PERGAMON

Computers & Industrial Engineering 36 (1999) 697–718

computers & industrial engineering

www.elsevier.com/locate/dsw

Automatic classification of block-shaped parts based on their 2D projections

J.-H. Chuang[a,*], P.-H. Wang[b], M.-C. Wu[b]

[a]*Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan*
[b]*Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan*

Abstract

This paper presents a classification scheme for 3D block-shaped parts. A part is block-shaped if the contours of its orthographic projections are all rectangles. A block-shaped part is classified based on its partitioned view-contours, which are the result of partitioning the contours of its orthographic projections by visible or invisible projected line segments. The regions and their adjacency in a partitioned view-contour are first converted to a graph, then to a reference tree, and finally to a vector form, with which a back-propagation neural network classifier can be trained and applied. The proposed back-propagation neural network classifier is in a cascaded structure and has advantages that each network can be limited to a small size and trained independently. Based on the classification results on their partitioned view-contours, parts are grouped into families that can be in one of the three levels of similarity. Extensive empirical tests have been performed; the pros and cons of the approach are also investigated.

Keywords: Part classification; Block-shaped parts; Neural networks

1. Introduction

Group technology (GT) is a philosophy which advocates that workpieces with similar characteristics should be grouped into a family in order to facilitate design and manufacturing processes. Traditional implementation of GT requires establishing a workpiece coding scheme with which workpieces can be coded and then grouped. Numerous GT coding schemes have

* Corresponding author. Tel: +886-3-5731829; fax: +886-3-5724176.

0360-8352/99/$ - see front matter
PII: S0360-8352(99)00160-6

been developed and applied in practical applications. Workpieces are normally coded manually from their engineering drawings. Hence the process is time-consuming and error-prone.

Most recent research therefore aimed at automating the task of workpiece coding and classification. Conceptually, most proposed approaches intended to recognize form features from a CAD file of workpieces, and then code or classify the workpieces based on the recognized features [1–7,9]. Form features are generally predefined shapes, such as holes, steps, and slots, which are essentially local by nature and describe only the local shape characteristics of a workpiece.

The global shape or general appearance of a workpiece, however, is important information for classification, but is rarely addressed in most previous studies. To this end, an appropriate global shape descriptor for characterizing workpieces is needed. Such direction had been pursued and proven useful in parts design retrieval [10,11]. In their recent studies [11], a 3D workpiece is first modeled by the contours of its three orthographic projections; each contour is then characterized by a simplified linear skeleton, and finally classified and coded by a neural network classifier. With the coding results from three projections, a 3D workpiece can be classified into a part family. A massive experimental testing has been performed and revealed a satisfactory result. The major limitation of the approach is that only the contour information is considered in the classification. Parts which are different in shape but are identical in their projection contours might be classified as in a same family. For example, both block-shaped parts shown in Fig. 1 are distinct in their general appearance, but are in the same family since their three projected contours are all rectangles. It is apparent that such an approach is deficient in handling block-shaped parts and, other than contour itself, we need more information within the contour.

The shape descriptor proposed in this paper considers not only the projected view contours but also geometric and topological information within them. Such interior information is built based on the visible and invisible line segments from three orthographic views. These visible and invisible line segments subdivide each corresponding view contour into regions. Regions within a view contour are modeled by a graph with the nodes representing the regions and the arcs for adjacency relations among regions. Such a graph is first converted into a tree and then into a vector representation that can be fed as an input to the proposed neural network classifier. The proposed back-propagation neural network classifier is in a cascaded structure and has advantages that each network can be limited to a small size and trained independently. Parts are finally classified into families based on their classification results from three projected views. Part families are assigned to be in one of the three levels of similarity. Such a hierarchical grouping of parts would be very valuable for the activity of design-by-retrieval.

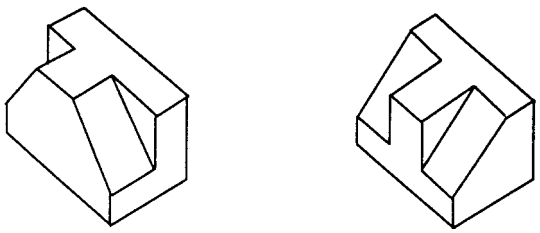


Fig. 1. Parts with identical projected contours.

Subsequent sections of this paper are organized as follows. Section 2 reviews the relevant literature. Section 3 gives an overview of the proposed classification scheme. Section 4 discusses the partitioned view-contours and their graph representation. Section 5 describes the conversion process from a graph to a reference tree. Section 6 explains how a reference tree is converted to a vector form. Section 7 addresses a cascaded BP neural network and how it is trained and used to classify parts. Section 7 explains how workpieces are grouped into families. Section 8 depicts the experiment result on a set of 24 parts. Concluding remarks are given in the last section.

2. Relevant literature

Numerous previous studies had applied neural network techniques to solve the workpiece classification problem. Kaparthi and Suresh [6] first applied a neural network model to automatically determine the *Opitz code* for rotational parts. Each part is modeled as a bitmap — a matrix of binary pixels. A back-propagation (BP) neural network model [8] is adopted, in which each node of the network denotes a pixel of the bitmap and the output node represents a numerical value for a particular digit of the Opitz code. Although this work was a milestone in the automatic coding of workpieces, it may result in an inappropriate GT code whenever the part is positionally translated, rotated, or scaled.

Kao and Moon in their early work [5] also applied the BP neural network to the workpiece classification problem. A customized coding system based on part features is adopted to code the workpieces. In this feature-based coding system, the code of a workpiece involves only binary digits, that is, each digit denotes the existence or absence of a particular feature. The input node of the BP network represents the workpiece code, and the output node models the intended groups of workpieces. Such a network can automatically classify similar workpieces into groups once they have been coded. This work lately had been modified by using another neural network model-ART (Adaptive Resonance Theory), with which the number of part families can be increased, that is, new part family can be formed whenever needed. Similar approaches that based on a feature-based CAD system are proposed in [7].

Chakraborty and Roy [2] applied *Kohonent's self-organizing feature maps* to cluster parts into families which are then used as the training templates to construct a workpiece classifier based on the BP neural network model. As a consequence, each part family needs to build up a neural network classifier that determines if the input workpiece is a member of the family. The classification capability of this approach is in general satisfactory although in real applications it may be deficient since a large number of classifiers need to be constructed.

In summary, most previous studies are either implicitly or explicitly based on a feature-based coding system. Such a coding system is designed to encode a workpiece by characterizing its form features, which are generally predefined as steps, grooves, or holes, and describes essentially only the local shape information of a workpiece. Approaches that take global shape information into account are proposed in [10,11]. Simplified linear skeletons for three orthographic projections serve to represent the global shape information of a workpiece. Such skeletons are converted into vector forms as inputs to a BP neural network for the classification.

3. The proposed part classification scheme

We assume that the input 3D workpiece is of block-shaped, which means that a local coordinate system can be defined such that the projected contour or silhouette of the workpiece's boundary along each axis is a rectangle. The projected contour or silhouette of the workpiece's boundary along each axis is denoted as *view contour*. In the proposed scheme, parts are classified based on the geometric and topological information obtained by subdividing each view contour using visible as well as invisible line segments from three orthographic views. We denoted such a subdivided view contour as *partitioned view-contour*. This stage proceeds in the following five steps:

1. All edge segments of the workpiece are projected along front/back, left/right, and top/bottom views. Such projection results in nine figures, six of which are from the projected contour lines and visible edge segments while the rest three consist of contour lines as well as invisible line segments. The projected visible or invisible edge segments partition each contour into regions.
2. Each partitioned view-contour is represented by a graph with the nodes representing the regions and the arcs for adjacency relations among regions. Each graph node is associated with a *representative ring code*.
3. Each graph is first converted into a reference tree and then a vector form, which will be the input to the BP neural network classifier.
4. Construct a cascaded BP neural networks, each of which will be trained by a set of manually selected representative partitioned view-contours. All partitioned view-contours of each part will be classified by the trained neural network: When a partitioned view-contour cannot be successfully classified, it will be put into a X-set. When the size of the X-set exceeds a prespecified number, some templates in the X-set are used to train another neural network. Repeating such a training process, a cascaded neural network classifier will be constructed.
5. After all partitioned view-contours are successfully classified, each input part is grouped into a part family, which can be in one of three levels of similarity.

Major computational components are detailed in the following sections.

4. A graph representation for partitioned view-contours

4.1. Nine partitioned view-contours

Other than view contours, to extract more shape information we take into account the projection of all other edges segments on the workpiece's boundary. Since these edge segments can be either visible or invisible, we have two categories of projected views: one for visible edge segments and the other for invisible ones. It is apparent that the first category contains six views from $+x$, $-x$, $+y$, $-y$, $+z$, and $-z$ while the second category has only three views from x, y, and z. In the figures, visible edge segments are denoted as solid line segments while invisible edge segments as dashed lines. These visible and invisible line segments partition a

view contour into regions with solid and dashed line segments, respectively. See Fig. 2 for an example.

4.2. Graph representation

The partitioned view-contour can be naturally represented as a graph, in which the nodes representing the regions and the arcs for the adjacency relations among regions. Since regions can be different in geometric shape, a *representative ring code* is derived and associated with each region. Such an coding information will later be taken into account in the classification. The graph arc can be directed or undirected, depending on the adjacency relation between two corresponding regions. A neighboring adjacency is represented by an undirected arc while a

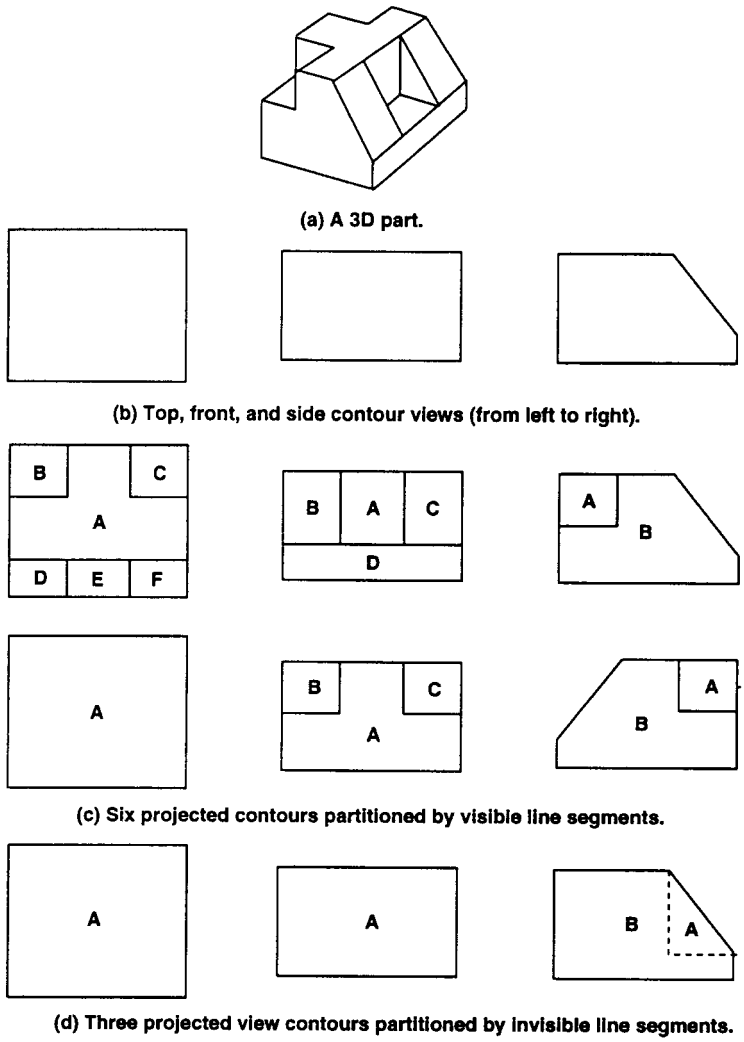


Fig. 2. Nine projected partitioned contours for a workpiece.

full-containment relation is denoted by a directed arc; see Fig. 3a and b. Moreover, each arc is associated with a weight representing the number of adjacent edges between two neighboring regions. For two regions with full-containment relation, their arc will be assigned weight 0 since no adjacent edge exists. Hence the graph here is a weighted graph and can be directed, undirected, or mix of the two; as shown in Fig. 3.

4.2.1. Ring code for a region

A ring code is a cyclic string of digits, each of which representing an oriented edge segment of the region boundary. To appropriately derive the ring code for a polygonal region, we proposed a two-layer octal coding system as shown in Fig. 4. Eight oriented vectors representing eight possible edge orientations. Each oriented vector is associated with two layers of digits, with the first layer for edges that have no adjacent region and the second for edges adjacent to some regions. The first layer of digits begins from 1 to 8 while the second from 9 to 16.

To derive a ring code for a region, we simply traverse the region's boundary clockwisely starting from a randomly selected edge. In determining the code for a particular edge, we always consider its preceding edge as oriented in the $+y$-axis and then obtain the code for this edge according to the traversing direction. Since we have chosen the octal coding system,

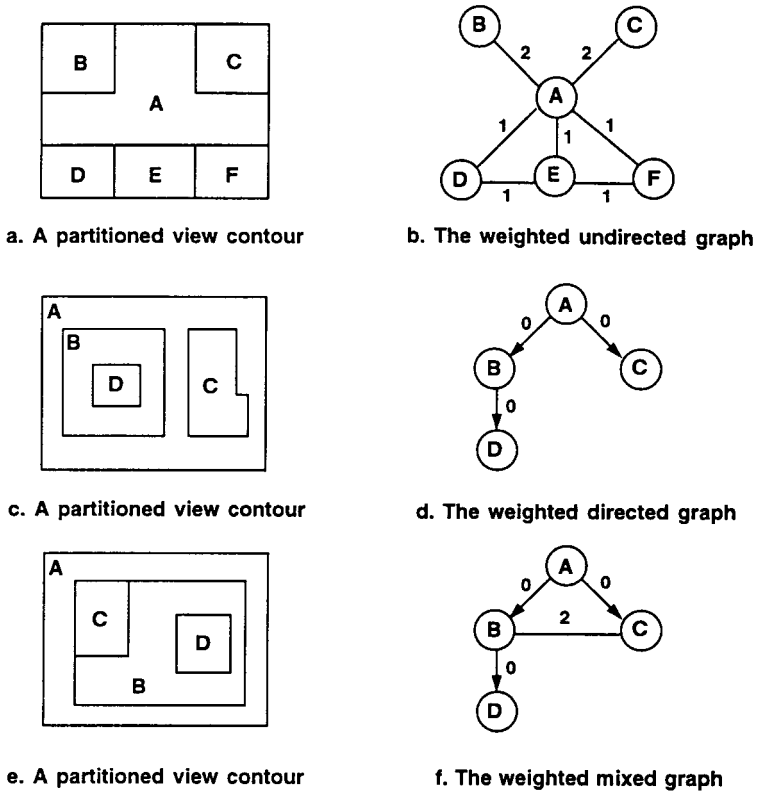


Fig. 3. Graphs for partitioned view contours.

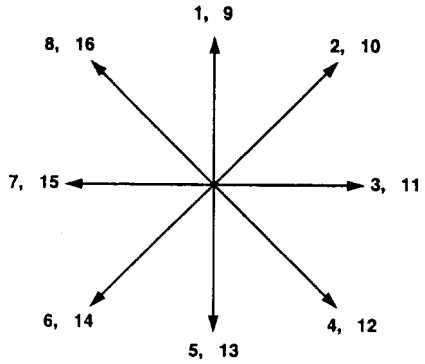


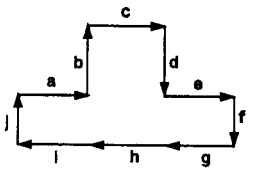
Fig. 4. The two-layer octal coding system.

edge's code will be determined by the vector whose direction is most close to the edge orientation. Take region A shown in Fig. 3a as an example. Suppose edge a in Fig. 5a is chosen as the starting edge, we first take it as oriented in $+y$-axis and determine the code for its proceeding edge b, which is 15. By taking b as oriented in $+y$-axis again, we obtain code 3 for edge c. Repeating such traversing and coding process, we finally derive a ring code (15-3-11-15-3-11-9-9-3-11) for region A. Similarly, ring codes for regions B and C are (3-11-11-3) and (3-3-11-11), respectively.

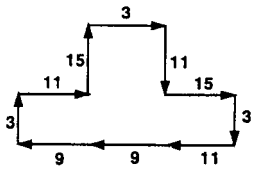
Such a coding system is orientation and uniform-scaling independent. That is, a region will have a same ring code no matter how it is rotated and uniformly scaled. This fact will facilitate the capability of the proposed classifier.

4.2.2. Representative ring code

The ring code derived in the previous section basically is cyclic in nature and not unique as a string for each region, that is, it varies depending on which is the starting edge. To facilitate the classification, a *representative ring code* that is unique as a string for each region is



a. Each edge is oriented as in +y–axis.



b. Derive the ring code starting from edge a.

Fig. 5. Derivation of a ring code.

necessary. Since a region has only one cyclic ring code, we need a way to identify which code is the leading one. Three heuristic rues that should be applied in sequence are given here to obtain such a leading code for forming the representative ring code:

1. The code that leads the longest string of identical codes in the derived ring code. For example, the representative ring code of (1-1-2-2-2-3-3-3-3) starts from the leading 3 in the longest string of codes 3-3-3-3. So the representative ring code is (3-3-3-3-1-1-2-2-2).
2. If there are more than one longest string of codes, among those strings the leading code of the string with the highest digit value. For example, the representative ring code of (1-1-4-4-4-4-3-3-3-3) starts from the leading 4 in the string of codes 4-4-4-4. So the representative ring code is (4-4-4-4-3-3-3-3-1-1).
3. If rules 1 and 2 fail to give the representative ring code, more than one longest string of codes have the same digit value. For such a case, the longest string of codes which has the largest digit immediately left to itself wins. Such a competition might be repeated until one string of codes wins. For example, the two longest strings of (1-2-4-3-3-3-4-3-3-3) are both 3-3-3 and the digits immediately left to them are the same. By repeated application of rule 3, we obtain the representative ring code (3-3-3-1-2-4-3-3-3-4).

For region *A* shown in Fig. 3a, its complete graph representation with a representative ring code associated with each node is shown in Fig. 6.

5. Converting graphs into reference trees

As stated in Section 3, the graph representation for a partitioned view-contour should be converted into a vector before a BP neural network can be applied. In such a conversion the graph is first converted into a reference tree and then into a vector. This section addresses the first part.

5.1. Structure of reference trees

Suppose each region in the partitioned view-contour has at most $m + 1$ adjacent regions, the second level of the reference tree will have $m + 1$ child nodes. Each node on level two or higher will have m child nodes since its parent already counts one. So the total number of nodes for a tree of level n is

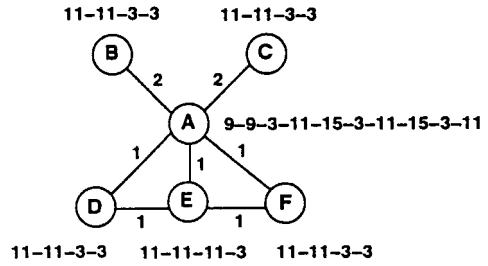


Fig. 6. The graph representation with representative ring code.

$$1 + (m+1) + (m+1)m^1 + (m+1)m^2 + \cdots + (m+1)m^{n-2} = 1 + (m+1)(1 + m^1 + m^2 + \cdots + m^{n-2}) = 1 + (m+1)(m^{n-1} - 1)/(m-1)$$

To preserve the adjacency relation on a graph, in addition to the parent–child links, tree nodes that are adjacent in the graph but not linked in the tree will be connected by auxiliary links. In figures, parent–child links will be denoted as solid links while the auxiliary links are dashed. Fig. 7 depicts the tree structure.

5.2. Converting undirected graphs into reference trees

To convert an undirected graph into a reference tree, we first associated each graph node with a weighting value and select the one with the highest weight as the tree root, and then assign graph nodes to tree nodes according to their associated weighting values.

The weighting values for graph nodes are determined in two stages. Each graph node is first assigned a priority order and from which the weighting value is then computed. The priority order for graph nodes is determined in sequence of the following three rules:

1. Suppose the representative ring code for the region denoted by a node is $(c_n - c_{n-1} - \ldots - c_2 - c_1)$. We compute

$$c = \sum_{i=1}^{n} i c_i$$

   The node with higher value of c gets higher priority order.
2. For nodes with the identical value of c, the one with longer representative ring code gets higher priority order.
3. For those nodes that are still even, the one with more adjacent nodes get higher priority order.

With the derived priority order, the weighting values are computed for nodes in the order of the derived priority. Let $w(i)$ denote the weighting value of the ith node in the decreasing priority order. The value of $w(i)$ is computed as

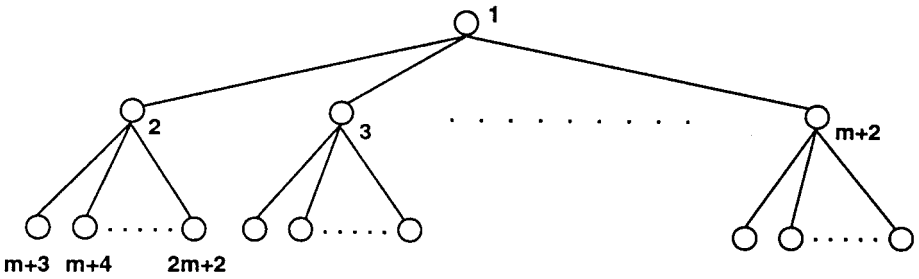


Fig. 7. Structure of the reference tree.

$$w(i) = \frac{(n-y)+(n-y-1)+\cdots+\left[n-y-(x+1)\right]}{x}$$

$$= \frac{nx-\left[y+(y+1)+\cdots+(y+x-1)\right]}{x} = n-y-\frac{x-1}{2}$$

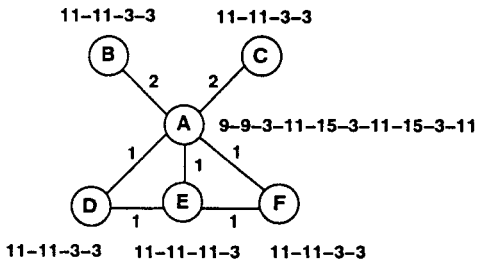
where n is the number of all nodes in the graph, y is the number of nodes whose weighting values have been computed, and x is the number of nodes that have the same priority order as node i.

After weighting values for all graph nodes are derived, the one with the highest weighting value is chosen as the tree root. Graph nodes that are immediately adjacent to the parent node are then assigned in the decreasing order of their weighting values to tree nodes from the left to the right on the next level. Nodes with the equal weighting value are broken even by the following two rules:

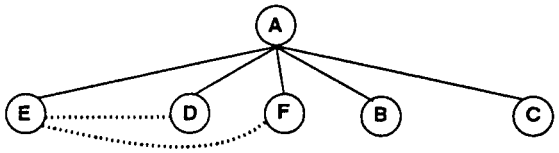
1. Among nodes with the equal weighting value, a node will be granted a higher privilege if it wins the competition on the highest weighting values of their immediately adjacent nodes. In this competition, any pair of nodes having been even will be broken based on their immediately adjacent nodes with the second highest weighting values. Such an even-broken process might be repeated.
2. For the remaining nodes that fail to apply rule 1, their next immediately adjacent nodes that are not yet considered will be compared as in rule 1. Rule 2 might be repeated until all nodes have been considered.

Remaining nodes that fail in applying rules 1 and 2 will have their order arbitrarily assigned.

The nodes in Fig. 8a are put in the decreasing priority order as $A > E > D=F > B=C$. The weighting value of A, $w(1)$, is $6-0-(1-1)/2=6$, and the weighting values of E, D, F, B,



a. The undirected graph for partitioned view contour in Fig. 3a.



b. The reference tree.

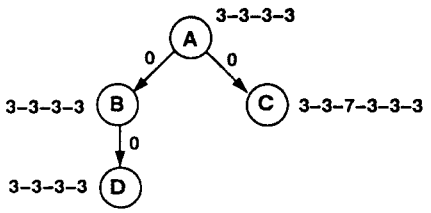
Fig. 8. Converting an undirected graph to the reference tree.

and C are 5, 3.5, 3.5, 1.5, and 1.5, respectively. So node A is chosen as the tree node. Nodes immediately adjacent to A are ordered as $E > D = F > B = C$ in their weighting values. These nodes are assigned to level 2 from the left to the right, in which the order for nodes in pairs (D, F) and (B, C) are arbitrarily assigned. Since nodes E and D are adjacent in the graph, they are connected by a dashed link. So are nodes E and F.

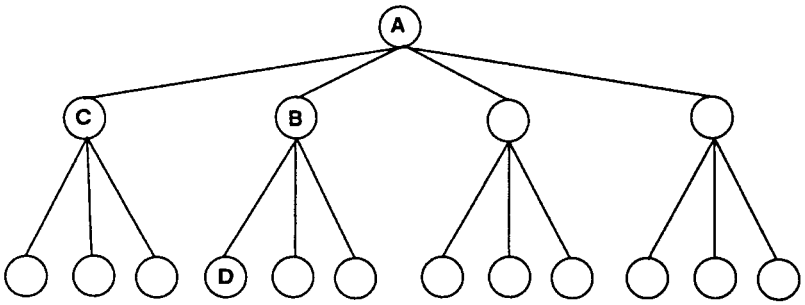
5.3. Converting directed graphs into reference trees

In converting a directed graph into tree, we first derive weighting value for each graph node as we do for the undirected graph. Since the directed graph used to present the partitioned contour is a tree-like graph, its root is naturally the root of the reference tree. The nodes adjacent to the root node are then put into the second level in the decreasing order of their weighting values. For each node k on the level l, its adjacent nodes that are not assigned yet are put to the level $l + 1$ in the decreasing order of their weighting values, but all are under node k. Nodes having the same weighting value are assigned to the tree nodes according to rules 1 and 2 described in the last subsection.

The nodes in Fig. 9a have the priority order $C > A = B = D$. The weighting values for C, A, B, and D are 4, 2, 2, and 2, respectively. Since node A is the toppest parent node in the graph, it is chosen as the tree root. Among those nodes immediately adjacent to A, node C has higher weight than B and hence nodes C and B are assigned to the second level from the left to the right. Node D is finally assigned to the left most child node of B on the third level.



a. The weighted directed graph for partitioned view contour in Fig. 3c.



b. The reference tree.

Fig. 9. Converting a directed graph to the reference tree.

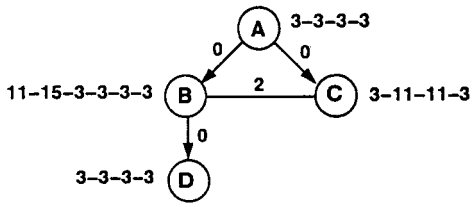
5.4. Converting mixed graphs into reference trees

Converting a mixed graph to its reference tree basically combines methods in previous two subsections. Weighting value is derived first for each graph node. The node with the highest weighting value will be the tree node unless it has ancestor nodes. If it has ancestor nodes, its highest ancestor node will be the root of the reference tree. Once the root is obtained, the other part of the graph is converted to the reference tree as what is done for the undirected graph.

The nodes in Fig. 10a have the priority order $B > C > A = D$. The weighting values for B, C, A, and D are 4, 3, 1.5, and 1.5, respectively. Node B has the highest weighting value and its highest ancestor is node A. So node A is chosen as the tree root. Among those nodes immediately adjacent to A, node B has higher weight than C and hence nodes B and C are assigned to the second level from the left to the right. Node D is finally assigned to the left most child node of B on the third level. Nodes B and C are connected by a dashed link since they are connected to each other in the graph by an undirected arc.

6. Converting reference trees to vector forms

Nodes of the derived reference tree are indexed by integers starting from 1. The indexing starts from the root on the toppest level down to the bottom and within each level from the left to the right. When each graph node has at most $m + 1$ adjacent nodes, the total number of the tree nodes is $1+(m+1)(m^{n-1}-1)/(m-1)$. To convert the reference tree to a vector form, we first associate each tree node with $6+m$ values in sequence and then form the vector



a. The weighted mixed graph for partitioned view contour in Fig. 3e.

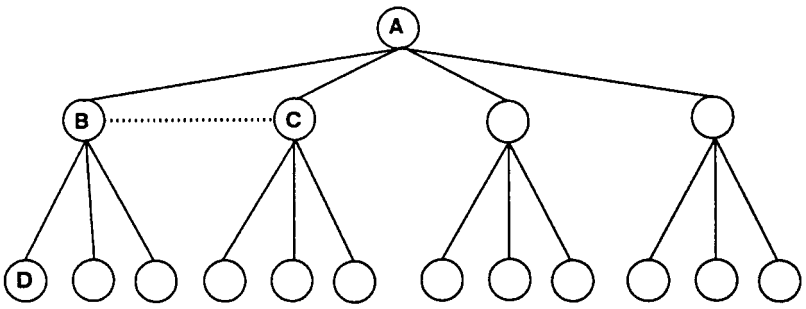


Fig. 10. Converting a mixed graph to the reference tree.

with the first $6+m$ elements for the tree root and the next $6+m$ elements for the tree node with index 2, and so on. Hence the resulting vector has $(6+m)[1+(m+1)(m^{n-1}-1)/(m-1)]$ elements. The $6+m$ values associated with each tree node are obtained based on heuristics that are come up after several experiments. The first four values derived from the representative ring code associated with the node. The fifth and sixth values are about the region associated with the node, while the rest values derived from the index values of relevant tree nodes.

For a tree node t_i, its $6+m$ values are as follows:

1. The length of the representative ring code associated with the node. For example, for the representative ring code $RRC=$(9-9-3-11-15-3-11-15-3-11) associated with the node A in Fig. 8, this value is 10.
2. The number of strings with consecutive identical codes in the representative ring code associated with the node. For RRC, this number is 9.
3. The number of different codes appearing in the representative ring code associated with the node. For RRC, this number is 4.
4. The average of codes appearing in the representative ring code associated with the node. For RRC, the average is 9.
5. The number of boundary edges of the region that are adjacent to other regions. For region A, this value is 7.
6. Values 6 up to $6+m$ are index values of those tree nodes that are connected to t_i by either solid links or dashed links. The values derived from dashed links are negated. After the values that derived from connected nodes are assigned, the rest values are assigned 0.

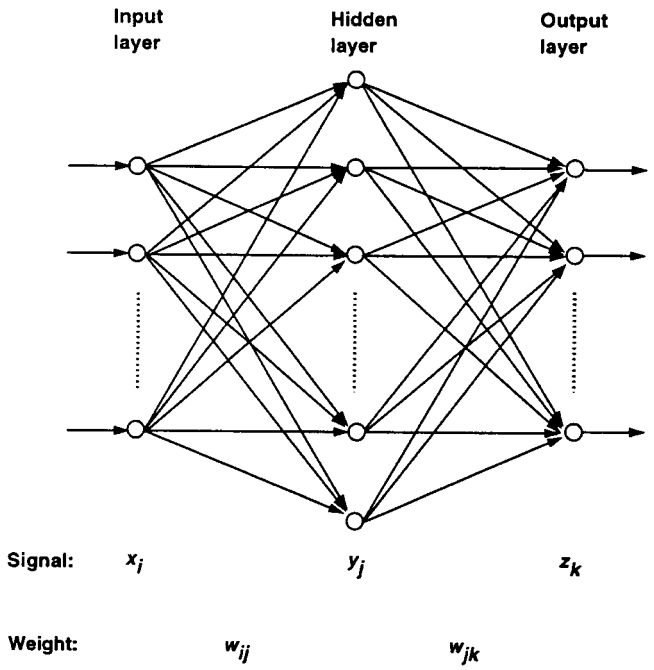


Fig. 11. The back-propagation neural network model.

7. Classification using a BP neural network

7.1. Back-propagation neural network

The back-propagation (BP) neural network model [8] is used to build up a classifier. As shown in Fig. 11, a typical BP neural network model is composed of three layers: an input layer, a hidden layer, and an output layer. In each layer, a node (or known as a *neuron*) stores a scalar as a signal value. Each neuron on a layer, if applicable, has an input link from each neuron on the previous layer. Each input link is weighted by a scalar indicating the strength of the relationship between two neurons.

On the hidden or the output layer, the lth neuron would receive a set of input signals, $U=(u_1, u_2, \ldots, u_n)$, from its previous layers and deliver an output signal v_l. The weighted sum of input signal values from the previous layer, $N_l = \sum_{k=1}^{n} u_k w_{kl}$, is derived for the lth neuron on the current layer and used to determine the output signal v_l by a transfer function F, which is a sigmoid function that constraints the value of each output signal between 0 and 1 [8]. That is,

$$v_l = F(N_l) = \frac{1}{1 + e^{-N_l}}$$

The value of each output signal v_l ranges from 0 to 1. Note that for neurons on the output layer, each output signal value is mapped to 0 or 1 by an appropriate thresholding scheme.

A BP neural network needs to be trained in order to build up its classification mechanism. That is, the weights on the input links should be obtained by an appropriate training process for a particular classification application. The first step in training a BP network is to select some representative input vectors and characterize their desired output vectors. Then, a training algorithm [8] is performed in order to obtain a set of weight values that are able to effectively map the selected input vectors in to their desired output vectors. With all the weights available, a trained network can be seen as a mapping function between the input and the output vector spaces, by which an input vector can be uniquely mapped to an output vector. The output vector will indicate the group to which a partitioned view-contour belongs. Such a vector mapping function therefore can serve as a classification mechanism.

7.2. Training of a cascaded BP network and part classification

The proposed part classifier is a cascaded BP networks, which are formed and trained in the process of part classification. It is detailed as follows:

1. Set $k = 1$ and $X=\emptyset$.
2. Select a set of representative parts from the input parts for the training of the first BP network, BP_1. Vectors are derived from part's partitioned view-contours and serve as the training templates for the network. The network BP_1 is trained as described previously.
3. For the next input part, vectors for nine partitioned view-contours are derived and fed in sequence to the cascade neural networks ($BP_1, BP_2, \ldots, BP_k$). If a partitioned view-contour can be classified by a BP_i in the cascaded neural network, it is finished; otherwise it is fed to

the next network BP_{i+1}. If the partitioned view-contour fails to be classified by all cascaded networks, it is appended to the set X.

4. When the size of X exceeds a predefined number, do the following:
   4.1. Some representative partitioned view-contours are chosen and removed from X to serve as the training templates for the new network BP_{k+1} in the cascaded structure. $k=k+1$.
   4.2. Set $X'=\emptyset$.
   4.3. Other elements are removed from set X and classified by BP_k. Those fail to be classified are appended to X'.
   4.4. $X=X'$.
   4.5. If $X\neq\emptyset$ then go to Step 4.

5. Go to Step 3.

Fig. 12 depicts the structure of the cascaded BP neural networks. The cascaded structure of networks is advantageous since each network can be trained independently and can be limited to a small size. If the cascaded structure is not used, a single neural network must be trained for all existing parts. Since such a single huge network should be periodically updated in order to classify new contours into appropriate classes and the training involved in each update should start from the scratch, the training and classification process is computationally expensive.

7.3. Formation of part families

The BP network classifier produces only one 1 for a particular neuron and 0 for all other neurons on the output layer. So there will be n_i distinct classes associated with network BP_i if it has n_i neurons on the output layer. All the distinct classes result from the cascaded network classifier can be coded by a 2D code. Hence each partitioned view contour of a part is classified by the BP network and assigned a 2D code. For a part, we concatenate three 2D

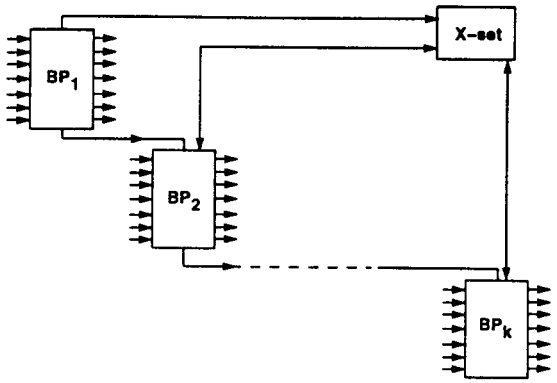


Fig. 12. The cascaded structure of BP neural networks.

codes that correspond to *x, y*, or *z* view to form the *directed code* for this particular view. The similarity level between two parts are determined by how many of their directed codes are *matched*. Two directed codes are matched if all their 2D codes are identical, regardless the order. For example, directed codes [1,2,5], [2,1,5], and [5,1,2] are matched. If all three directed codes are matched, the two parts have the highest level of similarity. On the other hand, two parts are completely irrelevant if all their directed codes are not matched. The more directed codes are matched, the more similar two parts are. Therefore, parts can be group hierarchically by setting the following three levels of grouping criteria:

Level 1: Low degree similarity grouping

Two parts that have one directed code matched are assigned to a particular part family at this level. If there are totally M output signals for the cascaded BP networks, there will be M part families at this level. Since each part has three directed codes, a part can be assigned to three distinct families.

Level 2: Medium degree similarity grouping

Two parts that have two directed codes matched are assigned to a particular part family at this level. There will be $C(M, 2)+C(M, 1)$ part families at this level. Note that a part family at this level is a subset of a family at level 1. Suppose a part family of this level is associated with two directed codes DC_1 and DC_2, this part family is a subset of the part families of level 1 that are associated with DC_1 and DC_2, respectively.

Level 3: High degree similarity grouping

At this level, all parts in a part family should have their tree directed codes matched. There will be $C(M, 3)+C(M, 2)+C(M, 1)$ part families at this level. At this level, a part can be associated with only one part family. A part family at this level is a subset of a part family at level 2.

The proposed criteria result in a hierarchical grouping of part families as shown in Fig. 13. Such a hierarchical grouping would greatly facilitate the activity of design-by-retrieval. That is, at the very beginning of a new part design, a designer can first retrieve and review similar part families at level 1 that are inspired by the designer's vague idea. With more information perceived from the review, the designer can go down to retrieve and examine the part families at level 2 and level 3 that are closer to his or her design intention.

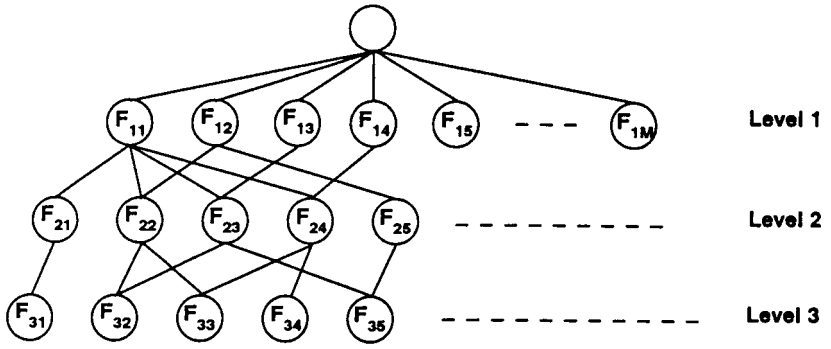


Fig. 13. Hierarchical grouping of part families.

8. Experiment results

To illustrate the proposed method, we consider 24 block-shaped parts. The 3D drawing of these parts are shown in Fig. 14.

8.1. Building the cascaded BP neural networks

All partitioned view-contours for all parts are derived and transformed to corresponding graphs, reference trees, and vectors. Each reference tree will have 17 nodes and each of which is associated with 9 values. As a result, the vector will have 153 elements. Among these partitioned view-contours, three with solid visible line segments, $R1$-T, $R1$-L, and $R3$-T, and three with dashed invisible line segments, $U1$-T, $U3$-S, and $U11$-F, are chosen as the training templates for the first network BP_1; see Fig. 15a. Each of the templates will correspond to a group. So the input layer of BP_1 has 153 neurons and the output layer has 6 neurons. The hidden layer here is assigned to have 10 neurons. Hence BP_1 will be trained to classify inputs to six groups G_1, G_2, G_3, G_4, G_5, and G_6.

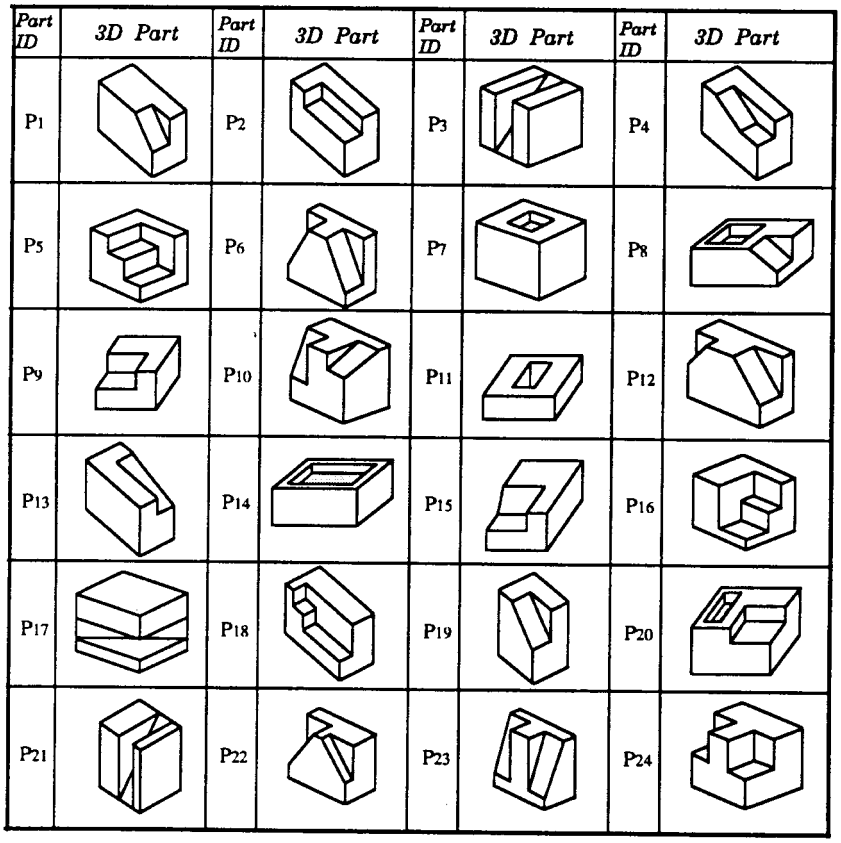


Fig. 14. The 24 tested parts.

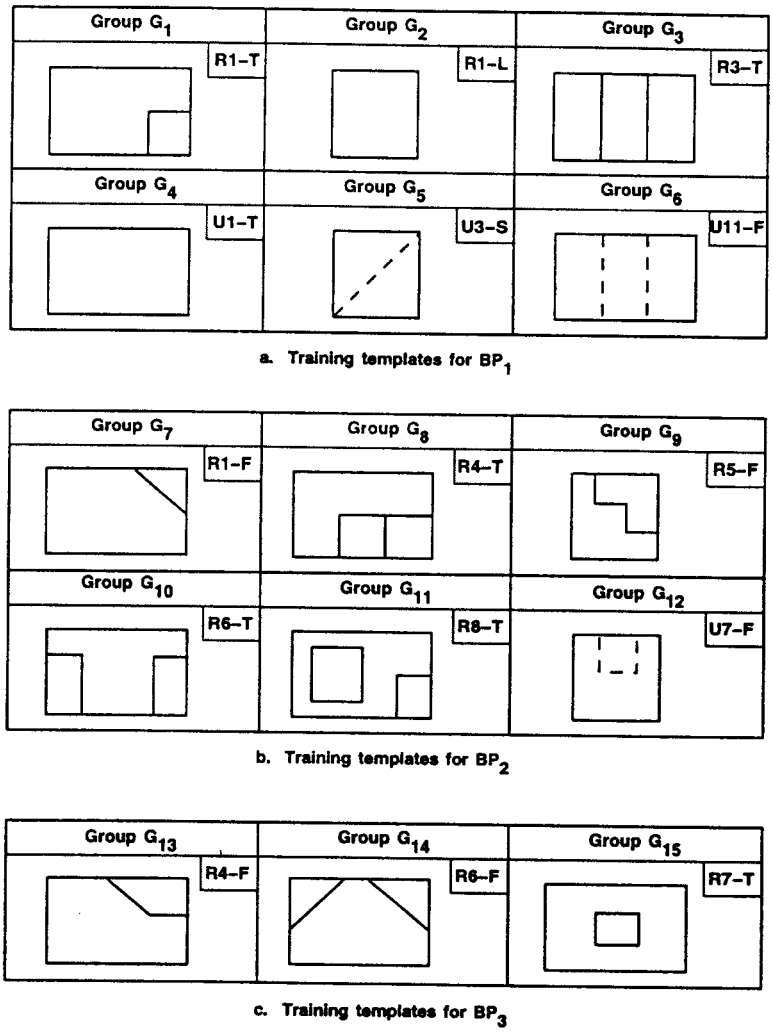


Fig. 15. Training templates for cascaded neural networks.

After BP_1 is trained using the six templates, all vectors of other partitioned contours are fed to BP_1 for classification. The second column of Table 1 depicts the classification result of nine partitioned view-contours for each part, in which $*$ mans the corresponding vector cannot be classified by BP_1. All these partitioned view-contours that have been marked $*$ form X-set, from which the second cascaded network BP_2 will be built. From the X-set, six contours $R1$-F, $R4$-T, $R5$-F, $R6$-T, $R8$-T, and $U7$-F are chosen as training template for BP_2; see Fig. 15b. BP_2 will be trained to classify inputs to six groups G_7, G_8, G_9, G_{10}, G_{11}, and G_{12}. All elements in X-set are then classified by BP_2. The result is shown in the third column of Table 1 and indicates that BP_2 fails to classify some partitioned view-contours, which again are put in X-set. Repeat the same process, the third network BP_3 is trained by the training templates shown in Fig. 15c

Table 1
Classification results

Part ID	Result of BP_1	Result of BP_1 and BP_2	Result of BP_1, BP_2 and BP_3	Directed codes
P_1	{1,2,4,∗,2,4,2,1,4}	{1,2,4,7,2,4,2,1,4}	{1,2,4,7,2,4,2,1,4}	[1,2,4], [7,2,4], [2,1,4]
P_2	{1,2,4,1,2,4,2,1,4}	{1,2,4,1,2,4,2,1,4}	{1,2,4,1,2,4,2,1,4}	[1,2,4], [1,2,4], [2,1,4]
P_3	{3,2,4,3,2,4,2,2,5}	{3,2,4,3,2,4,2,2,5}	{3,2,4,3,2,4,2,2,5}	[3,2,4], [3,2,4], [2,2,5]
P_4	{∗,2,4,∗,2,4,2,1,4}	{8,2,4,∗,2,4,2,1,4}	{8,2,4,13,2,4,2,1,4}	[8,2,4], [13,2,4], [2,1,4]
P_5	{∗,2,4,∗,2,4,2,∗,4}	{8,2,4,9,2,4,2,8,4}	{8,2,4,9,2,4,2,8,4}	[8,2,4], [9,2,4], [2,8,4]
P_6	{∗,2,4,∗,2,4,1,1,4}	{10,2,4,∗,2,4,1,1,4}	{10,2,4,14,2,4,1,1,4}	[10,2,4], [14,2,4], [1,1,4]
P_7	{∗,2,4,2,2,∗,2,2,∗}	{∗,2,4,2,2,12,2,2,12}	{15,2,4,2,2,12,2,2,12}	[15,2,4], [2,2,12], [2,2,12]
P_8	{∗,2,4,∗,2,∗,2,1,∗}	{11,2,4,7,2,12,2,1,12}	{11,2,4,7,2,12,2,1,12}	[11,2,4], [7,2,12], [2,1,12]
P_9	{1,2,4,1,2,4,1,2,4}	{1,2,4,1,2,4,1,2,4}	{1,2,4,1,2,4,1,2,4}	[1,2,4], [1,2,4], [1,2,4]
P_{10}	{∗,2,4,∗,2,4,∗,∗,4}	{10,2,4,10,2,4,7,7,4}	{10,2,4,10,2,4,7,7,4}	[10,2,4], [10,2,4], [7,7,4]
P_{11}	{∗,∗,4,2,2,6,2,2,6}	{∗,∗,4,2,2,6,2,2,6}	{15,15,4,2,2,6,2,2,6}	[15,15,4], [2,2,6], [2,2,6]
P_{12}	{∗,2,4,∗,2,4,1,1,4}	{10,2,4,∗,2,4,1,1,4}	{10,2,4,14,2,4,1,1,4}	[10,2,4], [14,2,4], [1,1,4]
P_{13}	{1,2,4,2,∗,4,2,1,4}	{1,2,4,2,7,4,2,1,4}	{1,2,4,2,7,4,2,1,4}	[1,2,4], [2,7,4], [2,1,4]
P_{14}	{∗,2,4,2,2,∗,2,2,∗}	{∗,2,4,2,2,12,2,2,12}	{15,2,4,2,2,12,2,2,12}	[15,2,4], [2,2,12], [2,2,12]
P_{15}	{∗,2,4,1,2,4,∗,2,4}	{8,2,4,1,2,4,∗,2,4}	{8,2,4,1,2,4,13,2,4}	[8,2,4], [1,2,4], [13,2,4]
P_{16}	{∗,2,4,∗,2,4,∗,2,4}	{8,2,4,9,2,4,8,2,4}	{8,2,4,9,2,4,8,2,4}	[8,2,4], [9,2,4], [8,2,4]
P_{17}	{2,2,5,3,2,4,2,3,4}	{2,2,5,3,2,4,2,3,4}	{2,2,5,3,2,4,2,3,4}	[2,2,5], [3,2,4], [2,3,4]
P_{18}	{∗,2,4,∗,2,4,∗,2,4}	{8,2,4,8,2,4,9,2,4}	{8,2,4,8,2,4,9,2,4}	[8,2,4], [8,2,4], [9,2,4]
P_{19}	{1,2,4,∗,2,4,2,1,4}	{1,2,4,7,2,4,2,1,4}	{1,2,4,7,2,4,2,1,4}	[1,2,4], [7,2,4], [2,1,4]
P_{20}	{∗,2,4,1,2,∗,2,1,∗}	{11,2,4,1,2,12,2,1,12}	{11,2,4,1,2,12,2,1,12}	[11,2,4], [1,2,12], [2,1,12]
P_{21}	{3,2,4,3,2,4,2,2,5}	{3,2,4,3,2,4,2,2,5}	{3,2,4,3,2,4,2,2,5}	[3,2,4], [3,2,4], [2,2,5]
P_{22}	{∗,2,4,∗,2,4,1,1,4}	{10,2,4,∗,2,4,1,1,4}	{10,2,4,14,2,4,1,1,4}	[10,2,4], [14,2,4], [1,1,4]
P_{23}	{∗,2,4,∗,2,4,∗,∗,4}	{10,2,4,10,2,4,7,7,4}	{10,2,4,10,2,4,7,7,4}	[10,2,4], [10,2,4], [7,7,4]
P_{24}	{∗,2,4,∗,2,4,1,1,4}	{10,2,4,10,2,4,1,1,4}	{10,2,4,10,2,4,1,1,4}	[10,2,4], [10,2,4], [1,1,4]

and classifies all the elements in X-set. The classified result of BP_3 will have groups G_{13}, G_{14}, and G_{15}. The final classification result is shown in the column four of Table 1. The directed codes for all parts are also shown in the last column of Table 1.

8.2. Formation of part families

Based on the directed codes formed by the classification result for each part, we now form part families for each level of similarity as follows:

1. Level 1 has 18 families as shown in Table 2.
2. Level 2 has 22 families as shown in Table 3.
3. Level 3 has 12 families as shown in Table 4.

To see the hierarchical structure, we take part family $F_{1,1}=\{P_1,\ P_2,\ P_4,\ P_9,\ P_{13},\ P_{15},\ P_{19}\}$ at level 1 as an example. Parts in F_{1-1} are further classified to part families $F_{2,1}=\{P_1,\ P_2,\ P_9,\ P_{13},\ P_{19}\}$ and $F_{2,5}=F_{2,6}=F_{2,7}=\{P_4,\ P_{15}\}$ at level 2. Parts in F_{2-1} are further classified to part families $F_{3,1}=\{P_1,\ P_{13},\ P_{19}\}$ and $F_{3,2}=\{P_2,\ P_9\}$ at level 3.

Table 2
Part families at level 1

Family ID	Directed code	Family set
$F_{1,1}$	{[1,2,4]}	{P_1, P_2, P_4, P_9, $P_{13,}$, P_{15}, P_{19}}
$F_{1,2}$	{[7,2,4]}	{P_1, P_{13}, P_{19}}
$F_{1,3}$	{[3,2,4]}	{P_3, P_{17}, P_{21}}
$F_{1,4}$	{[2,2,5]}	{P_3, P_{17}, P_{21}}
$F_{1,5}$	{[8,2,4]}	{P_4, P_5, P_{15}, P_{16}, P_{18}}
$F_{1,6}$	{[13,2,4]}	{P_4, P_{15}}
$F_{1,7}$	{[9,2,4]}	{P_5, P_{16}, P_{18}}
$F_{1,8}$	{[10,2,4]}	{P_6, P_{10}, P_{12}, P_{22}, P_{23}, P_{24}}
$F_{1,9}$	{[14,2,4]}	{P_6, P_{12}, P_{22}}
$F_{1,10}$	{[1,1,4]}	{P_6, P_{12}, P_{22}, P_{24}}
$F_{1,11}$	{[15,2,4]}	{P_7, P_{14}}
$F_{1,12}$	{[2,2,12]}	{P_7, P_{14}}
$F_{1,13}$	{[11,2,4]}	{P_8, P_{20}}
$F_{1,14}$	{[7,2,12]}	{P_8}
$F_{1,15}$	{[2,1,12]}	{P_8, P_{20}}
$F_{1,16}$	{[7,7,4]}	{P_{10}, P_{23}}
$F_{1,17}$	{[15,15,4]}	{P_{11}}
$F_{1,18}$	{[2,2,6]}	{P_{11}}

Table 3
Part families at level 2

Family ID	Directed codes	Family set
$F_{2,1}$	{[1,2,4], [1,2,4]}	{$P_{1,}$, P_2, P_9, P_{13}, P_{19}}
$F_{2,2}$	{[7,2,4], [1,2,4]}	{P_1, P_{13}, P_{19}}
$F_{2,3}$	{[3,2,4], [3,2,4]}	{P_3, P_{17}, P_{21}}
$F_{2,4}$	{[3,2,4], [2,2,5]}	{P_3, P_{17}, P_{21}}
$F_{2,5}$	{[8,2,4], [13,2,4]}	{P_4, P_{15}}
$F_{2,6}$	{[8,2,4], [2,1,4]}	{P_4, P_{15}}
$F_{2,7}$	{[13,2,4], [2,1,4]}	{P_4, P_{15}}
$F_{2,8}$	{[8,2,4], [9,2,4]}	{P_5, P_{16}, P_{18}}
$F_{2,9}$	{[8,2,4], [2,8,4]}	{P_5, P_{16}, P_{18}}
$F_{2,10}$	{[10,2,4], [14,2,4]}	{P_6, P_{12}, P_{22}}
$F_{2,11}$	{[10,2,4], [1,1,4]}	{P_6, P_{12}, P_{22}, P_{24}}
$F_{2,12}$	{[14,2,4], [1,1,4]}	{P_6, P_{12}, P_{22}}
$F_{2,13}$	{[15,2,4], [2,2,12]}	{P_7, P_{14}}
$F_{2,14}$	{[2,2,12], [2,2,12]}	{P_7, P_{14}}
$F_{2,15}$	{[11,2,4], [7,2,12]}	{P_8}
$F_{2,16}$	{[11,2,4], [2,1,12]}	{P_8, P_{20}}
$F_{2,17}$	{[7,2,12], [2,1,12]}	{P_8}
$F_{2,18}$	{[10,2,4], [10,2,4]}	{P_{10}, P_{23}, P_{24}}
$F_{2,19}$	{[10,2,4], [7,7,4]}	{P_{10}, P_{23}}
$F_{2,20}$	{[15,15,4], [2,2,6]}	{P_{11}}
$F_{2,21}$	{[2,2,6], [2,2,6]}	{P_{11}}
$F_{2,22}$	{[1,2,12], [2,1,12]}	{P_{20}}

Table 4
Part families at level 3

Family ID	Directed codes	Family set
$F_{3,1}$	{[1,2,4], [7,2,4], [2,1,4]}	$\{P_1, P_{13}, P_{19}\}$
$F_{3,2}$	{[1,2,4], [1,2,4], [2,1,4]}	$\{P_2, P_9\}$
$F_{3,3}$	{[3,2,4], [3,2,4], [2,2,5]}	$\{P_3, P_{17}, P_{21}\}$
$F_{3,4}$	{[8,2,4], [13,2,4], [2,1,4]}	$\{P_4, P_{15}\}$
$F_{3,5}$	{[8,2,4], [9,2,4], [2,8,4]}	$\{P_5, P_{16}, P_{18}\}$
$F_{3,6}$	{[10,2,4], [14,2,4], [1,1,4]}	$\{P_6, P_{12}, P_{22}\}$
$F_{3,7}$	{[15,2,4], [2,2,12], [2,2,12]}	$\{P_7, P_{14}\}$
$F_{3,8}$	{[11,2,4], [7,2,12], [2,1,12]}	$\{P_8\}$
$F_{3,9}$	{[10,2,4], [10,2,4], [7,7,4]}	$\{P_{10}, P_{23}\}$
$F_{3,10}$	{[15,15,4], [2,2,6], [2,2,6]}	$\{P_{11}\}$
$F_{3,11}$	{[11,2,4], [1,2,12], [2,1,12]}	$\{P_{20}\}$
$F_{3,12}$	{[10,2,4], [10,2,4], [1,1,4]}	$\{P_{24}\}$

9. Concluding remarks

We have proposed a classification scheme for block-shape parts based on their 2D projections. The parts are classified based on the interior information within the view contours. Such an interior information is apparently a valuable addition to the contour information. To extract the interior information, we partition a view contour by its projected visible or invisible line segments and, in consequence, obtain nine partitioned contours. Since the proposed neural network receives only a set of scalars, the regions and their adjacency in a partitioned contour are first converted to a graph, then to a reference tree, and finally to a vector form. The part classifier is a cascaded back-propagation neural network system that is trained by selective sets of partitioned view-contours. Such a cascaded structure has advantages that each network can be limited to a small size and trained independently. The input parts are classified into families that can be in one of the three levels of similarity. Such a hierarchical grouping of parts would be very valuable for the activity of design-by-retrieval, in which part families at the levels from lower to higher similarity are retrieved such that the design's intention can be gradually clarified.

References

[1] Bhadra A, Fischer GW. A new GT classification approach: a data base with graphical dimensions. Manufacturing Rev 1988;11:44–9.
[2] Chakaraborty K, Roy U. Connectionist models for part-family classifications. Comput Industr Engng 1992;9:189–98.
[3] Chen CS. A form feature oriented coding scheme. Comput Industr Engng 1989;17:227–33.
[4] Henderson MR, Musti S. Automated group technology part coding from a three-dimensional CAD database. Trans ASME J Engng Industr 1988;110:278–87.
[5] Kao Y, Moon YB. A unified group technology implementation using the backpropagation learning rule of neural network. Comput Industr Engng 1991;20(4):425–37.

[6] Kaparthi S, Suresh NC. A neural network system for shape based classification and coding of rotational parts. Int J Production Res 1991;29(9):1771–84.
[7] Liao TW, Lee KS. Integration of a feature-based CAD system and an ARTI neural network model for GT coding and part family forming. Comput Industr Engng 1994;26(1):93–104.
[8] Lippmann RP. An introduction to computing with neural nets. IEEE ASSP Magazine 1987;April:4–22.
[9] Moon YB, Kao Y. Automatic generation of group technology families during the part classification process. Int J Adv Manufact Technol 1993;6:160–6.
[10] Wu MC, Chen JR. A skeleton approach to modellin 2D workpieces. J Design Manufact 1994;3:229–43.
[11] Wu MC, Jen SR. A neural network approach to the classification of 3D prismatic parts. Int J Adv Manufact Technol 1996;9:123–8.