Bull. Nat'l Taiwan Nor.Univ.
Vol. 38. pp.37-64 (1993)

師大學報
第38期　第37-64頁(民國82年)

Constrained Tree Search for Feature Correspondence

Sei-Wang Chen

Department of Information and Computer Education,
National Taiwan Normal University

Abstract

Constrained search is a useful technique for dealing with feature correspondence problems frequently encountered in machine vision. In this paper, an indexing scheme, inspired by the Dewey decimal notation, is proposed for simplifying the implementation of the search algorithm. This scheme also enables the algorithm to be easily realized in hardware in view that the algorithm characterized by the scheme will behave like a digital counter. Through analyzing the complexity of the search algorithm, a process, that can be used to select a robust subset of features and constraints for the purpose of object recognition, is presented. Experiments demonstrate the feasibilities of both the algorithm and the process.

1. Introduction

Object recognition from sensory data is an interesting topic in the field of computer vision. Essential difficulties commonly encountered in object recognition include degradation of input data and complexity of problems. Sensory data are generally deteriorated during acquisition. Noise and distortion are two main sources of degradation. Factors that could increase complexity of problems include scenes containing multiple objects and objects having complex shapes. Degradation of data is usually improved at the earlier stage of processing, whereas complexity of problems in general needs great efforts to deal with. A feasible solution to the problem of complexity is to introduce a priori information. Object models, that are one of such information, play important roles at various stages of object recognition. For instance, at the early stage of feature detection, artifacts and irrelevant features might be discovered with the help of models. At the intermediate stage of shape reconstruction, models could provide clues in selecting suitable shape prototypes for fitting sensory data. Moreover, most recognition systems achieve their goals via matching object models with sensed objects. Such systems are typically referred to as model-based recognition systems.

Two general issues involved in the development of model-based recognition systems are : construction of object representation, and development of matching process. Major concerns in object representation include the decision of feature classes, the choice of feature constraints, and the selection of data structures. They are in collect referred to as the determination of representation schemes. Once a representation scheme has been selected, matching procedures characterized by the scheme can be developed. Two steps comprise matching procedures: consistent labeling as well as parameter estimation. Consistent labeling is to determine the correspondence between sensed and model features, whereas parameter estimation is to derive the transformation between them. To illustrate the

idea of model-based matching procedures, let us look at an example. Refer to Figure 1. Suppose that objects are represented as wire frames, such as the one shown in Figure1(a), and that a set of sensed features has been available. For simplicity,assume also that the 3D counterparts of sensed features can be derived. In this example, features are lines and surfaces (Figure 1(b)). The recognition problem then says that given a set of sensed features and a set of object models identify as well as locate objects appeared in the scene based on the features and models. Figure 1(c) shows a solution to this problem, which could be obtained as follows. For each model in the database, match its features with the sensed features. The matching process consists of three stages. In the first stage, the correspondence between the model and the sensed features is determined. Based on the resultant information of correspondence, transformation between the model and the sensed features can thereby be derived. If the transformation could properly overlap the model features with the sensed features, the scene object is recognized as the object represented by the model being tested and the pose of the object is specified by the transformation.

(a) Object model (b) Sensed features (c) Overlapped

Figure 1. Pose determination

In general, the step of consistent labeling takes much more time than the step of parameter estimation in a matching process. In this paper,we are devoted to the step of consistent labeling. Substantial methods have been proposed. Techniques include random sample concensus [Fis81], local feature focus [Bol82],

hypothesize and test [Bro84], dynamic programming [Bel57], relaxation [Wal75], and constrained tree search[Gri87]. Random sample concensus, proposed by Fischler and Bolles [Fis81], each time randomly selects three pairs of sensed and model features to calculate a transformation. Calculated transformation is then applied to the sensed features. Those transformed features, that can properly overlap model features, are collected. If the number of collected features is greater than a threshold, an optimization procedure is invoked to recalculate the transformation based on the collected features. Since this method doesn't examine constraints among features, all combinations of features will be examined in the worst case. Bolles & Cain [Bol82] presented another method, called local feature focus, that chooses only consistent pairs of sensed and model features to calculate transformations. Thus, Bolles and Cain's method greatly reduces the number of feature pairs to be tested. While this method inspects local consistency, it places no emphasis on global consistency. Grimson and Loazano-Perez [Gri84] proposed a constrained tree search technique, referred to as *interpretation tree search*, which takes care of both local and global constraints among features before calculating their transformations.

In this paper, we study in several aspects the constrained tree search technique. In Section 2, the fundamental idea of matching by constrained tree search is addressed. An indexing scheme, inspired by the Dewey decimal notation, for simplifying the implementation of search procedure is proposed in Section 3. This scheme also enables the algorithm to be easily realized in hardware. The algorithm characterized by the scheme is presented in Section 4. Section 5 analyzes the complexity of the algorithm. Based on the result of analysis, we propose a process that will be used to select a robust subset of features and constraints for applications of object recognition. Section 7 demonstrates the process. Concluding remarks are given in Section 8.

2. Matching by Interpretation Tree Search

Interpretation trees are basically a data structure used to describe the relationship between two groups of elements. To our knowledge,this data structure is originally reported by Gaston & Lozano-Perez in their work on tactile recognition and localization [Gas84]. Since then, this data structure has been widely employed in a variety of applications for solving such problems as image registration, feature correspondence, and consistent labeling. Grimson [Gri84, 86, 87, 88,89] has carried out a series of profound investigations into various aspects of the structure and associated mechanisms about the interpretation trees. In this section, the formation of interpretation trees for feature correspondence problems and the idea of matching by constrained tree search are addressed.

An interpretation tree for a feature correspondence problem is constructed as follows. Refer to Figure 2. A feature correspondence problem is sketched in the upper square of the figure. The object model selected to test is depicted on the left side of the square, which is a pyramid having 6 edges and 4 faces. A set of sensed features in 3D space shown on the right side of the square contains two edge segments (e_1 and e_2) and two surface patches (p_1 and p_2). The arrows drawn from the sensed features to the model features specify a consistent correspondence, subject to some predefined constraints, between the sensed and the model features.It is possible that there exist different correspondences which satisfy the same set of constraints. We call this the ambiguities of interpretation.

Figure 2(b) shows an interpretation tree formed from the correspondence problem shown in Figure 2(a). At the beginning of tree formation, a root(level 0) is first created, denoting the problem. Next, seven nodes are created on the second level (level 1). This level corresponds to the first sensed edge e_1. The first node on this level is a *nil* node, which means that e_1 may result from noise (an artifact edge) or belong to other objects (an irrelevant edge). The following six nodes mean that e_1 may correspond to one of the six model edges. Note here that model edges have been numbered for clarity. The above process continues for

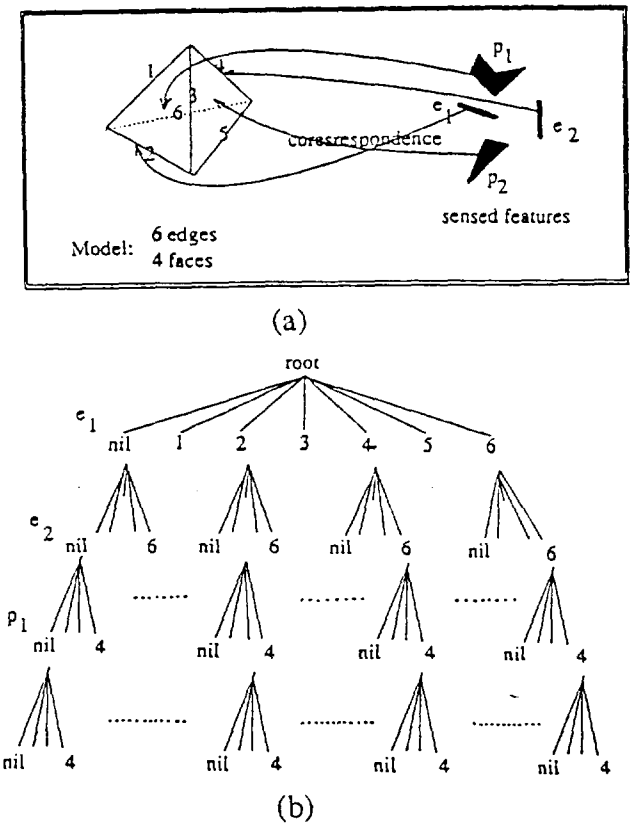

Figure 2. A labeling problem of looking for the correspondences between the sensed and the model features.

level 2 of the tree, which corresponds to the second sensed edge e_2. The resulting tree thus possesses three levels, level 0, 1, and 2, and 7^2 nodes on level 2. The above process then continues for sensed surfaces p_1 and p_2. Finally, a tree with four levels and $7^2 \cdot 5^2$ leaf nodes is constructed 〔Figure 2(b)〕.A path from a node on level 1 to a leaf node on level 4 indicates a possible correspondence between the sensed and the model features, referred to as an *interpretation* of the sensed features. Interpretations must be examined in order to determine consistent ones, which satisfy predefined constraints.

Apparently, the size of the tree will increase exponentially as either number of model or sensed features increases. Except extremely simple cases, it is neither possible nor necessary to construct the entire tree before looking for consistent interpretations. Strategies for pruning the tree are necessary. Examples include

the introduction of robust constraints, hypothesize-and-test, as well as optimal sensing positions for subspace isolation [Gri88].

3. Indexing Scheme

To develop a constrained tree search algorithm, we introduce an indexing scheme for the interpretation tree, which is inspired by the *Dewey decimal notation* [Knu73]. The indexing scheme associates each node of the tree to a number, called the *node number*. Assume that the tree has l+1 levels. Let n_1, n_2, $\cdots$, n_l be a sequence of node numbers along a path P from a node on level 1 to a leaf node. If the node numbers are concatenated sequentiall, a single number $n_1n_2\cdots n_l$ is obtained.Refer to this number as the *path number* of path P. Look at the example in Figure 2. If we replace *nil* nodes with number 0, then all nodes in the tree are numbered. Since the node numbers along the first path are all zero, their concatenation leads to 0000. The first path is thus represented as 0000. Similarly, the second path is represented as 0001, and the third path is denoted by 0002, and so forth.

(0000) (0001) (0002) (0003) (0004) (0010) (0011) (0012) (0013) (0014)

(0020) (0021) (0022) (0023) (0024) (0030) (0031) (0032) (0033) (0034)
(0040) (0041) (0042) (0043) (0044) (0100) (0101) (0102) (0103) (0104)

(0600) (0601) (0602) (0603) (0604) (0610) (0611) (0612) (0613) (0614)

(0640) (0641) (0642) (0643) (0644) (1000) (1001) (1002) (1003) (1004)

(6630) (6631) (6632) (6633) (6634) (6640) (6641) (6642) (6643) (6644)

The above set of numbers collects all path numbers of the tree in Figure 2. In this particular example, every path number is composed of four digits extracted

from the four levels of the tree, respectively. Note that each level of the tree corresponds to one sensed feature. If a digital counter is to be developed for automatically generating the path numbers, the counter should recognize radices of the digits of the path numbers. Unlike decimal numbers whose digits all have the same radix of 10, the path numbers have distinct radices of digits at different locations. For instance, the radices for the rightmost two digits of the above path numbers are 5, and the radices for the leftmost two digits are 7. Radices can be determined from the numbers of model features. For example, radix 5 comes from the number of model faces (4) plus one, whereas radix 7 comes from the number of model edges(6) plus one. We call the number 7755 formed from the concatenation of the radices of digits at different locations the *base number* of path numbers. Based on this base number, the counter will be able to figure out when and where to reset and carry a digit. For instance, the path number next to 0004 is 0010, which is obtained by reseting and carrying the last digit 4 of 0004. Similarly, the next path number to 0644 is 1000.

A path number indicates a possible correspondence between the sensed and the model features. To illuminate this correspondence , let us look at a path number, say 2041. This number suggests that the first sensed edge e_1 corresponds to model edge 2, and the second sensed edge e_2 corresponds to none of model edges, and that the first sensed surface patch p_1 corresponds to model face 4, and the second sensed patch p_2 corresponds to model face 1. A correspondence between sensed and model features is said to be *feasible* if the relationships among sensed features are consistent with the relationships among their corresponding model features.

Due to the above indexing scheme, an interpretation tree can be represented as a set of path numbers. Instead of prestoring all the path numbers in the procedure, the numbers can be generated during processing. Searching paths on an interpretation tree will be equivalent to scanning over the path numbers of the tree. We now turn to the issue of how to prune an interpretation tree in terms of its path numbers. A pruning operation can actually be implemented as a skip over

a sequence of path numbers. The problem now turns out to be the determinations of where to start a skip and when to stop the skip. Refer to Figure 3. Suppose that the number of the path being examined is $n_1 \cdot \cdot n_i n_{i+1} \cdot \cdot n_l$, and that the node being tested is n_i.If the node n_i is found to have violated some relational constraints with other nodes, We can remove the subtree rooted on node n_i. The next path to be tested will be the path with the number $\cdot \cdot n_1 \cdot \cdot n_1$+1 0$\cdot \cdot$0. That is, the numbers from$n_1 \cdot \cdot n_i n_{i+1} \cdot \cdot n_l$ to $n_1 \cdot \cdot n_i b_i$+1$\cdot \cdot b_l$ are omitted, where $b_1 \cdot \cdot b_i \cdot \cdot b_l$ is the base number.

The above pruning operation can be summarized as follows: (1) add one to the digit in the path number where a violation of relational constraint occurs, (2) check the resulting digit by referring to the base number to see whether or not a reset-and-carry operation is needed for the digit,and (3) zero all digits to the right of the digit being examined. Look at an example for illustration. Let the path number being examined is 1623. Suppose that the second node 6 of the path 1623 is found to be illegal.The subtree rooted at node 6 can be pruned. To find the next path number for examination, we add one to the digit 6, leading to digit 7. Then, check the resulting digit by referring to the base number 7755, and find that the digit needs to be reset and carried. This leads path number1623 to path number 20 23. Finally, zero the digits to the right of the digit 0. The path number to be examined next is thus 2000.

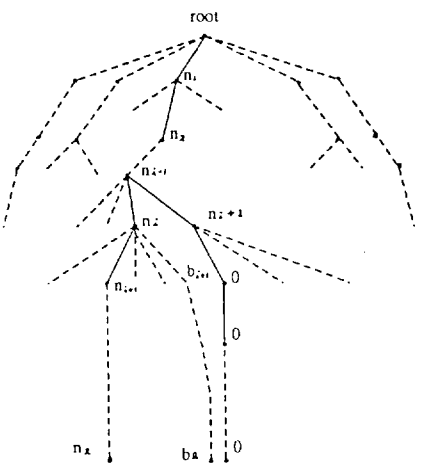

Figure 3. A pruning operation applied to the interpretatiion tree is equivalent to skipping a sequence of path numbers.

4. Constrained Tree Search

Consider a general case with w feature classes $\{c_i, i=1...,w\}$. Suppose that we have a set of N_s sensed features, in which n_1 features are of class c_1, n_2 features are of class c_2, and so forth, where $\sum_{i=1}^{w} n_i = N_s$. Suppose further that the selected model contains N_m features, in which m_1 features are of class c_1, m_2 features are of class c_2, *and soon*, *where* $\sum_{j=1}^{w} m_j = N_m$. Based on the above information about features, the base number can be formed as follows :

$$(m_1+1)\cdot\cdot(m_1+1)(m_2+1)\cdot\cdot(m_2+1)\cdot\cdot(m_w+1)\cdot\cdot(m_w+1).$$

The first n_1 digits of the base number possess the same value of m_1+1, the next n_2 digits of the number possess the same value of m_2+1, and so forth. Finally, the last nw digits of the number have the same value of m_W+1. Note that the total number of digits in the above base number is $\sum_{i=1}^{w} n_i = N_s$.

Having defined the base number, We use a procedure, called *PathGenerator*, to generate each time a candidate path number by given the current path number. This procedure is presented in the Appendix. At the beginning of the search algorithm, an initial path number is first selected and passed to *PathGenerator*. It is unnecessary that the initial path number should start with 00....0. In practice, it can be determined from a prior information or certain heuristics. For example, in the applications of 3D localization, those path numbers whose numbers of non-zero digits are smaller than three can be ignored. From a view of the interpretation tree, paths with their lengths smaller than three can be pruned from the tree. This is primarily due to the requirement that three pairs of corresponding sensed and model features are necessary to uniquely determine a transformation between the sensed object and the model. By properly selecting the initial path number, the search space can considerably be reduced.

Once a candidate path has been determined, another procedure, called *PathCheck*, is invoked to verify the feasibility of the path. Recall that a path

indicates a possible correspondence between the sensed and the model features. *PathCheck* verifies the correspondence via inspecting the relationships among features, called feature constraints. The detail of this procedure is given in the Appendix. If the candidate path passes the verification, namely, the relationships among the sensed features are consistent with those among the model features, the path is then referred to as a *feasible path*. The objective of the search algorithm is to find all feasible paths in the interpretation tree. Obviously, the fewer the number of feasible paths is, the better is for the later decision of recognition.One approach to reducing the number of feasible paths is via introducing robust constraints for pruning the tree.

In addition to the elimination of ambiguity of interpretations, the most important contribution of robust constraints is its reduction of search space. Grimson [Gri86] has claimed that a desirable set of constraints should hold both requirements of independence and completeness. The requirement of independence is to avoid redundancy among constraints, whereas the requirement of completeness is to prevent ambiguity of interpretations. From an empirical point of view,proofs of independence and completeness for a given set of constraints are neither trivial nor necessary. Redundancy is not always negative. Many experiments have shown that redundancy in databases can contribute accuracy to the final results. Furthermore, recognition systems with mechanisms of hierarchy and coarse-to-fine have widely been exploited. Such systems usually involve a great amount of redundancy in their structures. On the other hand, a complete set of constraints doesn't always lead to a unique interpretation if objects with highly symmetrical and incomplete shapes are involved. As a result, a small set of powerful constraints is preferable because it can achieve most of the purpose without spending too much time.In the next section, we will present a process for choosing such constraints from a given set of candidates.

The constrained tree search algorithm comprised the two procedures of *PathGenerator* and *PathCheck* is shown in the Appendix. Strategies for further improving the performance of the algorithm can also be embedded. An example

of such strategies is the hypothesize-and-test.With this strategy, the feasibility of a path can be determined earlier before completing examination of the entire path.

5. Analysis of Time Complexity

Let N_c be the total number of checks on feature constraints carried out during searching an interpretation tree. From a probabilistic point of view, N_c is typically a random variable. The time complexity of the search algorithm is defined in terms of the expected value $E[N_c]$ of the random variable N_c. Although there may be several different kinds of feature constraints, in this analysis we only consider two kinds: unary and binary constraints. Unary constraints specify the properties of individual features, whereas binary constraints concern the relational properties between two features. Constraints for more than two features are applicable, but are not considered in this analysis. Let $p_u$denote the probability that the unary constraint of a sensed feature is consistent with that of a model feature, and p_b be the probability that the binary constraint between a pair of sensed features is consistent with that between a pair of model features. p_u and p_b are sometimes referred to as the *consistency rates* of unary and binary constraints, respectively.

Without loss of generality, assume that the sensed features have been clustered according to their classes and been ordered into a sequence.Refer to the previous section. N_s and N_m are the total numbers of sensed and model features, respectively. n_i is the number of sensed features of class c_i, and m_j is the number of model features of class c_j. Let N_p be the random variable of surviving paths on the leaf level of the tree. Let S_i denote the expected value of surviving paths on level i. Therefore, $S_{N_s}=E[N_p]$. On the first level, there are m_1 nodes representing m_1 model features of class 1 which are the same class as the first sensed feature. On this level, only unary constraints between the sensed feature and the model features need to be examined. Since we assume that the consistency rate of unary constraint is p_u, the expected value of surviving paths on this level is thus m_1p_u, i.e., $S_1=m_1p_u$. Furthermore, since each surviving path will produce m_1 paths on the

next level, there are totally m_1S_1 paths on the second level. Among these paths, $p_up_bm_1S_1$ paths can survive the inspections of both unary and binary constraints. More explicitly, after checking on unary constraints$p_um_1S_1$ paths remain, and after checking on binary constraints $p_bp_um_1S_1$ paths remain. The expected value of surviving paths on the second level is thus $[m_1p_up_b]S_1$, i.e., $S_2=[m_1p_up_b]S_1$. In a similar vein,the expected values of surviving paths on various levels of the tree are, respectively,

$$S_1=m_1p_u,$$

$$S_2=[m_1p_up_b]S_1,$$

$$S_3=[m_1p_up_b^2]S_2,$$

.....

$$S_{n1}=[m_1p_up_b^{n_1-1}]S_{n_{1-1}},$$

$$S_{n_{1+1}}=[m_2p_up_b^{n_2}]S_{n1},$$

.....

$$S_{Ns}=[m_wp_up_b^{N_w-1}]A_{N_w-1}.$$

By iterative substitution, we obtain,

$$E[N_p]=S_{Ns}=[\prod_{i=1}^{w}m_i^{n_i}]p_u^{N_p}p_b^{\sum_{j=1}^{N_p-1}j}=[\prod_{i=1}^{w}m_i^{n_i}]p_u^{N_p}p_b^{N_p(N_p-1)/2}. \quad (1)$$

Next, consider the numbers of checks on feature constraints carried out on different levels of the tree. Let C_j denote the expected value of checks performed on level j of the tree. On the first level, there are m_1 nodes representing m_1 model features. Since model features on this level all have to be examined, m_1 checks on unary constraints are executed on the first level, i.e., $B_1=m_1$. On the second level, there are m_1S_1 nodes. Each node needs to compare its unary constraint with that of the second sensed feature; m_1S_1 checks are executed. Since the consistency rate of unary constraints is p_u,$p_um_1S_u$ nodes will pass the checks of unary constraints. The binary constraints of those nodes passing the above checks should be executed. This takes another $p_um_1S_u$ checks. The total number of checks for unary and binary constraints is thus $m_1S_1+p_um_1S_u=(m_1+m_1p_u)S_1$ for the second level, i.e., $C_2=(m_1+m_1p_u)S_1$.

The expected values of checks for all levels of the tree are shown below:

$$C_1=m_1,$$

$$C_2=(m_1+m_1p_u)S_1,$$

$$C_3=(m_1+2m_1p_u)S_2,$$

$$C_4=(m_1+3m_1p_u)S_3,$$

.....,

$$C_{t1}=[\,m_1+(t_1-1)m_1p_u\,]\,S_{t_1-1},$$

$$C_{t_1+1}=[\,m_2+t_1m_2p_u\,]\,S_{t1},$$

.....,

where $S_{ni}=[\,m_{ni}p_u p_b^{n_i-1}\,]\,S_{n_i-1}$, $n_i=1,....,N_s$. By iterative substitution, $S_{ni}=[\prod_{j=1}^{w}m_j^{n_j}]\,p_u^{n_i}p_b^{\sum_{k=1}^{n_i}k}$ We obtain the expected value of constraint checks for the leaf level of the tree to be,

$$C_{N_s}=(\prod_{i=1}^{w}m_i^{n_i})p_u^{N_s-1}p_b^{(N_s-1)(N_s-2)/2}\,[\,1+(N_s-1)p_u\,].$$

Finally, the expected value $E[N_c]$ of total constraint checksis the summation of C_i's.

$$E[N_c]=\sum_{i=1}^{N_s}C_i. \tag{2}$$

In practice, there may have several different constraints in each kind of constraints. Suppose that we have n_u different unary constraints with respective consistency probabilities $\{p_u j=1,...n_u\}$, and n_b distinct binary constraints with respective probabilities $\{p_{b_i}k=1,...,n_b\}$. The expected value of constraint checks for such a more general case can be acquired by replacingp_u with $\prod_{j=1}^{n_u}p_{u_j}$, and replacing p_b with $\prod_{k=1}^{n_b}p_{b_k}$ in the above equations.

It is clear that the number of checks will increase exponentially as the numbers of sensed and model features increase. In order to reduce the explosive number of checks, the consistency probabilities for various feature constraints should be low, namely, constraints utilized should be powerful enough. Since

each probability is less than one, the product of a number of probabilities should be less than individual probabilities.This reveals that more constraints will contribute more reduction to the search space. However, it is not always the case because there may have redundancy among constraints. In the next section, a process by which robust constraints can be determined from a given set of candidates is presented.

6. Selection of Powerful Constraints

Start with a simple case in which one single feature class, one kind of unary constraints as well as one kind of binary constraints are involved. For this simple case, the expected value of surviving paths on the leaf level of the interpretation tree is,

$$E[N_p] = N_m^{N_s} p_u^{N_s} p_b^{N_s(N_s-1)/2}. \tag{3}$$

This equation is a simplified version of Equation (1). For a little more complicated case of multiple feature classes, the expected value of surviving paths becomes,

$$E[N_p] = [\prod_{i=1}^{w} m_i^{n_i}]\, p_u^{N_s} p_b^{N_s(N_s-1)/2}, \tag{4}$$

where w is the number of feature classes, mi and ni are the numbers of model and sensed features of class c_i. If we further extend the case to a case with multiple constraints, the equation then is,

$$E[N_p] = [\prod_{i=1}^{w} m_i^{n_i}]\, [\prod_{j=1}^{n_u} p_{u_j}]^{N_s}\, [\prod_{k=1}^{n_b} p_{b_k}]^{N_s(N_s-1)/2}, \tag{5}$$

where n_u and n_b are the numbers of distinct unary and binary constraints, respectively, and p_{uj} and p_{bk} are their corresponding consistency probabilities.

From the above equation, clearly, the expected value of surviving paths depends on the parameters $w, m_i, n_i, n_u, n_b, p_u, p_{b_k}$, and N_s, but not on N_m, the number of model features. Let us examine the influences of these parameters on the performance of the search algorithm. Note that reducing the expected value $E[N_p]$ of constraint checks is equivalent to improving the performance of the algorithm.

Look at the first term of the equation. Parameters m_i and n_i only appear in this term,which reveals that if either number of model features m_i or sensed features n_i has been increased, the number of constraint checks would be increased exponentially. Intuitively, this is true. However, if we further look at the other two terms in the equation, we will find that it may not be the case. Since p_{u_i} and p_{bk} are probabilities both less than one, their productions and powers will greatly reduce the number of checks as well. This argument can become applicable only if p_{u_i} and p_{bk} are small enough; equivalently, only if feature constraints are powerful enough.On the other hand, the larger the parameters n_u and n_b are,the better is the performance of the algorithm. That is, the more distinct constraints are introduced, the better is for the performance of the algorithm. Unfortunately, there is a trade-off between the number of constraints and the processing time. Finally, since $N_s(N_s-1)/2$ is almost always larger than N_s (except $N_s=1 or 2$), the binary constraints in general play a more important role in reducing the number of checks than unary constraints.

A typical sequence of stages commonly followed by researchers for dealing with object recognition problems is as follows. First, a set of feature classes most suitable for discriminating objects in the problem domain is decided. Constraints characterized by the feature classes are then determined next. Thereafter, schemes for representing objects and matching procedures based on the representation schemes are finally developed.Unfortunately, the first two stages of feature classes and constraints are commonly accomplished heuristically. There seems to be no systematical ways to follow. In the remainder of this section, we proposed a process for selecting both suitable features and robust constraints from given sets of candidates.

From Equation (5), we divide both sides of the equation by $\prod_{i=1}^{w} m_i^{n_i}$ and then take logarithm of the resulting equation. A linear equation in log p_{u_i}'s and log p_{b_i}'s can be obtained,

$$\log\frac{E[N_p]}{\prod_{i=1}^{w} m_i^{n_i}} = N_s\prod_{j=1}^{n_u}\log p_{u_j} + \frac{N_s(N_s-1)}{2}\sum_{k=1}^{n_b}\log p_{b_k} \tag{6}$$

Empirically, parameters w, m_i, n_i, n_u, n_b and N_s can be known a priori, while $E[N_p]$ need to be estimated from experiments. Having known the above information, the consistency probabilities p_u's *and* p_b's for various feature constraints can be calculated from solving a system of linear equations characterized by Equation (6). Robust constraints are then selected as those having small consistency probabilities. To illuminate this,suppose that we want to estimate the probabilities p_{ui} and p_{bj} of a certain unary constraint i and a certain binary constraint j. To this end, we first set the probabilities of other constraints to one. Here, setting the probability of a constraint to one means to disable the constraint, or, equivalently, to omit the constraint. Equation (6) thus reduces to,

$$log\frac{E[N_p]}{N_m^{N_s}} = N_s log p_{u_i} + \frac{N_s(N_s-1)}{2} log p_{b_j}.$$

In this equation, N_m and N_s are known a priori. However, $E[N_p]$ can only be known from experiments. Substituting the values of these parameters into the equation, the resulting equation contains only unknows p_{ui} and p_{bj}. Two experimental results are necessary for uniquely solving p_{ui} and p_{bj}. Repeatedly applying the above procedure to different constraints, their consistency probabilities can be estimated. Although based on the computed probabilities of constraints, we can select those with small probability values as robust constraints , and we should also realize that a set of individually robust constraints may not still appear powerful in collect. It would be better to test directly various subcollections of constraints so as to determine which subcollection is the best. Next, due to the fact that constraints can only be determined after the set of feature classes has been selected, to choose a suitable set of feature classes, we may apply the above process of constraint selection to different subsets of given feature classes, and then select the subset that produces the best robust constraints.

7. Experimental Results

In this section, we demonstrate the process of feature and constraint selection presented in the last section. Experimental data, including model and sensed features, are acquired from a housing [Gri86] shown in Figure 4. Features considered in this experiment include edges and faces. Since our purpose here is to reveal the applicability of the proposed selecting process, without loss of major objective the problem has been simplified by choosing only a subset, instead of the entire set, of features of the housing as our experimental model features. Sensed features are created by applying operations of size reduction, noise addition, as well as spatial transformation to the chosen model features. Feature constraints are, afterwards, calculated for both model and sensed features. Feature constraints can be categorized into two classes: unary and binary constraints. Unary constraints include length of edges and size of faces, whereas binary constraints include angle and distance between features (e.g., edge-edge, edge-face, and face-face). The calculation of constraints has been detailed elsewhere[Che92]. The calculated constraints are candidate constraints from which robust ones are to be determined via our proposed process.

Refer to Equation (6). Three sets of data are employed. The first set contains 15 model edges, i.e., m_1=15, and 10 sensed edges, i.e., N_s=10. Therefore, this set of data contains only edge evidence, i.e., w=1 and n_1=1. Suppose also that two binary constraints, i.e., n_b=2, and no unary constraint, i.e., n_u=0, are employed. The last value needed to know for Equation (6) to calculate the consistency probability pb is the value of $E[N_p]$, which can be determined from experiments on the constrained search algorithm. Table 1 summarizes the experimental results using the first set of data. There are ten rows in the table, each corresponding to one level of the interpretation tree. The first three columns of the table show the respective numbers of reaching, died, and surviving paths. The next two columns

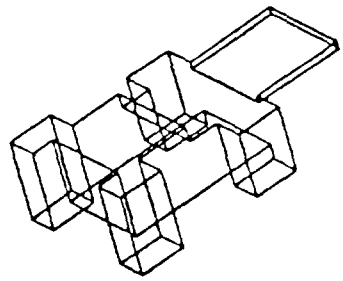

Figure 4. A housing given by [Gri86]

show the numbers of paths without and with *nil* nodes, respectively. The last column of the table shows the number of checks on feature constraints. To make more sense about this table, let us look at a row, say row 6, of the table as an example. Row 6 is associated with sensed feature 6,that in turn corresponds to the sixth level of the interpretation tree. There are 12432 paths reaching the sixth level. Of these 12432 paths, 11495 paths die out due to violating some feature constraints.Therefore, 937 paths are surviving on this level. Among these surviving paths, only one path has no *nil* node, and each of the others has at least one *nil* node on it. The above results are obtained after 12188 checks on feature constraints performed on this level.

Level	#Path reaching this level	#Die	#Survive	#Survive on this level without nil	#Survive on this evel with nil	#Checks on this level
1	16	0	16	15	1	0
2	256	195	61	30	31	225
3	976	778	198	25	173	950
4	3168	2856	312	3	309	3197
5	4992	4215	777	1	776	5900
6	12432	11495	937	1	936	12188
7	14992	13729	1263	1	1262	14889
8	20208	18267	1941	1	1940	22008
9	31056	27983	3073	1	3072	33755
10	49168	44729	4439	1	4438	56618

Table 1. The summary of experimental results using the first set of data (edge evidence only)

Having obtained the table from experiments, the value of $E[N_p]$ can then be determined from the table. First, consider the cases of paths with *nil* nodes. Let $E[N_p]$ be 4438, which is the number of surviving paths on level 10 of the tree. This number means that after comparing with the model features based on given feature constraints 4438 interpretations of the sensed features satisfy the constraints. Substitute the values of parameters into Equation (6). We obtain an equation as:

$$log4438 - 10log15 = 45logp_b.$$

Solve this equation for p_b. We get, $p_b = 0.66$. This value indicates that more than half of binary constraints between sensed features are consistent with those between model features. Unfortunately, the result is too large for the binary constraints to be powerful. Recall that there are two kinds of binary constraints employed: angle and distance constraints.See Figure 4. The vertices of the housing all possess 90 degrees. Angle constraints in this case will contribute nothing to the consistency probability. The above p_b value thus only indicates the consistency probability for distance constraints. It is obvious that constraints common to features will be less useful. It is also reasonable to think that if there are other constraints as powerful as the distance constraint, the resulting probability will decrease exponentially. More explicitly, if there are n such constraints, the resulting probability would be p_b^n.

Since checks involving *nil* nodes are ignored, equivalently, are assumed to be satisfied, *nil* nodes will also contribute nothing to the consistency probability. Let us look at paths without *nil* nodes. On level 10, the number of surviving paths without *nil* nodes is 1. Therefore, set $E[N_p]$ to 1, and the other parameters remain the same. We get, p_b=0.548. This value is still large. However, as we can see in the table, on level 5, the number of surviving paths already reached 1. This means that five sensed features are already enough for the search algorithm to locate the true path. More sensed features involved not only haven't improved the result, i.e.,further reduced the number of paths, but have decreased the powerfulness of constraints, i.e., increased the consistency probability. The reason is that

many redundant checks are executed. To figure out the actual robustness of the binary constraints, let us consider the first five sensed features, i.e., N_s=5 and n_1=5.We get p_b=0.258. If we reduce the number of sensed features, for instance, N_s=4, then the number of surviving paths is 3,i.e., n_1=3. We get, p_b= 0.197. Note that the last case has ambiguity because three paths remain. We cannot further reduce the number of sensed features because it will lead to a large ambiguity(25 surviving paths). The above results about the consistency probability of edge evidence are summarized in Table 4.

Level	#Path reaching this level	#Die	#Survive	#Survive on this level without nil	#Survive on this level with nil	#Checks on this level
1	16	0	16	15	1	0
2	256	185	71	40	31	225
3	1136	915	221	40	181	1324
4	3536	3049	487	19	468	3865
5	7792	6947	845	2	843	8533
6	13520	11431	2089	4	2085	17902
7	33424	28231	5193	8	5185	44668
8	83088	74783	8305	3	8302	96019
9	132880	119808	13072	3	13069	171333
10	209152	191489	17663	2	17661	247096

Table 2. The summary of experimental results for the second set of data (face evidence only).

Level	#Path reaching this level	#Die	#Survive	#Survive on this Level without nil	#Survive on this Level with nil	#Checks on this level
1	16	0	16	15	1	0
2	256	155	101	70	31	225
3	1616	1457	159	12	147	1567
4	2544	2300	244	2	242	2466
5	3904	3282	622	1	621	4253
6	9952	8680	1272	1	1271	11543
7	20352	17889	2463	1	2462	25980
8	39408	34860	4548	1	4547	51085
9	72768	65370	7398	1	7397	85749
10	118368	108429	9939	1	9938	133737

Table 3. The summary of experimental results for the third set of data (mixed evidence)

The above process is repeatedly applied to the second and the third sets of data. The second set of data, containing only face evidence, includes15 model faces and 10 sensed faces. The third set of data, containing mixed evidence, consists of 10 model edges, 5 model faces, 6 sensed edges, and 4sensed faces. Experimental results on the search algorithm using the second and the third sets of data are collected in Tables 2 and 3, respectively.From Equation (6), both numbers of checks on feature constraints and surviving paths should increase exponentially as the level of the tree increases, and so does the ratio of both numbers. However, look at the last column of each table. The ratio of the number of checks to the number of surviving paths increases linearly from 7 to 14 as the level of the tree increases from 2 to 10. It must be that an exponential number of paths has been eliminated at every level. This thus keeps the processing time within an reasonable range. Tables 2 and 3 provide values for $E[N_p]$ in Equation (6). The calculated consistency probabilities for distinct sets of data and for different values of $E[N_p]$ are shown in Table 4. This table reveals that using edge evidence and associated constraints in locating the correct path, i.e., the true interpretation of the sensed features, is better than using face evidence and the related constraints.However, mixed evidence performs as well as edge evidence.

This shows that the effect of face evidence has been suppressed by the effect of edge evidence when mixed evidence is used.

on this level	nil nodes	edge evidence	face evidence	mixed evidence
10	with	0.660	0.680	0.672
10	without	0.548	0.556	0.548
5	without	0.258	0.276	0.258
4	without	0.197	0.269	0.184

Table 4. Consistency probabilities of binary constraints calculated on different levels using different feature evidences.

8. Concluding Remarks

We investigated several aspects of the constrained tree search technique,that has been a robust technique for dealing with such problems as feature correspondence, image registration, and consistent labeling. An indexing scheme, inspired by the Dewey decimal notation, was originally proposed for the purpose of simplifying the implementation of the search algorithm. It later turns out that the resulting procedure characterized by the scheme behaved like a digital counter and can easily be realized in hardware.

From a theoretical point of view, sets of features and constraints selected for object recognition should satisfy both requirements of completeness and independence. However, from an empirical point of view, a small set of powerful features and constraints will be more adequate. Through an analysis of the complexity of the search algorithm, we developed a process, that can be used to select robust subsets of features and constraints. Experiments showed that the process is applicable. There are still problems remained to be solved, such as the ambiguities of interpretation, the implications of sensing error, as well as the applications on curved and articulated objects.

APPENDIX: The Constrained Tree Search Algorithm

Function : PathGenerator

Objective : Generates a path number from the current path number.

Input : PATH: the array contains the current path number.
POS: the index of array PATH indicates the digit to be altered.
BASE: the array contains the base number.

Output : PATH: the array contains the candidate path number.
MARK: the index of array PATH indicates the most significant digit modified.
COMPLETE: the flag informs the last path.

Procedure :

```
begin
    If (the current path PATH is the last path){COMPLETE = yes;
    return;}
    else COMPLETE = no;
    Increase the digit at POS of array PATH by one through and refer to
    the BASES array;
    MARK the most significant position of PATH where the digit is
    changed;
  end; (** end of procedure path generator **)
```

Function : PathCheck

Objective : Checks the legality of a path.

Input: PATH: the path array.
MARK: the highest position of the path array where the digit has been modified.
LENGTH: the length of the path.
CONSTRAINTS: constraint tables of features.

Output: POS:the position of the path array where the consistency check failed.
LEGAL: reports whether or not the path is a legal interpretation.

Procedure :

```
begin
    POS = LENGTH - 1;
```

```
        For (i = MARK to LENGTH) do
    begin
          Check consistency between the sensed feature and the model
          feature using the unary constraints of CONSTRAINTS;
          If (consistency fails) { POS = i; LEGAL = no; return; }
          For (j = i-1 downto 0) do
        begin
          Check consistency between pairs of sensed and model features using
          binary constraints of CONSTRAINTS;
          If (consistency fails) { POS = i; LEGAL = no; return; }
        end;
          Further check consistency among features using multiple constraints
          if available;
          If (consistency fails) { POS = i; LEGAL = no; return; }
    end;
    LEGAL = yes;
  end; (** end of procedure path check **)
```

Algorithm : Interpretation Tree Search

Objective: Looks for candidate interpretations.

Input: MODEL: model features.
SENSE:sensed features.
CONSTRAINTS: constraint tables of features.

Output: PATHS: candidate interpretations.

Procedure :

```
  begin
      LENGTH = number of sensed features; POS = LENGTH - 1; COMPLETE
      = no;Set up BASES array; Set up initial PATH array;
      While (COMPLETE is no)
      begin
          PathGenerator(PATH,POS,BASES,MARK,COMPLETE);
          PathChect(PATH,MARK,LENGTH,CONSTRAINTS,POS,LEGAL);
          If (LEGAL is yes) store PATH in PATHS;
      end;
    end; (** end of algorithm interpretation tree search **)
```

REFERENCES

[Bel57] R. Bellman, "*Dynamic Programming*", Princeton University Press, Prineceton,N.J. 1957.

[Bol82]R.C. Bolles and R.A. Cain, "*Recognizing and Locating Partially Visible Objects: the Local-Feature-Focus Method*", Int. J. Robotics Res. Vol.1, No.3, pp57-82, Fall, 1982.

[Bro84]R.A. Brooks, "*Model-Based Compute Vision*", UMI Research Press, Ann Arbor, Michigan, 1984.

[Che92]S.W. Chen, "*Computing Structural Constraints from a Set of 3D Geometrical Entities*", Bulletin of National Taiwan Normal University, Vol.37, pp.267-293,1992.

[Fis81]M.A. Fischler and R.C. Bolles, "*Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*", Graphics and Image Processing, Editor J.D. Foley, pp726-740, 1981.

[Gas84]P.C. Gaston and T. Lozano-P'erez, "*Model-Based Recognition and Localization from Sparse Range or Tactile Data*", Int. J. Robotics Res. Vol.3, No.3, pp3-35, 1984.

[Gri84]W.E.L. Grimson and T. Lozano-P'rez, "*Tactile Recognition and Localization Using Object Models: The Case of Polyhedra on a Plane*", IEEE, PAMI, Vol.6, No.3, 1984.

[Gri86]W.E.L. Grimson, "*Sensing Strategies for Disambiguating Among Multiple Objects in Known Poses*", IEEE J. of Robotics and Automation, Vol.RA-2 , No.4,pp196-213, Dec., 1986.

[Gri87]W.E.L. Grimson and T. Lozano-P'erez, "*Localizing Overlapping Parts by Searching the Interpretation Tree*", IEEE PAMI Vol.9, No.4, pp469-482, 1987.

[Gri88]W.E.L. Grimson, "*The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search*", ICCV 2, Pampa, Horida, pp218 -227, 1988.

[Gri86]W.E.L. Grimson,"*The Combinatorics of Local Constraints in Model-Based Recognition and Localization from Sparse Data*", Journal of the Association for Computing Machinery, Vol.33, No.4, pp658-686, 1986.

[Gri89]W.E.L. Grimson,"*On the Recognition of Curved Objects*", IEEE, PAMI, Vol.11, No.6, pp632-643, 1989.

[Knu73] D.E. Knuth The Art of Computer Programming: *Volume 1,Fundamental Algorithms*,2nd Edition, Addison-Wesley Pub. Company, 1973.

[Wal75]D. Waltz,"*Understanding Line Drawings of Scenes with Shadows*", The Psychology of Computer Vision, edited by P.H. Winston, McGraw-Hill Book Comp. New York, 1975.

供元件對應之限制性樹狀搜索法

陳世旺

國立台灣師範大學資訊教育系

摘　　要

在電腦視覺領域中，常會遭遇到兩組特徵元件對應的問題。而利用條件限制來作對應搜尋是一種處理這類問題相當有效的技術。本篇文章首先介紹一種索引架構，可以簡化上述搜索演算法的施行。這種索引架構，主要是起因於杜威的十進位符號觀念。同時由於我們所介紹的索引架構，搜尋演算法的行爲方式會如同數位式計數器，此種計數器可以很容易用硬體製成。本篇文章也對寫成的演算程式作複雜度分析。由分析的結果，我們提出一種程序，可以幫助我們來選取物體辨認時，較有效力的特徵元件及其相關的條件或限制。我們的實驗結果顯示上述搜尋演算法以及選取程序均具有實際應用的價值。