing through *reference* material, directories, manuals and encyclopedias. Is can also provide a structure for tutorial systems. Hypercard is able to attach text, programs, sound and animation sequences as well as static pictures to cards. Because of the *interactive nature* of the Hypercard environment it shows promise as a means for designers to organize and *externalize their* own directories of ideas and precedents.

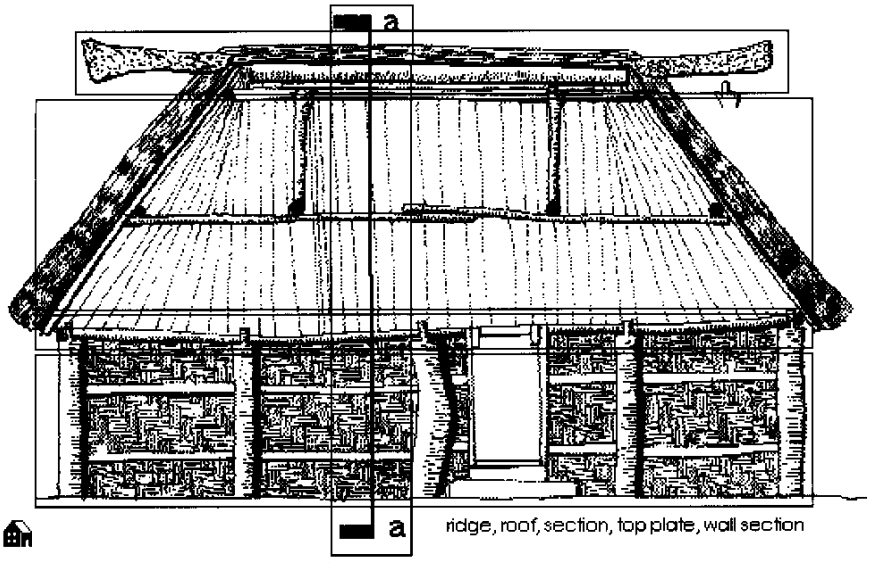

Figure 1 A card from a Hypercard stack showing buttons for recalling other cards by association.

Hypercard is a scaled down application of hypermedia, which makes use of a range of *electronic media,* such as laser disk and networked communications systems. Issues in the *development of* hypermedia are discussed by Smith and Weiss (1988), who include a summary of its history which dates back to 1945.
Of course, the full exploitation of navigating by association requires developments in the hypermedia technology beyond that currently available. In an ideal system for designers it should not be *necessary for* the links between pictorial ideas to be established a priori. How the system is able to form associations on the basis of arbitrary abstractions is an interesting problem in image understanding and pattern matching. Insight into this issue is provided by the second reasoning tool to be discussed *here*.

The Neural Network Model of Information Processing

In the hypercard approach information is stored as *discrete cards* (records). This approach results in useful and *accessible computer* systems, but it bears little relation to the way information is stored in the human brain. One of the limitations of this is that it is *extremely difficult* to model how associations are formed through ideas that match only approximately (a profoundly useful human capability)-for *example, recalling* a detailed picture of a particular house from an abstract sketch, or being prompted to recall a caravan or a dog kennel through its abstract resemblance to a house-in other words, recalling

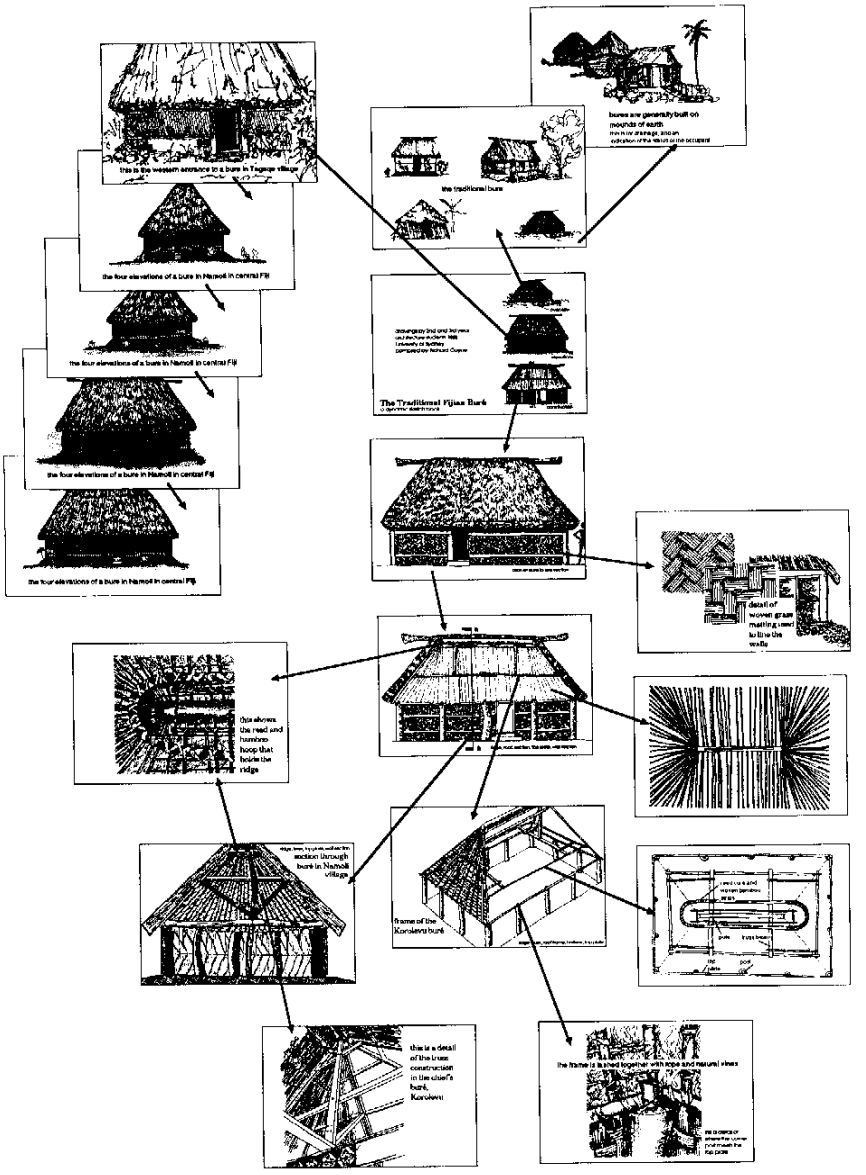

Figure 2 A Hypercard stack and links formed by associations between pictorial elements.

complete information given only approximate, abstract and incomplete information as a key.

This aspect of associative reasoning is addressed by the neural network idea. The neural network idea is powerful, but currently limited by the excessive computational cost involved in simulating parallel neural processes (as well as shortcomings in the development of the theory). In neural networks representation is at a much lower level than in a Hypercard database. Neural networks rely on simple, universal algorithms that must work very hard at a problem. Essentially these algorithms are concerned with optimizing a set of values in a network as a means of storing information (pictures). Recalling information is usually simpler and involves the simple multiplication and summation of values. McClelland et al (1986) provide a summary of techniques available for setting up neural network systems.

In essence the idea is to store patterns (pictures or records) in a data structure known as a neural network, usually as a regular matrix in a computer system. The network bears no discernable physical resemblance to the information being stored, and is intended to simulate the neural connections in the human brain, information is stored as values of nodes and values of connections in the network. The network looks similar when there are five picture stored and when there are five hundred, though the values on the nodes and connections will be different. The number and configuration of nodes and links has considerable bearing on the types of associations that can be established between information. A large number of nodes and linkages generally means a more powerful system, but computational cost rises sharply with larger networks.

The information is stored uniformly over the entire network, in much the same way that information is thought to be stored uniformly across large parts of the brain (Rumelhart and McClelland, 1986). As a new picture is presented to the system for storage the values on most of the nodes and arcs are adjusted slightly in accordance with an algorithm. The various algorithms used generally have a stochastic component and make use of principles of thermodynamics. Storing new patterns involves many iterations of a simple algorithm and can be a slow process.

As with the Hypercard example it is currently only possible to demonstrate associative reasoning at a very rudimentary level. In the examples presented here information is simply stored as coarse bit maps, and the process of finding a match is to present the system containing the stored pictures with a partial picture. The system then recalls the closest matching picture from memory. Theoretically, it is possible to recall pictures on the basis of abstract 'features' that pictures have in common. But this requires more sophisticated algorithms and faster computers than demonstrated here.

Neural network systems are also called 'parallel distributed processing (PDP) systems'. They lend themselves to parallel computer architectures, though the whole process can be simulated with iterative computer programs using standard hardware. Parallel distributed processing machines are considered by some to closely resemble the operations of the human brain at a micro-level. As pointed out by McClelland, Rumelhart and Hinton (1986) this leads to interesting and useful analogies between cognition and computing, and interesting explanations of cognitive phenomena.

Some of these features are summarized here. Once they have learned a set of patterns PDP systems can be regarded as operating 'holistically' in producing interpretations of patterns presented to them. Input patterns are not reduced to primitives for analysis but are processed in totality. The process of discovering similarities and categorising input features is essentially automatic.

PDP models are a good way of simulating the human capacity for pattern completion. Successful recall can be accomplished with imperfect patterns. Where there is poor or ambiguous input the system will always produce some form of output from its memory. This may be an amalgamation of stored patterns or it could be the pattern corresponding most closely to the key pattern. Unlike rule-based reasoning systems, POP systems provide at least some form of output for uncertain input. It could be said that the knowledge degrades gracefully.

PDP systems have a substantial contribution to make to machine learning. Unlike the acquisition of knowledge as rules (as exemplified by Winston [1975], Quinlan [1979] and Michalski et al [1986]), the content of the POP network is not generally meaningful on inspection. However, the examples from which learning takes place are not lost but can be reconstructed from the network. Furthermore, the memory of past events (stored pictures) constitutes a kind of knowledge base from which the system can reason analogically. This is thought by some to present a better model of knowledge-based reasoning for expert systems than the rule-based approach (Dreyfus and Dreyfus, 1986; Stanfill and Waltz, 1986)

An interesting parallel between the PDP method of storage and that used in the brain is that storage of information is not local. Destruction or modification to one part of the network generally results in uniform degradation of the performance of the system rather than the loss of any particular item of memory. Further parallels may be drawn with cognitive behaviour at a macro level. Rumelhart and McClelland (1986) describe an interesting experiment in which it was possible to replicate the behaviour of children learning the past tenses of verbs. The same sequence of standardizing and 'overregularizing' can apparently be observed in human behaviour and the behaviour of the PDP model.

The neural network model discussed here is much simpler than others that have been proposed (for example, the so-called 'Boltzman machine'-Hinton and Sejnowski, 1986). The network described here consists simply of a set of nodes, each connected to every other node. The nodes are arranged in an array, and the patterns to be stored and recalled constitute binary values on each of the nodes. Figure 3 shows the configuration of a network consisting of four nodes.

The parameters of the system are weights on the arcs and threshold values on the nodes. Learning, or storing, patterns of ones and zeros involves automatically adjusting the weights and threshold values. In recalling a pattern from memory given a partial pattern, the decision about whether or not a particular node should be on or off (that is, have a value of I or 0) is a function of the sum of the products of the weights of arcs directed towards the node and the value of the connected node. A net value greater than the threshold means that the node will 'fire' (producing a value of 1). Net values less than or equal to the threshold produce a 0. This kind of operation is propagated throughout the network to produce a configuration of values in response to a given partial pattern. The precise way in which this process occurs is described in greater detail by Coyne and Postmus (1988) based on the ideas of Rumelhart and McClelland (1986).

Spatial Applications of Neural Networks

Reducing information to binary patterns suitable for processing with a PDP system may appear somewhat limiting. However, if we configure these patterns into two-dimensional arrays the application to visual perception and design becomes apparent. Here we explore the use of two kinds of patterns: spatial patterns across a grid and patterns of relationships on a matrix.

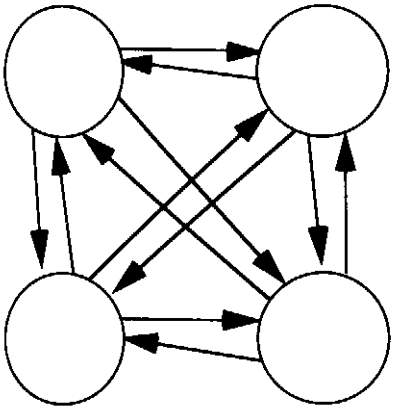

Figure 3. A neural network consisting of nodes and links.

Pattern completion in a graphical domain is demonstrated with the example in figure 4. Nine patterns are stored. Following the learning phase (that is, the stage during which patterns are stored) we present a partially complete pattern to the system (figure 5). In response the system produces the complete pattern of figure 6. The 'noise' in the image is due to the fact that parts of the partial pattern of figure 5 match more than one of the patterns of figure 4, though clearly one is more dominant than the rest. A certain amount of noise is also attributable to inaccuracies inherent in the stochastic process. The noise can be considerably reduced by feeding the noisy pattern back into the system. Half a dozen cycles of this kind generally produce a complete match.

It is interesting to observe what happens when the system is given an ambiguous pattern during the recall phase that is, where the partial pattern contains fragments of several different stored patterns, In this case the output will be a combination of the closest matching patterns: a kind of 'fuzzy union'. This result is not always particularly informative. We can reinforce the closest match by feeding the output pattern back into the system. The result of cycling through this feedback

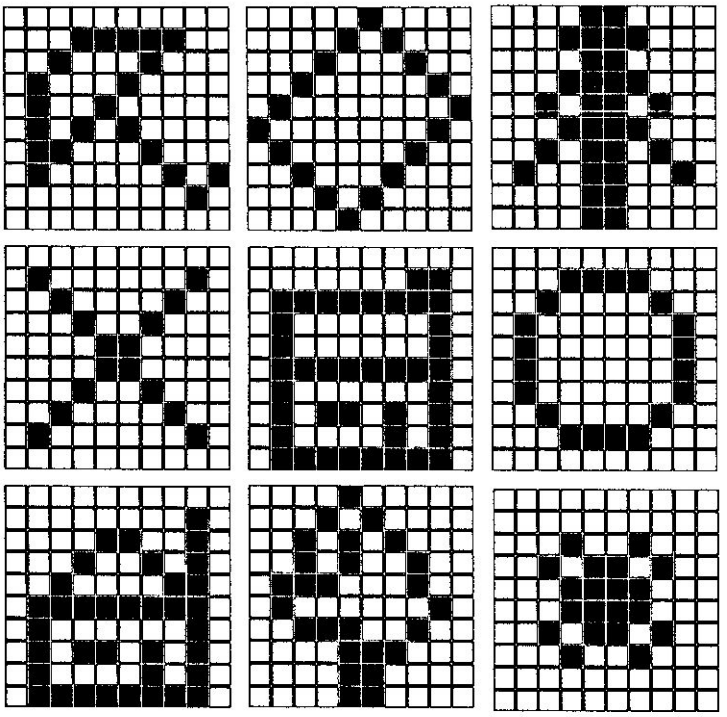

Figure 4. A set of patterns committed to memory by a neural network.

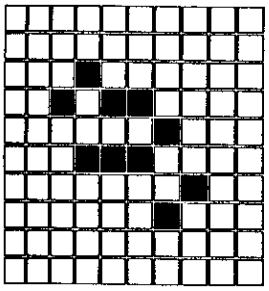

Figure 5 A partial pattern for matching against the memory of figure 4.

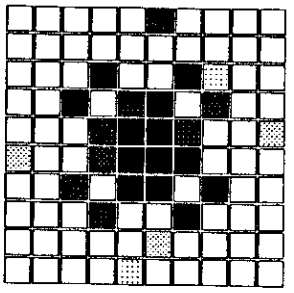

Figure 6 Completion of the pattern of figure 5.

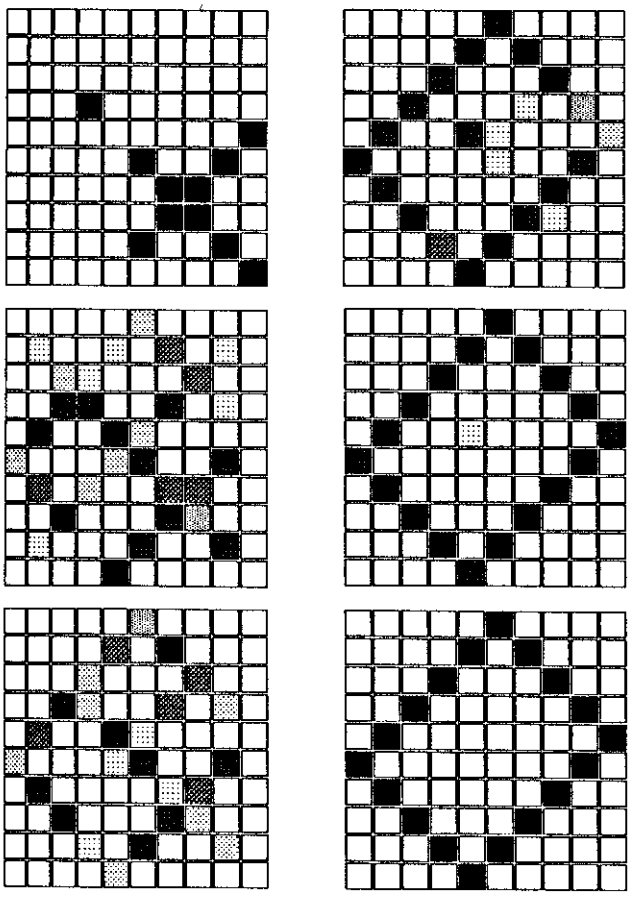

Figure 7 The results of iterative feedback. The original pattern is at the top left corner. The system "homes in on the closest matching pattern after successive iterations.

process for a particular input pattern six times is shown in figure 7. This operation reduces the noise due to overlapping and finds the image that matches the partial pattern most closely.

An interesting design application is where we match floor plan layouts to their performances. Because of the gross nature of pattern completion it is useful to think in terms of matching performances and types, specifically building plan forms. Figure 8 shows how the units can be configured to represent a set of plan forms and their attributes (performances). The attributes are a combination of high-level plan descriptors and the opinions of one designer as to the suitability of the various forms for certain uses and siting conditions.

Figure 8 shows nine plan types and their attributes. To be really useful we should teach the system all possible orthogonal plan forms (this is a large number but is certainly enumerable). If we then present the system with any one of the plans it will generate the attributes. It is interesting that the plan 'drawing' does not need to be perfectly accurate in order to find the right attribute match. Figure 9 shows a progression from the inaccurate, partial pattern presented to the system and the result after several feedback iterations.

Of interest to designers is the operation where we present the system with a set of attributes. The system then 'generates', or selects, a plan form. If we exploited the set theoretic properties of the system demonstrated above then we would produce loosely drawn unions of the closest matching plan types. There are three ways of treating this information. (i) It can be seen as a loose kind of synthesis for producing new types (assuming all types are not stored). Unfortunately the combination of plans rarely produces attributes that are simply derived from the union or intersection of the attributes of each of the plans. (ii) The overlapping patterns may serve to indicate the closest matching patterns from which the human operator of the system can make a choice. (iii) We can use the iterative feedback operation to produce the closest matching plan. Figure 10 demonstrates the latter approach.

In figure 10 we show the stages in the iterative feedback process by which the system 'homes in' on the closest matching plan form. Of course the iterative process need not be visible to the operator, but it is informative here in showing what the system is doing. There are three important observations we can make about this operation where we provide a partial pattern of only a small number of performances during recall. (i) We are attempting to find a matching form when there are in fact several forms. The most prominent forms appear shadowed after the first iteration in figure 10. (ii) We are attempting to complete a pattern where we supply the system with only a very small pattern

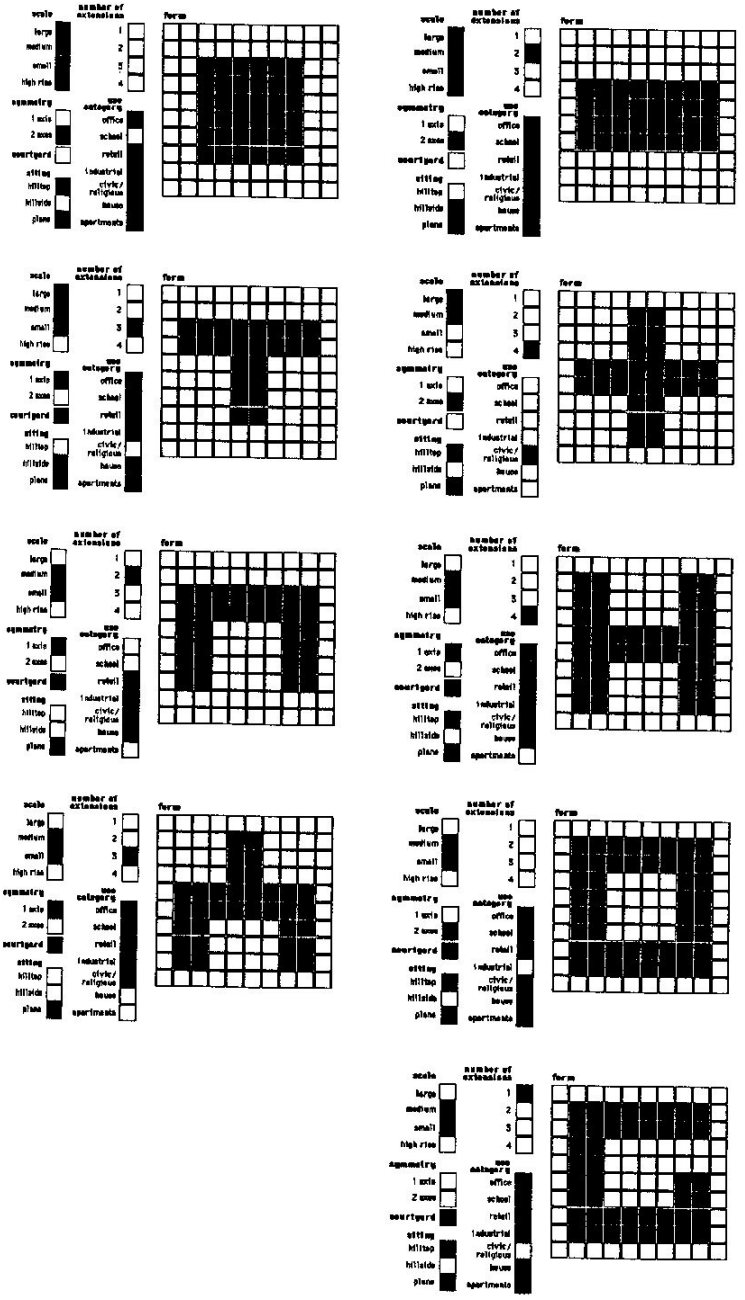

Figure 8 A typology of building plans

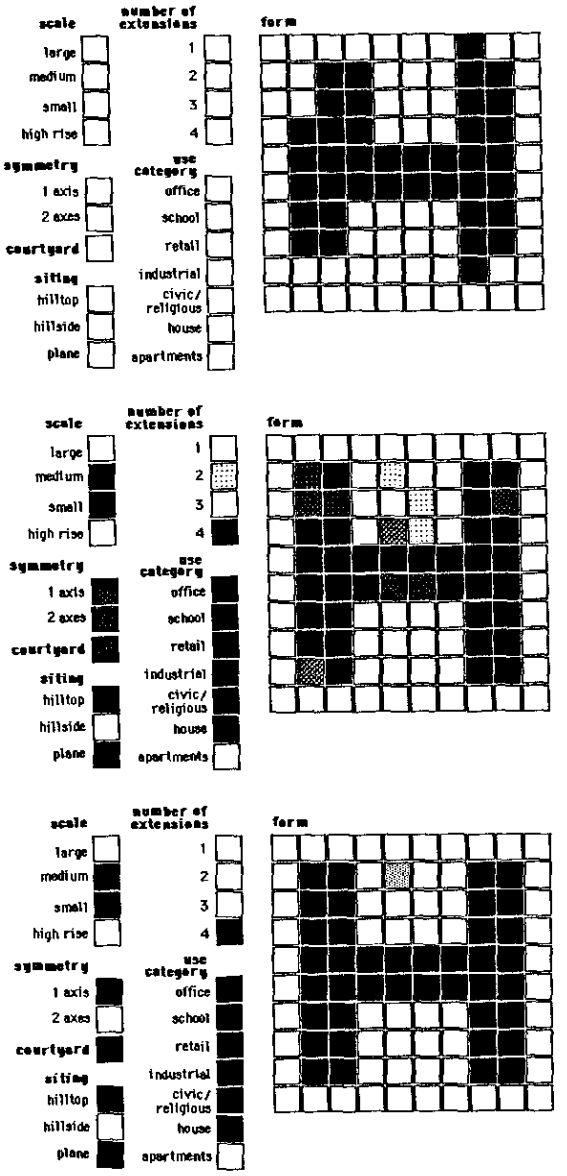

Figure 9 Matching a type from an approximate and partial image

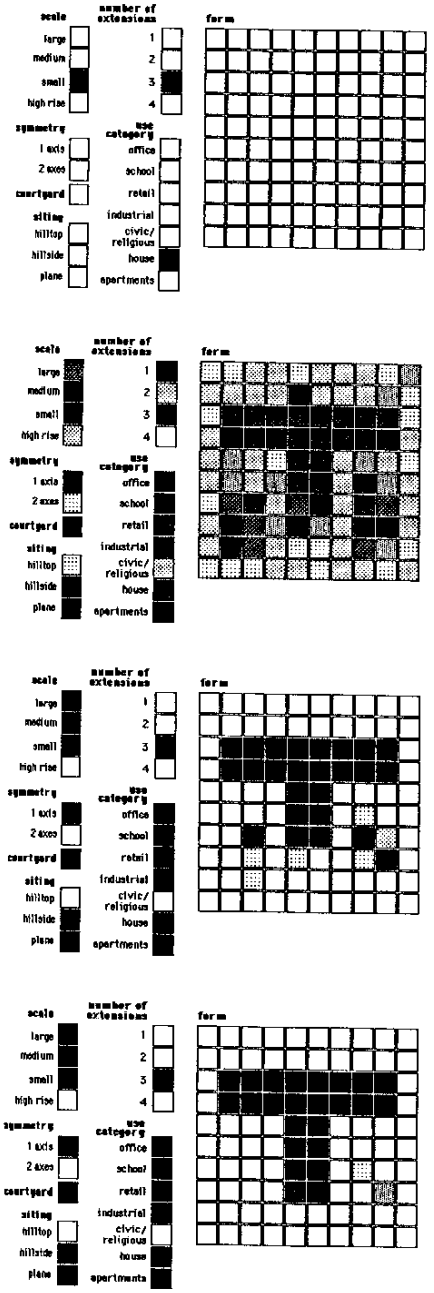

Figure 10 Producing a form from a given set of attributes

fragment. This considerably increases the noise in the result. (iii) The partial pattern may closely match with several plan forms. On successive iterations the system produces a plan shape that closely matches the combination of these plans. The resultant form may have very different performances to those specified in the original partial pattern. This can be overcome by introducing some kind of weighting factor into the representation so that one unit of performance is equal to several units on the form grid. Of course, for some applications the combination of plans may be very useful.

It is interesting to note that the neural network system described here was implemented using Hypercard and Turbo Pascal. The graphical interface to Hypercard provided a useful development environment. As well as being stored in the neural network, each of the stored pictures is retained as a record in Hypercard. Of course, it is not necessary, and indeed wasteful, to retain a database of individual pictures, as they are contained within the neural network. But the explicit storage of the individual pictures was of assistance during experimentation. It also serves to suggest that the ideas behind hypermedia and neural networks may be integrated to good purpose.

Conclusion

The idea of hypermedia provides a useful model of the storage and recall of information through associative links established a priori. It also makes use of well-established technology for storing information. However, the neural network idea accords more with the way in which information appears to be stored in the human brain, it has been demonstrated how neural networks facilitate the recall of stored patterns from partial patterns. The potential of neural networks to facilitate a crude kind of design synthesis has also been demonstrated.

In spite of its substantial history, the neural network idea is still in its infancy. Storing and recalling pictures is computationally expensive. There is considerable benefit to be gained from an integration of both approaches. Full pictorial information could be stored in hypermedia, while certain abstractions of those pictures, perhaps in the form of tables as depicted in figure 8, may be stored in a neural network representation. Navigation through the card system could be via associations established through the neural network. The integration of the two ideas poses a formidable research challenge, but may provide further insights into the design process, and assist in the production of better computer systems for designers.

Acknowledgements

This work is supported by a University of Sydney Special Project Grant. The assistance of Arthur Postmus is gratefully acknowledged.

References

Coyne, RD. and Postmus, AG. 1988. 'Spatial Applications of Neural Networks in Computer-Aided Design". Working Paper. Sydney: Department of Architectural Science, University of Sydney.

Dejong, C. and Mooney, R. 1986. "Explanation-based learning: an alternative view. pp.145-176. *Machine Learning. Vol.1,* No.2.

Dreyfus, H. and Dreyfus, 5. January 1986. "Why computers may never think like *people. p*p.4I-46. *Technology Review*.

Dyer, MG., Flowers, M. and Hodges, ,T. 1986. "EDISON: an engineering design invention system operating naively." pp.36-44. *Artificial Intelligence in Engineering.* Vol.1, No.1.

Hinton, G.E, and Sejnowski, T.J. 1986. 'Learning and Relearning in Boltzmann Machines". pp 282-314. In Rumelhart, D.E. and McClelland, J.L. (eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1 Cambridge: MIT Press.*

McCle!land, J.L., Rumelhart, D.E. and Hinton, G.E. 1986. 'The appeal of parallel distributed processing." pp.3-44. In Rumelhart, D.E. and McClelland, J.L. (eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1 Cambridge: MIT Press.*

Michalski, R.S. 1983. "A theory and methodology of inductive learning." pp.111-161. *Artificial Intelligence.* Vol.20.

Mitchell, TM., Keller, R. and Kedar-Cabelli, 5. 1986. pp.47-80. "Explanation-based generalisations: a unifying view.' *Machine Learning. Vol.1,* No.1,

Navinchandra, D. and Sriram, D. 1987. "Analogy-based engineering problem solving: an overview", *Artificial Intelligence in Engineering: Tools and Techniques,* D. Sriram and RA, Adey eds, Computational Mechanics, Southampton, pp.273-285.

Quinlan, JR. 1979. "Discovering rules by induction from large collections of examples", *Expert Systems in the Micro-Electronic Age,* D. Michie ad, Edinburgh University Press, Edinburgh, pp.I68-201.

Rumelbart, D.E. and McCIelIand, J.L. 1986. "On learning the past tense of English verbs". In Meclelland, J.L. and Rumelhart, D.E. (eds). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume* 2, *Psychological and Biological Models,* MIT Press, Cambridge, Massachusetts, pp.2l6-27I.

Schank, R.C. 1982. "Dynamic Memory: A Theory of Reminding and Learning." In *Computers and People.* New York: Cambridge University Press,

Schank, R.C. 1986. *Explanation Patterns: Understanding Mechanically and Creatively*.

Hillsdale, New Jersey: Lawrence Erlbaum.

Smith, J.B and Weiss, S.F. 1988. 'Hypertext". pp.8l6-8l9. *Communications* of *the* ACM. Vol.31, No.7,

Stanfill, C. and Waltz, 0. 1986. "Toward memory-based reasoning." pp. 1213-1228. *Corn in on* it *at bus of the* ACM. Vol.29, No.12.

Winston, P. 1975. *The Psychology of Computer Vision,* New York: McGraw-Hill.