INT. J. PROD. RES., 1995, VOL. 33, NO. 2, 405–426

Intelligent design retrieval and packaging system: application of neural networks in design and manufacturing

A. BAHRAMI†§, M. LYNCH‡ and C. H. DAGLI*

We describe a hybrid intelligent design retrieval and packaging system by utilizing techniques such as fuzzy associative memory, backpropagation neural networks, and adaptive resonance theory. As an illustrative example, a prototype of the proposed system has been developed to intelligently retrieve a design from a standard set of chair designs that can satisfy the required needs. The system then automatically passes the design to an intelligent packaging system which locates the parts needed from a designated area and packages the parts in the packaging area. This novel application of neural networks could establish the basic foundation of a true intelligent manufacturing system.

1. Introduction

The quest for building systems that can function automatically has attracted a lot of attention over the centuries and created continuous research activities. As users of these systems we have never been satisfied, and demand more from the artifacts that are designed and manufactured. The current trend is to build autonomous systems that can adapt to changes in their environment. While there is lot to be done before we reach this point it is not possible to separate manufacturing systems from this trend. The desire to achieve fully automated manufacturing systems is here to stay.

Decision complexity is and will be an issue in manufacturing systems. This is due to the fact that excessive design and operation alternatives exist in the products to be produced and the choice among appropriate combinations is not an easy task. Recent changes in the global economy, and intense international communication capability, has created the global market which necessitates flexibility in manufacturing systems to be able to compete effectively with companies emerging around the globe. There is a definite trend to move into customized products with short life cycles and response to market changes pretty much instantaneously. The flexibility that we are searching for requires integration among basic functions of manufacturing, namely; product and part design, process planning, programming for machines, robots, automated guided vehicles, production planning, manufacturing, receiving, storage and shipping. Manufacturing flexibility demands customized high quality and low cost products with inexpensive components. This translates into autonomous machine setup procedures, automation of design and process planning, and

Revision received April 1993.

†Computer Information Systems, Department of Economics and Management, Rhode Island College, Providence, RI 02908, USA.

‡Department of Mechanical and Industrial Engineering, Louisiana Tech University, Ruston, LA 71272, USA.

*Department of Engineering Management, University of Missouri-Rolla, Rolla, MO 65401, USA.

§To whom correspondence should be addressed.

well integrated manufacturing information systems to get the first part right the first time. Hence, flexibility requires intelligence, and this needs to be integrated with real time control to adapt to changes both in the market and the manufacturing environment on the shop floor.

Utilization of artificial neural networks in the classification and coding of parts for group technology applications have attracted the attention of many researchers (Dagli and Huggahalli 1991). In this paper, however, the application of artificial neural networks in design retrieval and manufacturing is discussed. In order to test the concept, a system is prototyped for designing and packaging a set of standard chairs. The design phase has been automated by retrieving designs from a database of design solutions that can satisfy the marketing demands. Once the appropriate design is retrieved from the database, the system locates the required parts for the selected design within an inventory area and moves the parts to a packaging area.

The intelligent design retrieval and packaging system composed of four IBM PCs networked through IBM token-ring. Figure 1 show the various nodes making up the intelligent design retrieval and packaging system.

1.1. *Automatic design retrieval system (ADRS)*

The ADRS evaluates and maps the marketing characteristics into a set of physical structures based on the *pre-designed* substructures and the knowledge of existing designs for automating the design process.

1.2. *Intelligent packaging system (IPS)*

The IPS consist of two subsystems, the machine vision and robot controller. The machine vision subsystem adds intelligence to the packaging of the design parts. The use of machine vision allows for the random placement of standard parts within the

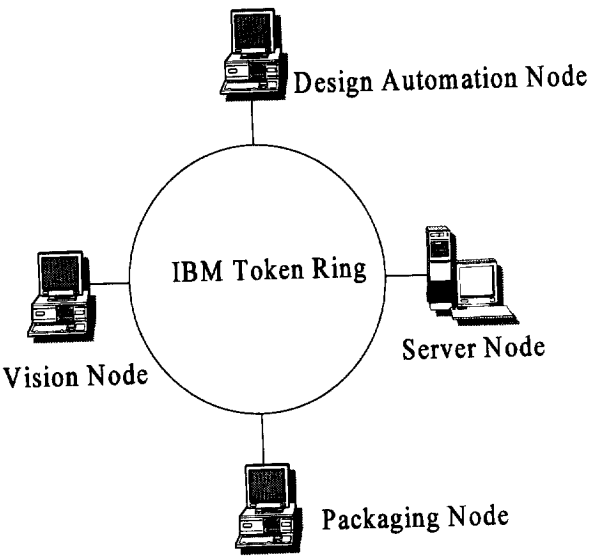

Figure 1. System network.

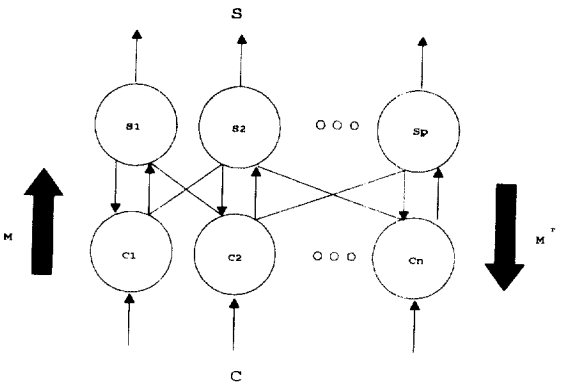

Figure 2. FAM is a two layer feedforward fuzzy classifier.

inventory area. The machine vision subsystem identifies and locates parts, as well as monitors the assembly area to check the packaging of the required parts. The robot controller is responsible for the proper packaging of all the required standard parts in the package. It controls the robotics arm which moves the specified parts from the inventory area to the packaging area.

The rest of the paper is organized into six sections. Section 2 provides some background in fuzzy associative memory (FAM), backpropagation neural networks (BP), and adaptive resonance theory (ART1); §3 describes the design retrieval module; §4 explains the machine vision subsystem and §5 the robot controller subsystem; §6 reports the experimentation and implementation of the proposed system and §7 presents the summary and conclusion.

2. Artificial neural networks—backgrounds

Artificial neural networks (ANNs) are a new information processing technique, they simulate biological neurons using computers. Although most of the physiological details are eliminated by this hardware/software model, artificial neural networks retain enough structure to work like a biological neural processing unit. They are mathematical models of theorized mind and brain activity. ANNs provide a greater degree of robustness or fault tolerance due to the massive parallelism in their design. Neural networks are used in the situations where only a few decisions are required from massive amount of data, or a complex nonlinear mapping needs to be learned.

Here three ANN paradigms that have been incorporated in the proposed system will be examined. These are FAM, BP, and ART1.

2.1. *Fuzzy associative memory (FAM)*

Fuzzy associative memories FAMs (Kosko 1987) are knowledge-based information processing systems. FAM is a two layer 'feedforward' network—where signals allow only information to flow among nodes or Processing Elements (PEs) in one direction (Fig. 2). FAM is 'hetroassociative' since it stores a pattern pair (C_i, S_i). Therefore it operates as a fuzzy classifier that stores an arbitrary fuzzy spatial pattern pair (C_k, S_k) using fuzzy Hebbian learning, where the kth pattern pair is represented by the fuzzy sets $C_k = \{c_1^K, \ldots, c_1^K\}$ and $S_k = \{s_1^K, \ldots, s_m^K\}$. Fuzzy composition is defined as:

$$M = C^T \circ S \tag{1}$$

or, in pointwise notation as [9]:

$$m_{ij} = \min(\mu_c(c_i^k), \mu_s(s_j^k)), \tag{2}$$

where $\mu_c(c_i^k)$ is the degree of membership of the ith member of the fuzzy set C (with the range of 0 to 1), m_{ij} is a fuzzy connection strength from the ith to the jth neuron of the max–min composition between the two layers of the network.

2.2. *Backpropagation neural networks (BP)*

The backpropagation paradigm is a multi-layer network consisting of only feedforward connections (no network feedback). The network can learn both nonlinear functions and hetroassociative pattern classifications through a supervised training sequence. The knowledge about the learned relationships are stored as weights on the interconnections between the network's processing elements. The network learns the input to output relationships through a training processing elements. The network learns the input to output relationships through a training process. In this training process, the network is exposed to a training set consisting of possible inputs and a desired output for each input. The backpropagation algorithm performs the input to output mapping by minimizing a cost function using the squared error. The squared error is the squared difference between the computed output value and the desired output value of each neuron (processing element) in the network. Errors are minimized through making adjustments to the weights on the processing elements' interconnections. So the backpropagation algorithm is a multi-layer, gradient descent, error correction encoding algorithm. It is said these errors are propagated back into the network in order for the network to learn the relationships or patterns more accurately. For details of this process, Simpson (1990) provides an excellent description.

The backpropagation network paradigm has been the most widely used paradigm, primarily for its ability to learn both nonlinear relationships and pattern classifications, its simplicity in coding, and its generalization capabilities. The backpropagation algorithm was developed by Werbos (1974), but the paradigm achieved its popularity after Rummelhart *et al.* (1986) demonstrated its power and showed its great potential. The elementary backpropagation network is a three-layer feedforward network and is illustrated in Fig. 3. It is possible, as will be shown below, to have more than three layers in the network.

2.3. *Adaptive resonance theory (ART1)*

Binary adaptive resonance theory neural networks (ART1) are pattern classifying networks capable of storing an arbitrary number of binary spatial patterns. The ART1 paradigm was originally proposed by Carpenter and Grossberg (1986). The network compares incoming binary patterns to patterns already stored in its long term memory (LTM). If the pattern is sufficiently close to a pattern already stored, the pattern is recognized as the previously stored pattern. If the pattern does not sufficiently match a pattern stored in memory, the network stores the pattern as a new pattern class. This network differs from the backpropagation paradigm in several interesting aspects, which are discussed below.

The ART1 paradigm is a feedback type network having recurrent connections between layers. Figure 4 illustrates an ART1 neural network architecture showing

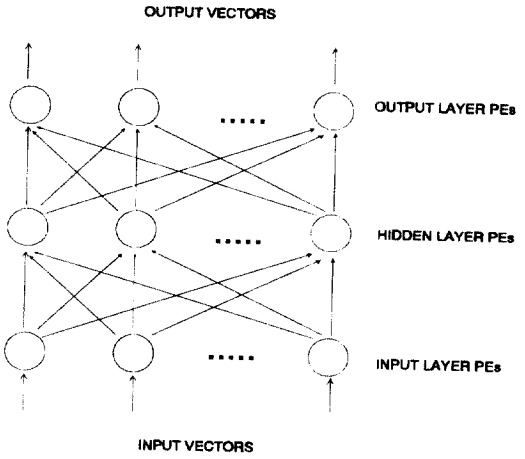

Figure 3. An elementary backpropogation topology.

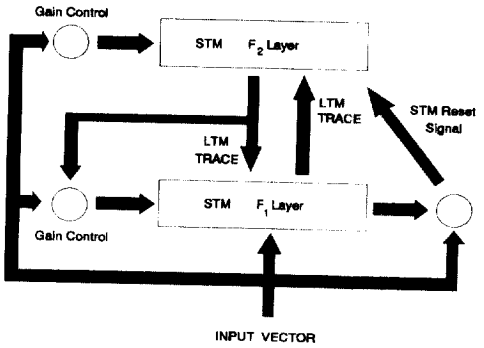

Figure 4. Basic features of ART neural network.

that the network is a two-layer network with both feedforward and feedback connections between the layers. The layers (or fields) are indicated as F1 and F2. The input layer F1 handles the feature representation, while the second layer F2 is for pattern class representation. The feedforward and feedback connections are indicated as the upward pointing LTM trace and downward pointing LTM trace, respectively. The LTM designation represents the long term memory of the network, where the STM represents the short term memory of the network. The long term memory is used for the distributed storing of the pattern information. The short term memories are an integral part of the control and stabilization mechanism that allows for the search of the pattern stored in LTM and then saving of any new patterns in LTM.

The network also utilizes unsupervised learning where there is no training phase. As previously described, input patterns are compared to stored patterns; if the patterns do not match they are stored as a new pattern class. The network performs

this function on its own, thus allowing the creation of a self-organizing pattern matcher.

The coarseness of the matching process is controlled by a parameter called the vigilance factor. The setting of the vigilance factor determines how close the input pattern must be to the stored pattern before a match is determined. The higher the vigilance factor is set the closer the input pattern must be to a stored pattern before a match is found. The vigilance factor is implemented into the architecture by using it as the level to perform a STM reset signal. The STM reset signal indicates that a match has not occurred and the search should proceed to the next stored pattern.

3. Automatic design retrieval system (ADRS)—motivations

The main objective of ADRS is to automate design receival by retrieving a design based on how well a design solution satisfies the marketing demands and users' specifications (Bahrami 1993). Design involves continuous interplay between what we want to achieve and how we want to achieve it (Suh 1990). A designer usually associates sets of fuzzy functional requirements with sets of physical structures. Then during the design process, design solutions are retrieved from memory to solve the design problem.

Customers and the design-build team usually communicate through a natural and somehow ambiguous language that cannot be modelled by conventional two-valued logic. Fuzzy sets can exploit the complexity and ambiguity that may result from communication gaps between users and designers, and turn them into manageable problem domains without ignoring the important factors involved. Fuzzy sets give us the building blocks for dealing with imprecise and overwhelming complex representation problems of input requirements. For instance the class of comfortable chairs can be viewed as a fuzzy set. An executive chair is definitely a member of this fuzzy set with a high degree of membership (close to 1) and a classroom chair is more out of the set than it is in it.

FAM rules, unlike the bivalent rules as symbols in AI systems, consist of numerical entities. These rules are not single value propositions but they are an embodiment of multi-valued sets. These numerical entities of associated 'rules' relax or de-emphasize the articulated, expertly precise nature of the stored knowledge. In the real-world application of a knowledge base, a knowledge engineer can hardly capture the articulated rules; he can only incorporate these rules inexactly and imprecisely.

FAM Paradigm has been selected for the ADRS module because of its capability of mapping a fuzzy spatial pattern pair and ability to reason about fuzzy functional requirements. The one-pair-storage capability of the FAM provides us with a rich knowledge representation mechanism that is modular and at the same time avoids interference between stored patterns.

3.1. *The design problem*

Given sets of fuzzy functional requirements (FRs) and design constraints (Cs) generate the design solution(s) that can satisfy the input requirements. For instance, input requirement for designing a chair may stated as:

Design a very comfortable chair that can be used more or less in public.
FR_1 = chair must be very comfortable.
C_1 = use more or less in public.

Having defined *FR*s and *C*s, how do we generate design solution(s), namely a very comfortable chair that can be used more or less in public. Design constraints have been differentiated from functional requirements since they are basically concerned with the boundary of the design such as size, weight and cost (the input constraints), or capacity, geometric shape and usage (the system constraints) (Suh 1990). In this paper the utilization of FAM in automatic design retrieval is illustrated.

3.2. *Fuzzy knowledge representations*

To experiment with the concept, six generic functional requirements and 11 design constraints have been generated for designing various classes of chairs. Functional requirements are defined as: Ability to adjust the chair (FR_1), ability to move the chair (FR_2), ability to fold the chair (FR_3), stability (FR_4), ability to stack the chair (FR_5), and comfortability (FR_6). Design constraints are defined as: Cost (C_1), size (C_2), weight (C_3), use for dining (C_4), use for office (C_5) use for relaxation (C_6), use for home (C_7), use for classroom (C_8), use for public (C_9), use for typing (C_{10}), aesthetic look (C_{11}). The crisp universal sets FRs of the functional requirements and Cs of design constraints are defined as follows:

$$FRs = \{FR_1, FR_2, FR_3, FR_4, FR_5, FR_6\}$$

$$Cs = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}\}$$

A database of 11 different design solutions has been created (Fig. 5). Associated with each design solution is a list of required parts along with their positions in the final packaging configurations. Each design solution (chair) satisfies a certain set of functional requirements and design constraints. For instance an executive chair may be defined as follows:

$$fr\ (chair2) = \{1{\cdot}0/FR_1, 1{\cdot}0/FR_2, 1{\cdot}0/FR_6\}$$

$$cs\ (chair2) = \{1{\cdot}0/C_5, 0{\cdot}7/C_6, 0{\cdot}3/C_7, 1{\cdot}0/C_{11}\}$$

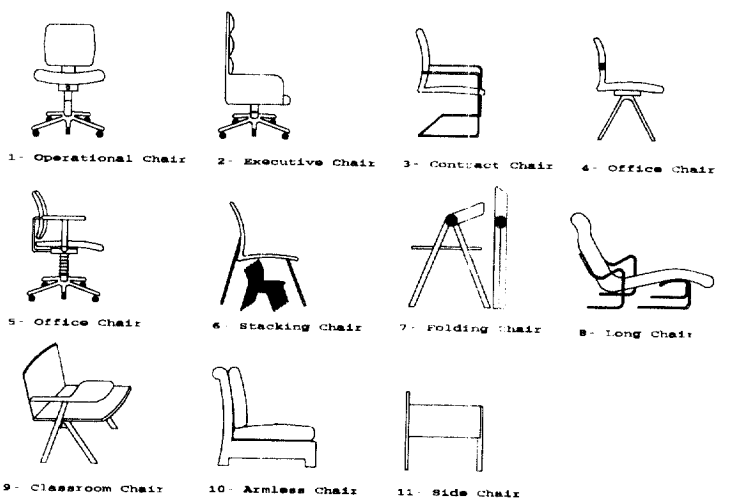

Figure 5. Design solutions.

The above fuzzy sets may be interpreted in English as follows: An executive chair is a chair that is very comfortable with a high degree of adjustability and moveability, furthermore it is used more in the office than home, it also can be used for relaxation; finally the appearance is very important.

3.3. *Triggering and conflict-resolution strategy*

Fuzzy associative memory is used to retrieve a design solution based on how well a design satisfies the user's specifications. The primary disadvantage of FAM is in one-pair-storage capacity. Therefore, for each pattern pair (a design), we must create a separate network. The main problem in implementing the multi-FAM model is the selection of a network based on an incoming pattern.

An algorithm has been formulated to associate an incoming vector to the most applicable network (a design solution). The proposed algorithm consists of two components, these are triggering, and conflict-resolution procedures. During triggering the incoming vector is fed forward and backward through all the networks and triggers each network to generate a vector. Conflict-resolution procedure resolves the conflict by selecting the network that has stored the closest vector to the incoming vector.

3.3.1. *Trigger procedure*

Assume that there are n networks, one for each design solution or associations as follow:

$$\mathrm{M}_i = \mathrm{C}_i^{\mathrm{T}} \circ \mathrm{S}_i,\ i \in \mathbb{N}_n \tag{3}$$

Where C_i and S_i are the antecedent and consequent of the FAM rule i, and $\mathbb{N}_n$ contains the indices to the stored designs.

For instance if we have fuzzy vectors C_i and S_i as:

$$C_i = [1{\cdot}0 \quad 0{\cdot}3 \quad 0{\cdot}7 \quad 0{\cdot}0]$$

$$S_i = [0{\cdot}9 \quad 0{\cdot}0 \quad 0{\cdot}0 \quad 0{\cdot}4]$$

Then M_i is computer as a max-min ($\circ$) of C_i^{T} and S_i:

$$M_i = C_i^{\mathrm{T}} \circ S_i$$

$$M_i = \begin{bmatrix} 1{\cdot}0 \\ 0{\cdot}3 \\ 0{\cdot}7 \\ 0{\cdot}0 \end{bmatrix} \circ [0{\cdot}9 \quad 0{\cdot}0 \quad 0{\cdot}0 \quad 0{\cdot}4]$$

$$M_i = \begin{bmatrix} 0{\cdot}9 & 0{\cdot}0 & 0{\cdot}0 & 0{\cdot}4 \\ 0{\cdot}3 & 0{\cdot}0 & 0{\cdot}0 & 0{\cdot}3 \\ 0{\cdot}7 & 0{\cdot}0 & 0{\cdot}0 & 0{\cdot}4 \\ 0{\cdot}0 & 0{\cdot}0 & 0{\cdot}0 & 0{\cdot}0 \end{bmatrix}$$

Max-min composition ($\circ$) resembles vector-matrix multiplications, where multiplication is replaced by pair minima and summation is replaced by pair maxima.

Now suppose that an *l-dimensional* input fuzzy vector R is the fuzzy input requirement to the system. The vector R is fed through the first layer of each network and fires every FAM rules in parallel but to a different degree. Each FAM rule generates an *m-dimensional* output fuzzy vector P_i, where $i \in \mathbb{N}_n$.

$$P_i = R_i \circ M_i^{\mathrm{T}},\ i \in \mathbb{N}_n \tag{4}$$

Next, pattern P_i is fed back through the second layer (output layer) of each network. Each FAM generates pattern Q_i.

$$Q_i = P_i \circ M_i^{\mathrm{T}},\ i \in \mathbb{N}_n \tag{5}$$

3.3.2. *Conflict-resolution procedure*

The conflict-resolution modeule compares pattern Q_i with the incoming pattern R. Equation 6 is used for measuring the similarity of patterns R and Q_i (Bahrami *et al.* 1991):

$$S(R,Q_i) = \sum_{j=1}^{n} Q_{ij}R_j - \left| \left(\sum_{j=1}^{n} (R_j - Q_{ij})^2 \right)^{1/2} \right| \tag{6}$$

The network with the maximum value of S is the winner and it will be selected to do the mapping.

3.4. *Network topology and data set*

The input, as we discussed earlier, consists of two fuzzy sets, namely FRs and Cs. For the purpose of simplicity, these two sets have been represented as a list. Further, the symbolic representation has been replaced by the positional importance of the degrees of memberships in the list. The input layer is designed to have 17 neurons. One neuron for every functional requirement and design constraint. For example, the functional requirements and design constraints for an executive chair (chair 2) are represented as follow:

$$C_2 = (1{\cdot}0 \quad 1{\cdot}0 \quad 0 \quad 0 \quad 0 \quad 1{\cdot}0 \quad 0 \quad 0 \quad 0 \quad 1{\cdot}0 \quad 0{\cdot}7 \quad 0{\cdot}3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1{\cdot}0)$$

The first value corresponds to FR1 (Ability to Adjust the Chair), second to FR2 (Ability to Move the Chair) and so on. Each degree of membership is assigned to a separate neuron.

The output (second) layer of FAM is designed to have 11 neurons, one for every class of chair. For instance Chair 2 is represented as follows:

$$S_2 = (0 \quad 1{\cdot}0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

3.5. *ADRS performance*

The main issue here is the performance of the ADRS when the number of stored designs are increased. This may raise the question should the growth of the search space be of major concern (in terms of the computational complexity as well as the storage capacity)? The answer is no, for two reason. First, the expert articulation and fuzzy engineering estimation produce only a small fraction of the total possible fuzzy system (Kosko 1987). Second, the fuzzy systems admit degrees. Therefore, FAMs can generalize elegantly based on a single fuzzy association and are able to encode many possibilities. Fuzzy systems map a region of the input space to a region of the output space (A_i, B_i) by utilizing fuzzy-set samples (Fig. 6). Fuzzy systems do not use a numerical-point sample (x_i, y_i) that neural systems use. FAM is not concerned with each point in the input-output product space and it exploits few fuzzy subsets of the input-output product space. Thus, as the actual number of 'needed' (non redundant) FAM rules or the search space grows in a range of $(\log_2^k)$

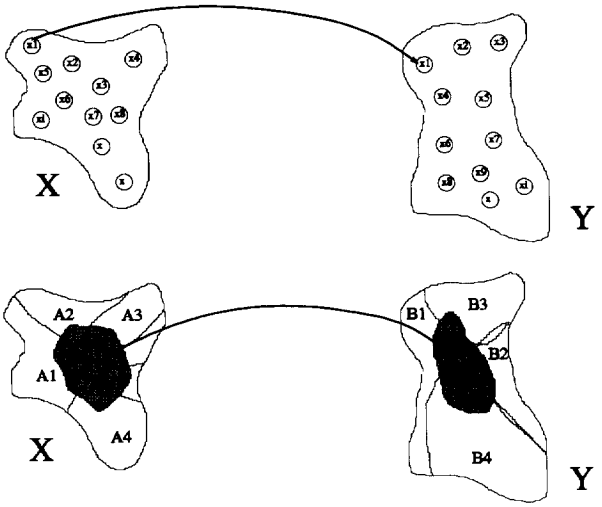

Figure 6. Fuzzy systems map a region of the input space to a region of the output space (Kosko 1992).

to $(k \log_2^k)$. This claim is only an estimate without any mathematical proof and remains a research issue.

To test the system, 20 design scenarios with known solutions have been created. ADRS was capable of successfully generating 18 correct solutions, (Fig. 7). In scenario number 11, ADRS could not generate any solution. Overall ADRS error rate was $<10\%$.

4. Intelligent packaging system (IPS)

The intelligent packaging system (IPS) provides three functions. The first function is the identification, location, and orientation of the standard chair parts located in the inventory area. The second function is packaging of the required

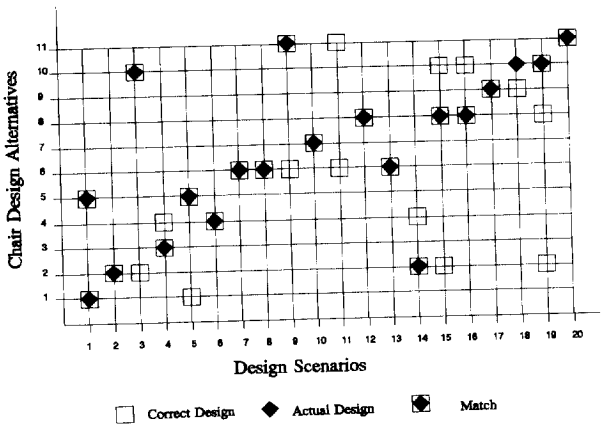

Figure 7. Experimental results.

parts. This function entails moving the designated parts from the inventory area to the packaging area. The third function is the inspection of the packaging area to ensure that the proper parts are included in the package. The IPS is composed of two subsystems. These subsystems are the machine vision and robot controller.

The machine vision subsystem provides the image acquisition, image processing, and part recognition functions needed to locate parts in both the inventory and packaging areas. The subsystem utilizes both conventional and neural network technology in performing the processing needed to locate and package the parts.

The robot controller subsystem accepts the parts' location and orientation information from the machine vision subsystem and the packaging information from the automatic design retrieval system to create a control program for the robotics arm. The control program is downloaded to the robotics arm controller and executed in order to move the desired parts from the inventory to the packaging areas.

4.1. *Machine vision and object identification subsystem*

The function of this subsystem is to identify parts, their locations, and their orientations in both the inventory and packaging areas. It is possible to perform this function using well established, conventional algorithmic approaches, but a goal in this development was to investigate how neural networks could be utilized in system operations.

The processing performed by this subsystem can be broken down into two operations. The first operation is to acquire and process video images in order to determine relevant properties and attributes about objects shown in the images. The attributes would be the objects location (centre of mass) and orientation (angle of the principal axis). The second operation is to use some of the property information to classify each object found in the image into a part category, essentially identifying all the objects found in the image. Figure 8 shows a raw video image acquired of the inventory area.

In determining the subsystem's functions for implementation with neural networks several factors were considered. These factors included:

(1) appropriateness of neural network implementation

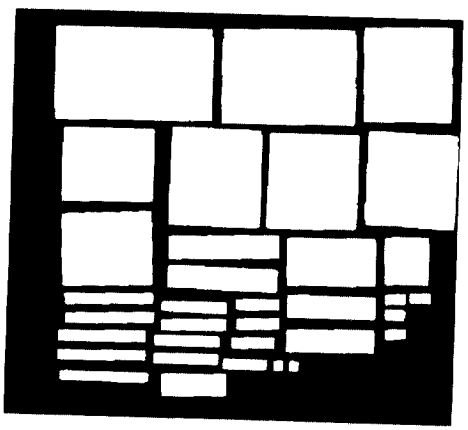

Figure 8. Raw video image of the inventory area.

(2) size of network required for implementation
(3) utility of a neural network implementation

A good area for neural network implementation would have been in the image processing itself; but creating networks large enough to process an entire image was not practical with the current technology. Because of this, all of the image processing was done with conventional algorithmic approaches.

After considering the above factors, two areas were chosen for neural network implementation. One area was that of classifying the objects found in the image into part categories (identifying each object). The second area was the calibration needed to convert the image points found, through the image processing, to the robotics arm coordinates needed to pick up and move the parts.

Most of the neural network applications in vision have been for object classification and recognition (Barnard 1989). In these applications the images are processed in order to classify the objects into classes for proper identification. Neural network classifiers are capable of piece-wise linear or piece-wise quadric decision boundaries (Barnard 1989). Gonzalez and Woods (1992) provide a good description of using neural network classifiers for pattern classification. While the most popular neural network classifiers are those using the backpropagation paradigm, many other paradigms are available to choose from, each with varying capabilities and capacities. However, since ART1 provides on-line learning capability and is able to distinguish small differences in the input patterns, it was selected for the vision subsystem object classification.

Vision system/robotics arm calibration has typically been done by deriving a mathematical model of the systems and then estimating the transformation parameters from the machine vision coordinate system to the robotics arm coordinate system. A neural network implementation for calibration offers a novel approach to solving this problem.

The next several §§ discuss the specific processing occurring in this subsystem. Figure 9 summarizes the operations by expressing the knowledge transformations that occur during the processing. As shown in the figure, the processing is broken

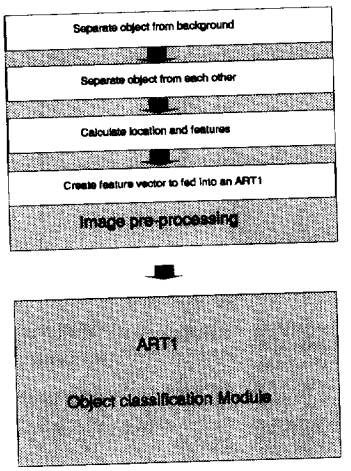

Figure 9. Knowledge transformations in image processing module.

down into two modules, the image pre-processing module and object classification module. In the image pre-processing module, the objects found in the video images are separated from the background and then the objects are separated from each other. With each object found in the image the object's location, orientation and other features are calculated. Each object's features are used by the object classification module to identify what part each object found in the image represents.

4.1.1. *Image pre-processing*

In order to process the scenes as seen from the cameras, digital representations must be obtained. Images are acquired from the cameras, digitized, and stored as two-dimensional arrays for processing. The images are of size 480 rows by 512 pixels in each row. Each pixel is represented by one byte (8 bits) of information indicating the intensity level of the pixel. A value of 0 represents the darkest intensity, while a value of 255 represents the brightest pixel intensity. The following is the algorithm used for the pre-processing.

begin
  while (there is an object) *do*
  begin
    Threshold the image to form a binary image of parts. The thresholding operation will separate the objects from the background.

    Process the binary image to separate and label the different objects.

    Identify each object and calculate its position and orientation.

  end
end

The outcome of the pre-processing operation is the creation of a feature vector to be fed to an ART1 network for classifications. The classification process results in the creation of a database of parts in inventory along with each part's location and orientation. The next sections discuss the pre-processing operations in more detail.

4.1.2. *Thresholding*

The purpose of thresholding is to separate the objects from the background. The result of thresholding is to create a two level image, called a binary image, in which the background is at one level (0) and the objects are at the other binary level (1). The following formula illustrates the operation.

$$f(x,y)=\begin{cases}1 & \text{if } \xi_{xy}\geqslant T\\ 0 & \textit{otherwise}\end{cases} \tag{7}$$

where $f(x,y)$ is the resulting binary image, ξ_{xy} is the original grey level image, and T is the thresholding level.

4.1.3. *Object separation*

With the objects separated from the background the next step is to differentiate objects in the image. Two approaches are commonly used to separate and label different objects. These approaches are the *recursive* and the *sequential* labelling algorithm (Horn 1986).

The sequential algorithm was selected for labelling since it avoids recursive calls, thus requiring less system memory, and provided a straightforward procedure for object labelling. The sequential algorithm scans the image sequentially and labels the components in one pass. In this algorithm, a pixel is analysed along with the pixel's immediate neighbours located directly to the left, top and upper left (diagonal).

begin
V_A = value of the pixel under consideration for labelling;
V_B = value of the pixel on top of A pixel;
V_C = value of the pixel to the left of A pixel;
V_D = value of the pixel to upper left corner of the A pixel;

while (there is a pixel) *do*
begin
if $V_A = 0$ then *continue*
if $V_A = 1$ *and* V_D is *labelled then* $V_A := V_D$
if V_B is *labelled then* $V_A := V_B$
if V_C is *labelled then* $V_A := V_C$
if $V_B = V_C$ *and* both are *labelled then* $V_A := V_B$ (or V_C)
if $VB \neq VC$ and both are *labelled* then VA: = VB or VA: = VC
(note the two labels are equivalent and correspond to same object)
end
end

The above algorithm was successful in labelling the objects regardless of their position and orientation.

4.1.4. *Position and orientation calculations*

With all of separate components labelled in the image, calculations are performed to compute the area, the centre of mass, the angle of the principal axis, and the length along the principal axis. The following formula, taken from Gonzalez (1992), illustrates the moment calculations being performed on each object.

$$M_{ij} = \int_{-00}^{+00} \int_{-00}^{+00} x^j y^k f(x, y)\, dx\, dy \tag{8}$$

where j and k take on all non-zero values, and $f(x, y)$ is the binary image. There are two first order moments M_{10} and M_{01}. The moments along with the area are used to calculate the centre of gravity (centroid) for each object. The centre of gravity (X_{cog}, Y_{cog}) is calculated with the following formulas.

$$X_{cog} = \frac{M_{10}}{M_{00}} \quad Y_{cog} = \frac{M_{01}}{M_{00}} \tag{9}$$

The centre of gravity is used as the location of each object in camera coordinates. The central moments are used to compute the orientation of each object. The following formula, again taken from Gonzalez (1992), is used to perform this computation.

$$\mu_{ij} = \int_{-00}^{+00} \int_{-00}^{+00} (x - x_{cog})^j (y - y_{cog})^k f(x, y)\, dx\, dy \tag{10}$$

In designating orientation, the angle of the principal axis is calculated with the following formula.

$$\tan 2\theta = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{11}$$

Once the angle of orientation is computed, the length along the principal axis can be calculated. The length is calculated by starting at the centroid and moving along the principal axis in one direction until the edge of the object is found. Then we start again at the object's centroid and move in the other direction along the principal axis until the edge of the object is detected. The distance between these two end points is the length along the principal axis.

In creating the feature vector, the length and area are broken up into several ranges. Each range is represented by a couple of bits in the feature vector. When the area and length along the principal axis are calculated, the bits corresponding to the range the length and area fall into are set to 1, with the rest of the bits set to zero. Since all of the standard parts are rectangular in shape, these two features will allow for the recognition of all parts. The features can be quickly calculated for each object, since most of the calculations have been performed by the preprocessor. The feature vector is then passed to the neural network classifier so each object is recognized as a certain part.

4.1.5. *Object classification through ART1*

With a feature vector calculated, the parts are classified with an ART1 neural network classifier. The network performs a nearest-neighbour classification of the feature vector. When an object's feature vector is presented to the network, the network compares this feature vector to the patterns stored in the network. If the input is sufficiently close to a stored pattern, a match is made and the part recognized from the previous exposure. If no pattern is found which closely matches the input pattern, a new pattern class is created and the pattern stored as the new class.

When a new pattern class is created, the system consults the operator to provide the name of the new part. If the object is classified as a previously seen part the system will add the part to the current inventory list and display it in that class colour with its name. Figure 10 shows an image where the objects have been classified into part categories. Each object is labelled with the part it was identified as.

The coarseness of the classification process is controlled by the setting of the vigilance factor. The higher the setting, the finer the classification process. The tradeoffs to consider in setting the vigilance factor is to set it high enough so that different parts will be classified into different categories, but low enough such that noisy representations of objects will be correctly classified. In testing the system, a vigilance factor setting of 0·85 provided an adequate level of part discrimination and handling of system noise.

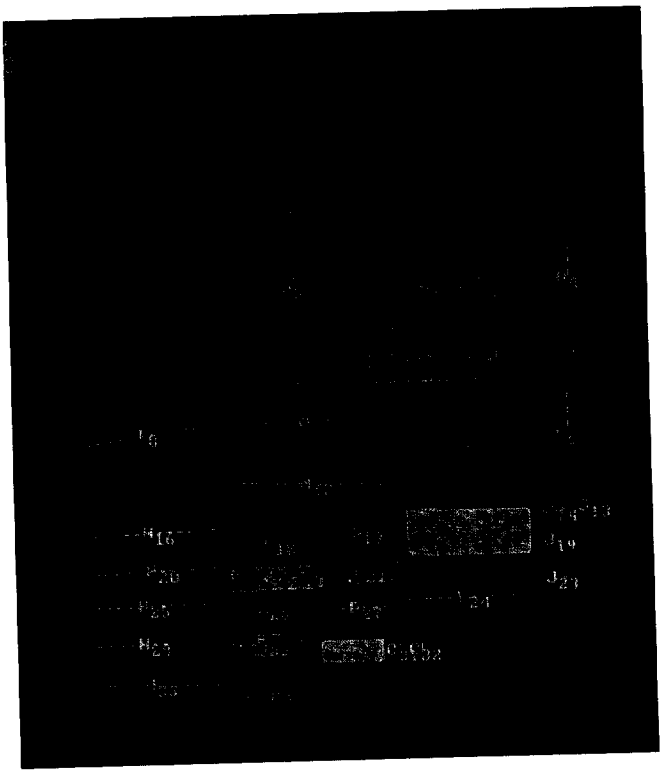

Figure 10. Objects have been recognized and labelled by utilizing ART1 neural network.

The ART1 neural network classifier allows for on-line part recognition, without the requirement of a training phase. The capability of a self-organizing part recognition system is a significant impact for manufacturing. New parts can be added to the system without having to modify existing software or stopping production for a training phase. As long as the capacity of the neural network is not exceeded, the system can classify new parts from the initial exposure of the part.

4.2. *ANN based system calibration*

The second area utilizing neural network technology is that of system calibration. System calibration is an important step in any machine vision application. Before obtaining accurate location information, the cameras of the vision system must be calibrated to a global coordinate system. Since the objects are going to be moved by a robotics arm, it is convenient to use the robotics arm's coordinate system as the global coordinate system. The calibration procedure establishes the transformation of points from image coordinates to robotics' arm coordinates.

The purpose of developing the ANN based calibration system was to investigate whether the calibration network could be trained with sufficient accuracy to transform points from camera to robotics arm coordinates. The setup was to be very straightforward and easily automated and any fixturing or test objects needed in the calibration were to be easily produced.

The main advantage of the neural network calibration over conventional techniques is it does not require an elaborate model of the imaging system. Another

advantage, that the proposed method offers, is that no *a priori* knowledge of camera placement, the effective focal length of the camera lens, or the pixel spacing of the camera's pixel array is required. Other techniques estimate these parameters in order to establish the relationship between the camera and robot coordinates. While the neural network does not estimate these parameters explicitly, the parameters are learned as the network trains on how to transform image coordinates into robot coordinates.

A possible disadvantage the ANN based calibration has, is its moderate accuracy of calibration. While the network was accurate enough for this application, it does not obtain the accuracy of less than a millimetre claimed by the other calibration techniques. A second disadvantage with the neural network approach is that it cannot provide the real-time calibration that can be performed by other approaches. The training of the neural network needs to be performed off-line. This disadvantage is not an important limitation, since the camera remains fixed in a single position, requiring calibration only if the camera is moved relative to the robotics arm.

4.2.1. *Machine vision/robotics arm calibration*

The cameras to be calibrated are located approximately 70 inches above the work table of the robotics arm. The ANN based calibration procedure is designed to learn the relationship between the camera coordinate system and the robotics arm coordinate system. The field of view of the system's two cameras are approximately $2 \times 1{\cdot}5$ ft.

The first step in the calibration procedure was to place objects in known locations within the field of view of the camera. In this setup, twenty white plexiglass blocks, 0·5 inch square, were picked up by the robotics arm and placed evenly throughout the field of view of the camera. Points that were within the field of view of the camera, but outside the robotics arm work area were ignored. Figure 11 illustrates the setup from a top view looking down on the robotics arm work space. From this setup a training set is obtained. The training set consists of vectors created of from each calibration object's coordinates. The vector is composed of an object's robot coordinates (obtained from where the objects were placed) and image coordinates (obtained from the centroid calculations).

The next step in the calibration procedure is to train the neural network to learn the transformation between image (camera) coordinates and robot coordinates. The

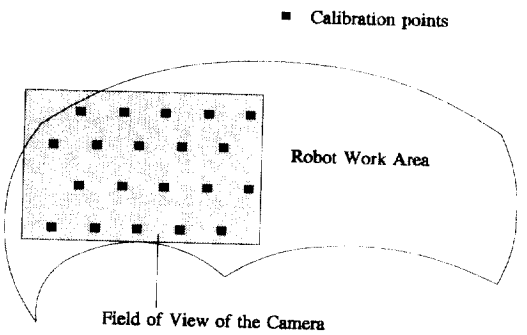

Figure 11. Robot arm area and camera field of view.

network is a four layer network consisting of one input layer, two hidden layers, and one output layer. The input layer has two input neurons (processing elements) corresponding to the normalized x and y object's centroid from the camera. There are also two neurons in the output layer corresponding to the object's centroid robot coordinates. Each of the hidden layers contain twenty neurons. Initial testing of the network indicated that a quicker convergence could be obtained with two hidden layers rather than with a single hidden layer.

During the training sequence, each vector pair (image coordinate/robot coordinate) was presented to the network and the network weights were modified appropriately. Training continued until an acceptable accuracy was obtained. During each pass a network error was calculated for the entire training set. The network error was the accumulated sum of the difference between the desired and actual output. The following formula illustrates the error calculation:

$$Net\ error = \sum_{i=1}^{i=20} |output_{\text{desired}} - output_{\text{Network}}| \tag{12}$$

The network was trained until the accumulated error for the entire twenty vector training set was 0·0049. The initial training of the network took place over several days in order to obtain the accuracy mentioned above. But after the initial training session, if the camera was moved slightly ($<1{\cdot}00$ inch) the system could be recalibrated to this accuracy with only an additional hour of training. So once the initial calibration network has been trained, any recalibration can be done relatively quickly.

While the training time might seem excessive it is possible to use more sophisticated training procedures which could substantially decrease the training time. It is also important to note that during operation, the only time required to perform the conversion between machine vision coordinates to robotics arm coordinates is the propagation delay through the neural network.

In evaluating the ANN approach, the calibration was substantially easier to setup and perform compared to a conventional approach where a sophisticated mathematical model of the imaging system must be created, coded, and validated. Then parameter estimation must be done through images acquired of calibration objects. The overall calibration process was much quicker with the ANN implementation even with the long training time.

In testing the ANN based calibration, the calibration objects were set in locations between the training set locations. Images were taken and the centroids of the calibration objects calculated. These centroids were then converted using the ANN calibration network to produce robotics arm coordinates. These coordinates were then compared to the actual points at which the calibration objects were placed. The results of the ANN based calibration were encouraging and the average error (neural network output to actual location) was less than 0·04 inch with the maximum error being 0·056 inch.

5. Robot controller subsystem

The robot controller subsystem constitutes the last part of the IPS. It is composed of two components, an IBM 7335 robot used for packaging and a host computer to control it. The inputs to the subsystem are the pick and place coordinates of each of a chair's parts needed for packaging. The pick coordinates are

obtained from the vision system after the image processing and object classification, as described in the previous sections. The place coordinates are determined by the design module. Figure 12 illustrates the functionality of the robot controller subsystem.

The sequence-generator portion of the subsystem reads the information supplied by the vision subsystem, and the design system, i.e. the pick and place coordinates, and determines a valid sequence of pick and place motions. The second step involves on-line programming of the robotics arm in AML (A Manufacturing Language). AML is a high level robot programming language. The on-line program generator, which is an object oriented software written in C++, generates the AML code with correct syntax in ASCII format. Thus the output of the second step is an AML code consisting of a sequence of pick and place function calls with unique pick and place coordinates in each call. The third step involves compiling the AML code into the machine language of the robot. This is performed by another C++ member function which invokes the AML compiler and provides as input the AML program. The result is another file with the machine instructions ready to be transmitted to the robot controller.

The fourth step is executed by a robot interface program. This program is responsible for sending the file consisting of a series of machine instructions to the robot controller by using an RS-232 communication protocol. This is essentially a serial communication and involves a good amount of handshaking for ensuring error-free transmission.

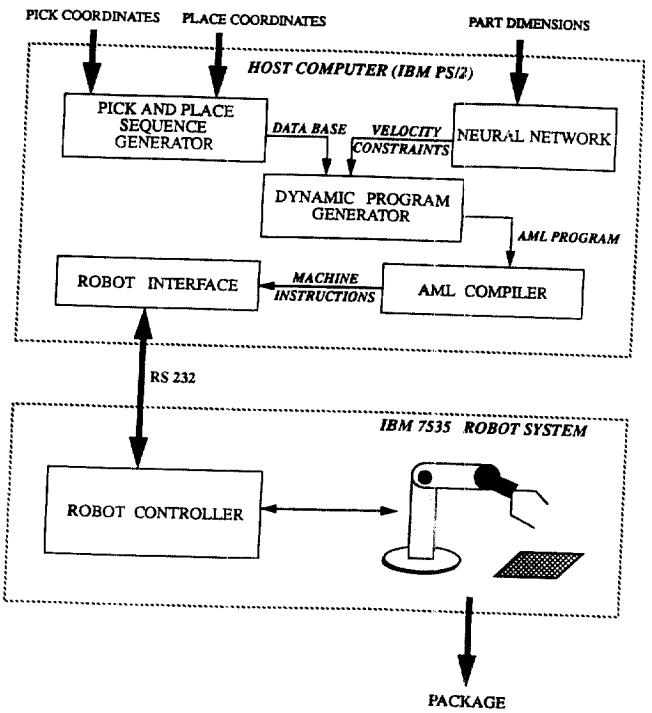

Figure 12. Functional description of robot controller module.

At the beginning of the fifth step, the robot controller has in its memory a sequence of machine instructions to control the robotics arm. The controller reads these instructions sequentially and decodes them into the necessary electrical signals to make the robotics arm move and perform an operation. After the packaging is complete, the robotics arm moves back to its home position and is ready for the next operation. The result of this step is a packaged product which is ready to ship.

5.1. *ANN based robot velocity control*

A backpropagation neural network has been used to control the velocity constraints for each part based on the part's length and width. The network architecture consist of three layers (one input, one hidden and one output layer). The input layer consists of eight neurons, the hidden layer consists of 15 neurons, and two neurons are used in the output layer.

The training set is composed of dimensions of sample parts and the recommended speeds for each. Table 1 shows a list of dimensions (inch) and the recommended speeds. The length and width parameters together form the eight-bit input vector to the network and the two-bit speed parameter represents the desired output of the network. The speed is classified into four classes, namely, *very slow* (0), *slow* (1), *moderate* (2) and *fast* (3).

Similar to calibration procedure, training of the network continued until an acceptable accuracy was obtained.

6. Implementation

The system has been implemented based on the ideas discussed here. The intelligent design retrieval and packaging system composed of four IBM PCs networked through IBM token-ring (Fig. 1). Table 2 shows the list of the computers used in the system.

The ADRS is divided into two separate sections, front-end and back-end. The front-end provides the user interface, and the back end consists of fuzzy associative memory. The front end itself is divided into two sections, pre-processor and post-processor. The pre-processor communicates with the designer and assists them in setting up the functional requirements and design constraints. The post-processor on the other hand converts the recall vector of the FAM to a design solution. The back-end does the actual mapping of input requirements to the design solutions.

As previous described, the IPS is divided into two subsystems, the machine vision and robot arm control subsystems. These subsystems were run separately on the vision node and packaging node, respectively. The operations were coordinated through the monitor program which is briefly described below.

Length	Width	Feature vector (input vector)	Speed	Desired output vector
6=(0 1 1 0)	2=(0 0 1 0)	0 1 1 0 0 0 1 0	Very slow	0 0
8=(1 0 0 0)	6=(0 1 1 0)	1 0 0 0 0 1 1 0	Slow	0 1
4=(0 1 0 0)	2=(0 0 1 0)	0 1 0 0 0 0 1 0	Moderate	1 0
4=(0 1 0 0)	4=(0 1 0 0)	0 1 0 0 0 1 0 0	Fast	1 1

Table 1. The representative of component dimensions and their recommended speeds.

Function	Computer type
Automatic design retrieval system (ADRS)	PS/2 model 80
Machine vision	PS/2 model 30
Network server	IBM 7552
Robot controller	IBM AT

Table 2. List of the computers.

The network server provides the shared resources for the system. The server is set up to share a hard disk as well as a virtual disk. The system monitor program runs on this system. The monitor checks the status of the system and facilitates communications between the different processes running concurrently. Table 3 shows the various modules of the system and the programming language used for their implementation.

7. Summary and conclusion

In this paper the application of neural networks in design and manufacturing has been examined. The system has incorporated three neural network paradigms such as fuzzy associative memory (FAM), Backpropagation (BP), and adaptive resonance theory (ART1) to achieve its task. FAM is utilized to automate the design retrieval or map the marketing characteristics to pre-designed structures. ART1 has been utilized for object identification in the machine vision subsystem. A backpropagation neural network has been used for camera calibration, another backpropagation neural network is used to control the speed of the robot arm based on the size of a given part. This unique application of neural networks in design and manufacturing can be extended to more sophisticated tools for concurrent engineering and a future automated factory.

Acknowledgement

We wish to acknowledge the programming efforts of Mr Mahesh Vellanki. This work was supported in part by AT&T, and in part by IBM, and in part by Intelligent Systems Center, University of Missouri-Rolla.

Module	Computer language
ADRS pre-processor	LISP
ADRS post-processor	LISP
Fuzzy associative memory	C++
Vision pre-processing	C++
Backpropagation neural network	C++
Image classification ART1	C++
Robot controller	C++
Monitor	C++

Table 3. System modules and implementation languages.

References

BAHRAMI, A., DAGLI, C., and MODARRESS, B., 1991, Fuzzy associative memory in conceptual design. *Proceedings of the IEEE International Joint Conference on Neural Networks*, **I**, 183–188, Seattle, WA.

BAHRAMI, A., and DAGLI, C. H., 1993, From fuzzy input requirements to crisp design. *International Journal of Advanced Manufacturing Technology*, **8**(1).

BARNARD, E., and CASASENT, D., 1989, Image processing for image understanding with neural nets. *Proceedings of the IEEE International Joint Conference on Neural Networks* **I**, 111–115, Piscataway, NJ.

CARPENTER, G., and GROSSBERG, S., 1986, Adaptive resonance theory: stable self-organization of neural recognition codes in response to arbitrary lists of input patterns. *Proceedings of the 8th Annual Conference of the Cognitive Science Society* (Hillsdale NJ: Lawrence Erlbaum), pp. 45–62.

DAGLI, C. H., and HUGGAHALLI, R., 1991, Neural network approach to group technology. In *Knowledge-Based Systems and Neural Networks: Techniques and Applications*, R. Sharda, J. Y. Cheung and N. J. Cochran, (eds), (New York: Elsevier), pp. 213–228.

GONZALEZ, R. G., and WOODS, R. E., 1992, *Digital Image Processing* (Reading: Addison-Wesley).

HORN, B., 1986, *Robot Vision* (Cambridge: MIT).

KOSKO, B., 1987, Fuzzy associative memories. In *Fuzzy Expert Systems*, A. Kandel (ed.) (Reading: Addison-Wesley).

RUMELHART, D., HINTON, G., and WILLIAMS, R., 1986, Learning representations by backpropagating errors. *Nature*, **323**, 533–536.

SHASHIDHAR K., and SURESH, N., 1991, A neural network system for shaped-based classification and coding of parts. *International Journal of Production Research*, **29**, (9), 1771–1784.

SIMPSON, P., 1990, *Artificial Neural Systems* (New York: Pergamon).

SUH, N. P., 1990, *The Principles of Design* (Cambridge: Oxford University Press, Inc.).

WERBOS, P., 1974, *Beyond regression: new tools for the prediction and analysis in the behavioural sciences*. PhD thesis, Harvard University.