# A Deployed Engineering Design Retrieval System Using Neural Networks

Scott D. G. Smith, Richard Escobedo, Michael Anderson, and Thomas P. Caudell

*Abstract*— **We describe a neural information retrieval system (NIRS), now in production within the Boeing Company, which has been developed for the identification and retrieval of engineering designs. Two-dimensional and three-dimensional representations of engineering designs are input to adaptive resonance theory (ART-1) neural networks to produce clusters of similar parts. The trained networks are then used to recall an appropriate cluster when queried with a new part design. This application is of great practical value to industry because it aids in the identification, retrieval, and reuse of engineering designs, potentially saving large amounts of nonrecurring costs. In this paper, we review the application, the neural architectures and algorithms, and then give the current status and the lessons learned in developing a neural-network system for production use in industry.**

*Index Terms*— **Adaptive resonance theory (ART), clustering, design retrieval, group technology, information retrieval, neural network.**

## I. Problem Statement

INTRODUCING A NEW part into a manufacturing environment can cost many thousands of dollars. This includes expenses not only for part design, but also for associated processes such as tool design, tool fabrication, process planning, production scheduling and control, and records and documentation. For companies producing large customized systems, the proliferation and maintenance of new part designs can be very costly. Often identical designs are inadvertently produced, wasting time and money. This happens frequently on large systems that involve multiple teams designing parts at different sites. One designer may have no knowledge of another's work unless the technology exists to classify, store, and retrieve designs.

In industrial engineering, group technology (GT) embraces the concept of the reuse of processes, where appropriate, in order to conserve resources, reduce flow time, and reduce production costs. The reuse of engineering designs is one area of GT that has received much attention. Several studies have attempted to ascertain the magnitude of the part proliferation problem. Gunn [1], in an article in Scientific American, depicts a particularly bleak picture. He estimates that "only 20% of the parts initially thought to require new designs actually need them; ... 40% could be built from an existing design and 40% could be created by modifying an existing design."

Manuscript received February 3, 1997; revised March 5, 1997.

S. D. G. Smith, R. Escobedo, and M. Anderson are with the Research and Technology Organization, Integrated Support Services Division, The Boeing Company, Seattle, WA 98124-2207 USA.

T. P. Caudell is with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131 USA.

Publisher Item Identifier S 1045-9227(97)04827-3.

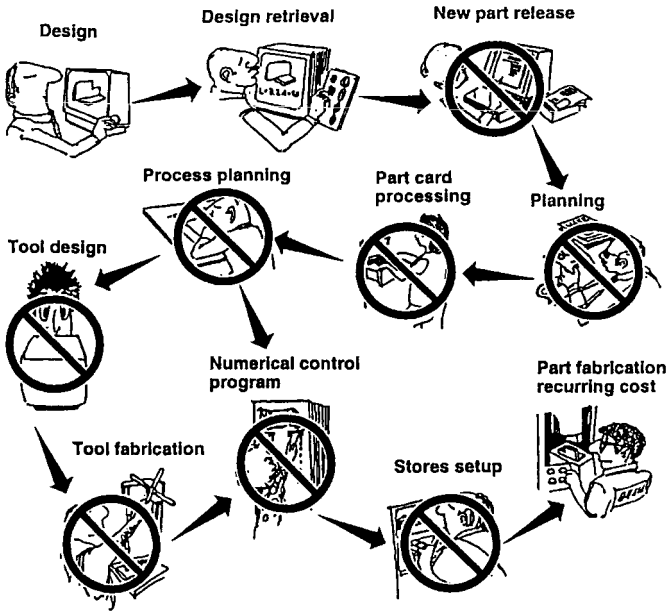



Fig. 1. Typical nonrecurring manufacturing steps. Crossed out figures are the steps that can be avoided if an existing design is reused.

The results of a study [2] published in 1989 by Hyer and Wemmerlov document the benefits of part design retrieval. Hyer and Wemmerlov surveyed 53 U.S. manufacturing companies, including several aerospace firms, that represent a wide range of sales revenues and a broad array of product lines. Among those companies implementing group technology in part design retrieval, an existing unmodified design was used in an average of 20% of the instances in which a new design was considered. In an additional 18% of the cases, existing designs were slightly modified instead of creating new ones.

Fig. 1 shows some of the typical nonrecurring manufacturing steps that must occur for every released design in preparation for fabrication of the part. The design phase represents only a small fraction of the cost of this process. If a similar or duplicate design can be found before the engineer releases a seemingly new design, a large fraction of these down stream processes can be reused, thus saving their development costs. In addition, if a design is reused, all of the quality issues and documentation for that design will be reusable, improving the general quality of the product.

Previous methods of design retrieval have often relied heavily on indexing or coding schemes based on part features.

Unfortunately, systems based on these methods have proved cumbersome to create, maintain, and use. Also, they are labor-intensive and lack the capability to comprehensively include a large variety of relevant shapes and attributes. Methodologies such as relational databases, object-oriented systems, and feature-based expert systems, have been applied to the part design retrieval problem, with similar results.

For GT to be optimally successful, it is necessary that it be as painless as possible to integrate into the existing environment. The application should be largely transparent to the user, who must be able to query the database and relate the representations contained therein to similarly configured part designs with minimal involvement.

In the following sections, we review a neural-network solution first described in Caudell *et al.* [5]. In the latter sections, we give the current status of this system and share some of the practical leasons learned in deploying a large industrial neural network.

## II. Neural- Network Architecture

We use the adaptive resonance theory (ART) neural-network model developed by Carpenter and Grossberg [3]. The version of this model that processes binary input patterns is referred to as ART-1. The ART-1 neural-network model is represented by a set of coupled ordinary nonlinear differential equations [3]. If appropriate assumptions are made about the relationship between the learning rates, the dynamical time constants, and presentation times, this system of equations can be reduced to a procedural algorithm [4]. This "fast learning" mode of operation requires that the learning process stabilize on an input pattern before the next one is presented.

This algorithm implements self-organized, or unsupervised, learning, grouping binary input patterns into clusters of similar patterns. The number of clusters is not preset at the beginning, but is determined by the distribution of input patterns used during training. The number of clusters is also affected by a small set of network parameters, the most significant of which is the vigilance value.

Two-dimensional (2-D) and three-dimensional (3-D) engineering designs are extracted from CAD systems and converted to binary patterns for input to the ART-1 neural networks. After training, the neural networks are used as a "neural database." The engineer queries the trained networks with a new part design. The closest cluster is found and the list of part designs within this cluster is returned to the engineer to decide if any is close enough for direct reuse or minor modification.

Hierarchies of ART-1 networks are used to allow matching of parts at multiple levels of similarity. The ability to select the degree of similarity at retrieval time is accomplished by training a tree of ART-1 networks in which higher levels in the tree are trained with larger vigilance values (Fig. 2). The training occurs by first presenting all of the input patterns to the ART-1 network with the lowest vigilance value. When training is stabilized on this network, the next level of the tree is trained by presenting all of the patterns in a cluster to a new ART-1 network with the next larger vigilance value. This process is repeated, thus forming a tree in which the networks at the top of the tree have the greatest discrimination, while the one at the bottom has the least. When a query occurs, the lowest ART-1 module places the design into one of its clusters. Clusters at this level represent the most general abstraction of the designs stored in the system. After the system has selected a winning cluster at the first level, the appropriate ART-1 module within the next level is activated, if necessary. This module in turn classifies the design into one of its clusters, and the process repeats. The user selects the level of abstraction at retrieval time according to the requirements of the current design.

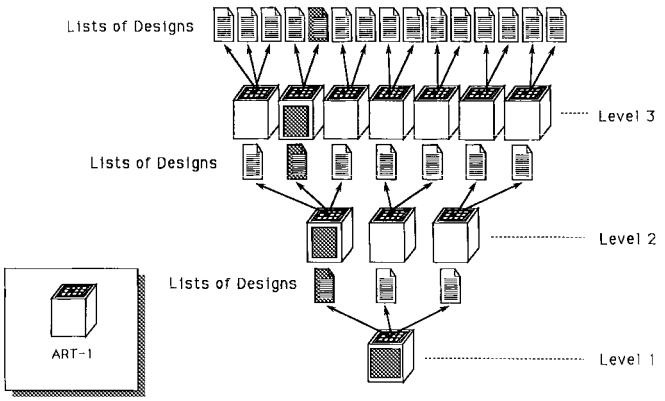



Fig. 2. An example of an ART Tree database structure. Each cube represents an ART-1 neural network.

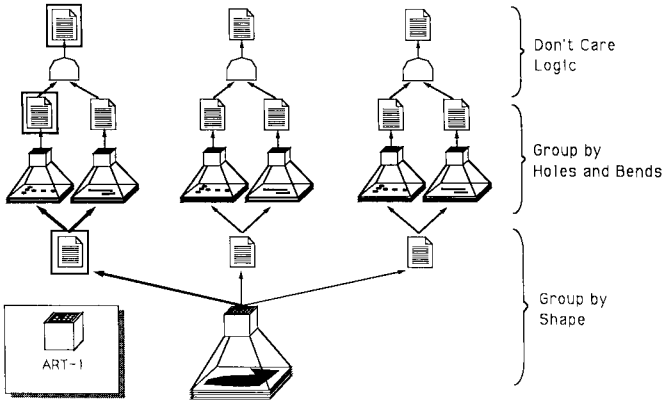
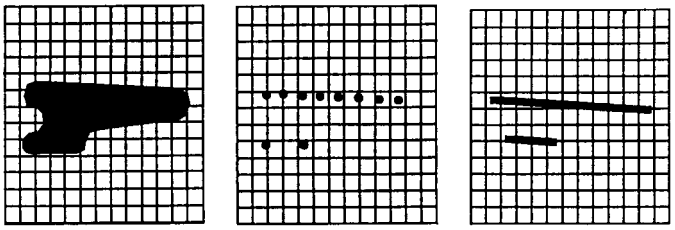



Fig. 3. The macrocircuit of ART-1 modules that implements a feature selection option.

An additional feature of our system allows users to query the database for designs that have the same general shape, and then to further discriminate these according to design features such as the location of bends or fastener holes. The implementation of a feature selection option strongly affects the neural-network architecture, and is implemented here as a loosely connected collection of ART-1 modules called a macrocircuit.

Fig. 3 gives the details of how sets of three ART-1 modules are connected to implement a feature selection option. The detailed structure of this macrocircuit evolves during the training process, where a training set of part designs is repetitively presented to the networks. Within this macrocircuit, the shape representation is considered first by the bottom "shape" ART-1 module. For each cluster formed by this module, an additional



(a) (b) (c)

Fig. 4. Three representations of features of a design: (a) the silhouette of the part, (b) the location of fastener holes, and (c) the location of bend lines. Each of these is converted into a linear binary vector for input to the neural networks.

pair of ART-1 modules, the "holes" and the "bends" modules, is spawned for secondary training. The top of this structure in Fig. 3 represents those parts that satsify both the holes and bends requirements. It is the intersection of the two lists below it. This architecture gives the user the ability to discriminate on 1) shape alone; 2) shape and holes; 3) shape and bends; or 4) shape, bends, and holes.

## III. Input Representations

In this application, the neural networks are trained on part design representations derived directly from graphical descriptions generated by CAD systems. Before the representations can be created, the parts must be moved to a standard position and orientation. First, the center of mass of the part is used to shift the part to a common origin. The moments of inertia are then used to determine the principal axes of the part, which define a standard orientation. See [5] for the details of this process.

For a 2-D design, the simplest input representation is a binary pixel map, or silhouette, centered in a coordinate system: "on's" (1's) where there is solid material and "off's" (0's) where there is none. An example is shown in Fig. 4(a). The resolution of the representation must be chosen to resolve the finest salient features in any design to be stored in the system, and is generally set empirically. Currently, we are using a resolution of 400 by 400 binary pixels.

Other forms of design information may be represented as binary patterns, as well. For example, Fig. 4(b) illustrates how the position of fastener holes can be represented in a different array with dimensions the same as the shape silhouette, this time with "on's" in the neighborhood of a hole, and "off's" otherwise. A bend in the metal can be represented by plotting the line of the bend within a 2-D array [Fig. 4(c)].

The representation of 3-D designs requires more consideration than the simple silhouettes presented above. The logical extension of this method would be a binary "voxel" map, where a voxel is defined to be a volume or 3-D pixel. The design would be represented in a 3-D array. In this array, a voxel value of one represents the presence of material, while a zero represents air. For a reasonable resolution of $400 \times 400 \times 400$ voxels, however, the length of the binary vector becomes 64 000 000 b! In addition, this representation is computationally extremely complex. Instead, we use a different approach.

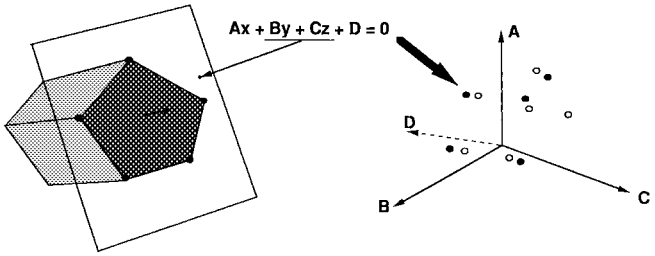



Fig. 5. Polygon parameter representation.

In CATIA, polygonal surfaces are used to generate 3-D designs. The plane of a polygonal face can be represented by the values of $A, B, C$, and $D$ in the equation $Ax + By + Cz + D = 0$, where the constants are chosen to satisfy the equation $A^2 + B^2 + C^2 = 1$. We decided to utilize this information by representing each such polygonal surface, or "facet," as a point $(A, B, C, D)$ in 4-D space (Fig. 5). The many polygons that constitute the surface of a 3-D part will then form a constellation of points in 4-D space. Alternatively, the area of the facet can also be coded, resulting in a 5-D representation. At useful resolutions, there are far fewer points in this representation of a part than "on" voxels in the volume representation. This representation is also much faster to compute.

One obvious weakness of this sparse-polygon parameter space is that the shapes of the polygons are unrepresented. The effect that this has on the ability to accurately match and retrieve similar parts is still being evaluated.

## IV. Training the Neural Database

The training of the neural database takes place off-line by first moving the design data onto the workstation and running a program which loops over all of the designs, building representations for all of the parts. These representations are saved into a file so that only new designs need to be parsed when the database is updated. After all of the designs are parsed the neural networks are trained using the representations created. The input patterns are presented to the ART-1 networks in ascending size, where size is measured by the L1 norm. Georgiopoulos *et al.* [6] have shown that this order minimizes the number of training presentation epochs needed for the networks to stabilize. Currently, the ART-1 hierarchy is trained with five levels of similarity. The training parameters, neural-network "memories," and cluster membership lists are saved to a file which forms the neural database. This file is moved to the platform used to host the neural database.

## V. Querying the Neural Database

Querying the neural databases can be performed in either of two ways. For a query with a part design, a binary representation is first created, using the methods described in Section III. NIRS then uses this representation to look for a match. If a cluster is located that is sufficiently similar, the list of parts in this cluster is returned to the user.

Alternatively, the part number of an existing part can be used as a query. In this case, the database is searched for the cluster that contains that part number. If one is found, the other part numbers in this cluster are returned. In either case, the primary feature used for the searches is the shape of the part. The searches can be further constrained by any combination of the features fastener holes, bends, and manufacturing data (e.g., material, finish, stock size).

Several user interfaces have been built to enable engineers to query the system. The user interfaces allow the engineer to indicate the part geometry or part number with which to query, specify additional query constraints and the level of similarity, view lists of returned part numbers, and compare the geometry of the returned parts.

### A. PC Interface Using CADKEY

With this interface the NIRS code and the neural database reside on a dedicated file server, allowing multiple users access to the system across a local area network. The designer creates geometry using CADKEY and then invokes the NIRS user interface from a DOS shell. This allows the user to specify the query constraints and whether the system should return manufacturing data and/or geometry in addition to the list of similar parts. The interface communicates with the remote server, which performs the query and returns data to the user's PC. The response time for a query on the PC stations is under 30 s. Graphical sketches of the returned part designs are in CADKEY format allowing the user to load them into CADKEY for comparison and analysis.

### B. Mainframe Interface Using CATIA

This interface allows engineers to interactively perform queries while within CATIA. A first screen allows the engineer to select from multiple neural databases and to specify the query parameters. The engineer is able to query using a part number, a solid model, or 2-D geometry from the current workspace. NIRS returns a list of similar parts on a second screen. The engineer is then able to make selections to get overlays of the geometry or to view additional information about the parts. The solid models of the similar parts are retrieved directly from the repositories that store the released designs.

### C. World Wide Web Interface

This interface allows users on the corporate Intranet to fill out an HTML form that specifies the query constraints. The web server processes the form using a CGI-bin script that loads the neural database, performs the query, and formats the response as HTML that is returned to the user. Returned part geometry is imbedded as small in-line images, which can be selected to generate larger images of the parts.

## VI. System Status

The neural databases are currently trained on a SUN SPARCStation 10 workstation. There are two CATIA production databases on the mainframe, one for solid models and one for 2-D geometry. The production solid model database currently has over 55 000 parts, while the 2-D database has 95 000. There are also a number of customized databases for individual design groups.

It is now accessible by several thousand engineers within the state of Washington, with plans to include other Boeing sites in the future.

There is an ongoing study of representation issues, user feedback, and overall system performance as we strive to fine-tune the system for maximal benefit to the user community.

## VII. Lessons Learned and Conclusions

The development of this project has provided a variety of "lessons learned" in practical system deployment. Foremost is the necessity of close communication with the end users of the system during the entire development. Weekly informal meetings were held to discuss project status, user feedback, refinements to specifications, problems encountered, and problem resolution. There were also periodic, more formal "design reviews." Several initial requirements were modified or deleted by the customer, while others were added. In addition, the user interface and the information returned to the user evolved over many versions. Continuous, interactive communication with the end user was indispensable in ensuring the development of a product that closely met the needs of the intended users, increasing the chance that it would be welcomed by the engineering design community when placed into production. Flexibility on the part of the development team was necessary to respond to the changing, evolving design requirements. All of this was essential since building a comprehensive requirements document up front was not possible given that the users did not know the capabilities of the technology nor did they have experience with any comparable retrieval system.

Another important consideration when developing a production neural-network system is that most of the development time and effort is likely to be spent on nonneural-network issues. One of the most difficult portions of the design retrieval system to develop was the module used to parse the CAD designs and determine the geometry of the parts. This involved complex algorithms to locate, recognize, and determine connectedness of primitive graphical entities, such as lines and arcs. Other neural networks could be used to aid in this process, particularly when attempting to extract feature information from designs that predate the application of CAD technology, such as those found on blueprints, mylar, or microfilm.

Memory constraints and performance requirements were a continual driving force behind the development of NIRS. A large amount of time was spent in optimizing the code for memory usage and speed so that it could be deployed on existing workstations and personal computers. This was essential for deploying into the engineering environment, since acquiring new high-end or special purpose hardware platforms was out of the question.

We will soon complete the transfer of the system reponsibility to another group that is chartered, staffed, and funded to maintain large production-level systems. In order to smoothly

effect technology transfer, it is best for the developers to identify early the appropriate group within their company. Working with this group as early as possible, holding regular meetings, developing a transfer schedule, conducting code walkthroughs, and incrementally transfering responsibility plays a major role in making the transfer possible and effective.

In conclusion, we have shown by way of example that neural information systems have reached a level of maturity suitable for use in large-scale industrial applications. In this process we have found it necessary to expend much time and effort on such issues as scalability, preprocessing of data, representation, optimization, system compatibility, and close continual coordination with all affected parties, especially the intended end-users.

## ACKNOWLEDGMENT

The authors would like to acknowledge the assistance and support of D. Campbell, K. Chalfan, M. Healy, T. Murphy, M. Peterson, R. Scott, and S. C. Smith on this project. Without their steadfast support, these systems would never have been deployed. The authors would also like to acknowledge the contributions of F. Holman, B. Marti, C. Shreve, W. Smith, R. J. Tippets, and C. Ward in taking over production responsibility for NIRS. In assuming responsibility, they faced the daunting task of having to comprehend, then carefully document, a large, complex production-level system that evolved out of the research lab.

## REFERENCES

[1] T. Gunn, "The mechanization of design and manufacturing," *Sci. Amer.*, vol. 247, pp. 114–130, 1982.

[2] N. L. Hyer and U. Wemmerlov, "Group technology in the U.S. manufacturing industry: a survey of current practices," *Int. J. Prod. Res.*, vol. 27, no. 8, pp. 1287–1304, 1989.

[3] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, no. 37, pp. 54–115, 1987.

[4] B. Moore, "ART-1 and pattern clustering," in *Proc. 1988 Connectionist Models Summer School*, D. S. Touretzky and G. E. Hinton, Eds. San Mateo, CA: Morgan Kaufmann, 1989.

[5] T. P. Caudell, S. D. G. Smith, R. Escobedo, and M. Anderson, "NIRS: Large scale ART-1 neural architectures for engineering design retrieval," *Neural Networks*, vol. 7, no. 9, pp. 1339–1350, 1994.

[6] M. Georgiopoulos, G. L. Heileman, and J. Huang, "Properties of learning related to pattern diversity in ART-1," *Neural Networks*, vol. 4, pp. 751–757, 1991.

**Scott D. G. Smith** received the B.S.E. degree in computer science from the University of Pennsylvania, Philadelphia, in 1989.

He is now a Senior Principal Scientist in the Research and Technology organization at The Boeing Company, Seattle, WA. He leads the Adaptive Neural Systems project, which is developing neural network technology and applications to use in improving Boeing's products and processes. This project has had great success in transfering neural network technology to production use within the company.

**Richard Escobedo** received the A.B. and Ph.D. degrees in mathematics from the University of California, Berkeley.

He was an Assistant Professor of mathematics at The University of California and San Jose State University, San Jose, CA. He is currently in the Research and Technology organization of The Boeing Company, Seattle, WA, doing applied research in neural networks, with an emphasis on improving engineering and manufacturing processes within the The Boeing Company.

Dr. Escobedo was the chair of The Clinic on Neural Network Applications for Manufacturing Product/Process Control in 1993.

**Michael Anderson** received the A.B. degree from Harvard University, Cambridge, MA, in 1964, the Ph.D. degree from The Johns Hopkins University, Baltimore, MD, in 1969, and the M.S. degree from the University of Washington, Seattle, in 1989.

Since 1986 he has worked for The Boeing Company, Seattle, in Research and Technology. His research interests include industrial applications of neural networks.

**Thomas P. Caudell** (M'91), for a photograph and biography, see p. 474 of the May 1997 issue of this TRANSACTIONS.