論　文

会話型識別木作成システム*

天満　勉** 花木真一** 吉田　博** 杉本　正***

Abstract

A man-machine interactive design tool called TBS has been developed, which, with a storage-type display and a keyboard, helps a human designer to build intuitively a decision tree for character recognition. A recognition tree has been formulated by distinguishing node data from procedures called feature-routines. The TBS node modification section works on the node data and allows insertion, deletion, and replacement of nodes in a recognition tree. The TBS recognition diagnosis section either allows to make a performance-test of an obtained tree or provides helpful information about a way of the selected features working on the actual data. In fact, three recognition trees have been built, one of which, KATAKANA tree, has a size of 2,000 nodes and achieved 98% of correct recognition rate. The TBS has been evaluated through experiences of building large recognition trees. Some future problems in intuitive design of a recognition tree are also discussed.

1. まえがき

文字などの図形の識別を，その幾何学的特徴に基づき識別木によって行う方法がある．複雑な幾何学的特徴を用いた識別木を自動的に設計することは現段階では無理であり，人間の直観的な認識能力を充分に活用した試行錯誤的な設計法がとられる．しかし，直観的に設計した識別木を実際のデータによって修正し，識別率を向上させて行く作業には，多大の労力と時間とがかかるため，従来，大規模で実用的な識別木を作成することは困難であった．

パターン認識への識別木の利用は，更に一般化した形で仮説—検証型のモデルとして CYCLOPS-1[1]に用いられており，図形の幾何学的特徴に基づく仮説の系列によって，画面解析を伴う強力な認識能力が示された．他方，パターン認識機械の実際的な設計のために人間の直観を活用して，識別対象パターンの性質に最も適した認識方式や，そのパラメータを実験的に決定する道具として，グラフィックス機能を備えたコンピュータを用いたインタラクティブ・システムの意義が強調されている．このような観点の道具は，IPACS[2]と総称されており，OLPARS[3]をはじめとするいくつかのシステムが開発されている．

筆者らは，遠隔描画装置上に書かれた手書き文字を識別する[4]ための識別木を，インタラクティブに作成する識別木作成システム[5](以下 TBS****と略称する)を開発した．

本報告では，まず，識別木を２つの部分に分けて定式化する．１つはプログラムの木状の流れを制御する節データの集合であり，これを論理部と呼ぶことにする．他は実際に図形の特徴を調べるプロシージャ[6]で，これを処理プログラム部と呼ぶことにする．このような識別木に，テスト，修正などの改良過程を会話形式で繰り返すことにより，複雑な識別木を比較的少ない労力と短い時間とで作成できる会話型識別木作成システム-TBS について述べる．TBS は，蓄積管型 CRT ディスプレイとキーボードとを備えた NEAC-3100 コンピュータに組込まれた．さらに，開発した TBS

---

* A Recognition Tree Building System using Man-machine Interaction by Tsutomu Temma, Shin-ichi Hanaki, Hiroshi Yoshida, and Sei Sugimoto (Nippon Electric Co., Ltd.).

** 日本電気（株）中央研究所
*** 日本電気（株）公害防止技術研究所

**** Tree Building Sytem の略．

を用いて，実際に片仮名，英数字，ひら仮名などの識別木を作成した．この経験を通じて，TBS の評価と，文字識別の問題に対する大規模な識別木を人間の直観を利用して設計する場合の問題点を検討する．

2. 識 別 木

識別木は節と枝の集合であり，1つの節から次段の節への枝は高々2本しか出ない2分岐木[6]を用いる．

各節を一定形式のデータとして扱い，次のような2つの型の節を定義する．

NODE-1 : (I, P, a, b, Y, N)

NODE-2 : (I, C)

ここで，I は各節に与えられる名前であり，名前 I を持つ節を，以下，節 I と呼ぶ．NODE-1 型は処理節と呼ばれる．処理節においては，P は節 I で用いられる特徴である．実際のコンピュータへの組込みにおいては，P は2つの要素から成っている．すなわち，p_1（特徴ルーチン名）と，p_2（p_1 に与えるパラメータ），の2つである．節 I で，パラメータ p_2 のもとに特徴ルーチン p_1 が実行されると，その結果，1個の正規化された特徴値 v が得られる．この特徴値 v に対して，$a \leq v \leq b$ の時，節 I の次に用いる節として，Y で示される節が，そうでない時，N で示される節が，それぞれ選択される．

NODE-2 型は終端節と呼ばれ，節の名前 I とカテゴリ名 C とで構成される．

コンピュータへの組込みでは，連続した数語をブロックとし，1個の節に1個のブロックを割当てる．1語 18 ビットの NEAC-3100 コンピュータに組込まれた節の形式を **Fig. 1** に示す．節ブロックを一定語長にするために，可変語長のパラメータ・ブロックを別に設け，p_2 をそのパラメータ・ブロックを指すポインタにする．

これらのブロックの取り扱いとメモリ管理の効率化のため，L[6],[7] を参考として，ML[6] と呼ぶルーチン群を開発した．ML[6] ルーチン群は，プログラマがメモリ領域（以下領域と呼ぶ）の場所と大きさを指定することにより領域内に自由に使える任意の大きさのブロックを確保し，これらを管理する．プログラマは，領域内に固定して取られるレジスタ群（バッグと呼ぶ）から発するポインタを用いて任意のデータ構造を作ることができる．

コンピュータで使用できる内部メモリの大きさに比して取り扱いたい識別木が非常に大きい場合には，内部メモリ内に識別木全体を格納することができない場合が起こる．このため，磁気ディスクなどの外部メモリに何枚かのページを用意し，識別木を分割して格納するようにした．ページは ML[6] の扱う領域と一致しており各々のページに 26 個のバッグがある．各ページのバッグのうち 16 個がそのページ内の独立した部分木の根を指すベース・レジスタ用に確保されており，各ページは最大 16 個までの独立した部分木を収容できる．

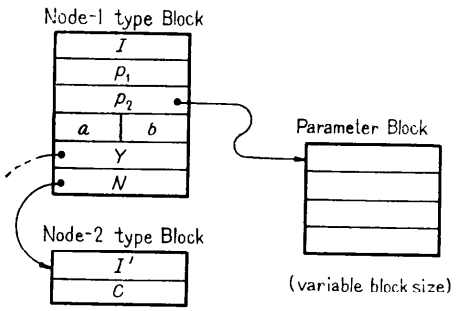

Fig. 1 Node implementation.

節を辿る機能は，識別木の処理において基本的に必要な機能である．ページ内にある部分木の節を上から下に辿る（下向する）のは，節内のパラメータ Y と N によるリンクで行われる．逆に，時としてページ内で下の節から上の節に辿る（上向）ことが必要になる．上向は節名 I の構造を利用して行われる．節名は TBS が自動的かつ一義的に生成する番号であり，ページ番号とページ内節番号とから成る．ページ内の節番号は次のように定められる．すなわち，節 m に対し，節 m のリンク Y で接続されている節の節番号を $2m$，節 m のリンク N で接続されている節の節番号を $2m+1$ とする．ページ内の部分木の根の節は 16 個のバッグのいずれかから接続され，節番号として 16～31 のどれかが与えられる．このような節番号の割当てによって，与えられた節からそのページ内で部分木の根に戻ることができる．根から，該当する節までリンクを辿って下向すれば，上向に必要な情報が得られる．

枝がページを渡る場合も下向は基本的にはページ内での方法と同様だが，上向は少し異なる．**Fig. 2**（次頁参照）に示した例によりページをまたがる枝について説明する．下向用リンクは，ページ L の節 E からページ M 内の空いているバッグ K を指し，K から部分木の根の節 F を指して形成する．上向用リン

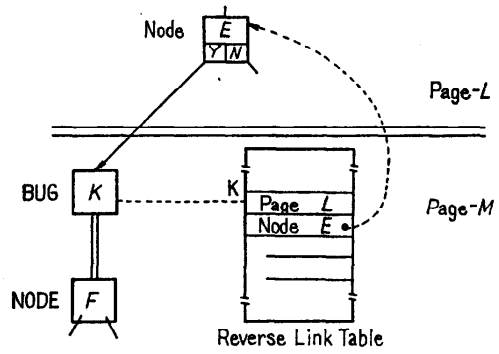

Fig. 2 Links between nodes in different pages, forward by a pointer and backward by a reverse link table.

クは，各ページにリバース・リンク表があり，ページ M 中の同表のバッグ K に対応する欄にページ L と節 E の情報が書かれて形成される．

識別木のある節において，本来分流することの望ましくない同一カテゴリの文字図形データが，相異なる枝に分流してしまうことが避けられない場合がある．このような場合に，先に相異なる枝に分流した同一カテゴリの文字図形データを，それぞれ，他のカテゴリの文字図形データと分離した後，合流させ，それ以後の処理を共通に行いたい場合がある．このような要求に対処するため，NODE-2 と同一形式で，カテゴリ名 C の代りにジャンプ先の節への道筋の情報を持つジャンプ節を導入した．分岐した枝が下方の節で再び合流するような構造は，通常の木の定義に反するが，ここでは拡張された木に対して，依然,「識別木」という呼び方を続けることとする．

3. 識別木作成システム：TBS

TBS は，節修正部と識別診断部の 2 部分から成っている．節修正部は，識別木の論理部を取り扱い対象とし，識別木を静的状態で修正するための部分である．これに対し，識別診断部は，論理部と処理プログラム部を総合して実際の図形データを作用させ，識別木を部分的にあるいは全体的に動作させて，その動作状態を観測するための部分である．

3.1 節修正部

節修正部では，識別木のうち現在扱っている節を含む部分木を CRT ディスプレイに表示し，操作者がキーボードから入力する節情報に基づいて識別木の論理部の作成および修正を行う．論理の作成および修正のための手続きとして，節挿入，節除去，節置換の 3 種

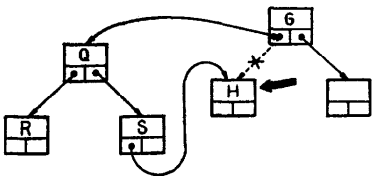

Fig. 3 An example of node insertion.

の機能がある．

Fig. 3 は節挿入の一例である．先ず，操作者がキーボードから節 H を指定すると，節 G から節 H へのリンクが切れ，代りに擬似節 Q がとられてリンクされる．続いて，節 Q の中の各要素 P_Q, a_Q, b_Q, Y_Q, N_Q の各データが TBS から要求され，操作者がキーボードからこれらのデータを順に入力する．さらに，節 R を形成するのに必要なデータ P_R, a_R, b_R および Y_R, N_R を入力した後，節 S のためのデータ P_S, a_S, b_S, N_S を入力する．操作者が挿入作業終了の合図を出すと，節 S の残った要素 Y_S に，最初に指定した節 H へのリンクが自動的に格納される．このようにして，節 G と節 H との間に，節 Q, R, S から成る部分木を挿入することができる．

節除去では，節名と次段の節へのリンク Y (または N) が与えられ，指定された節と，Y (または N) で指される節を根とする部分木が，識別木から除去される．節置換では，節の要素 P, a, b を個別に修正することができる．

Fig. 4, **Fig. 5** (次頁参照) は，論理の作成及び修正時に，CRT ディスプレイに表示される部分木の一例である．**Fig. 6** (次頁参照) は，作成した識別木の全論理をラインプリンタに出力した例の一部である．

Fig. 4 Displayed sub-tree in a rough view.

Fig. 5 Displayed sub-tree in a fine view.

Fig. 6 A part of printed-out tree.

3.2 識別診断部

識別診断部は，作成された識別木に対し改良すべき個所を発見するための補助手段を提供する．識別診断部には3種の機能を用意し，それぞれ，節トレース，データ・フロー，特徴ヒストグラムと呼ぶ．

節トレースは，論理の試行時，実際に通過する節ごとに算出される特徴値をラインプリンタに出力する機能である．識別木の設計者は，このトレース結果を調べて，入力図形データに対して識別過程の各節で使用された特徴の種類と特徴値の系列とを知り，誤識別を生じた分岐が識別木のどの節であったかを知ることができる．

データ・フローは，ファイル化されている多数の図形データが加えられた時，それぞれのデータが識別木のどの節を通過するかを調べて，各節を通過するデータの個数をカテゴリごとに表にして CRT ディスプレイ上に表示する機能である．表示の一例を **Fig. 7** に示す．設計者はこの表示から，図形データの分流が識別木の設計の意図通りに行われているかどうか，不都合がどの節に起因しているかを判断することができる．

特徴ヒストグラムは，指定した特徴に対し，用意したファイルから多数個の図形データを加えて得られる特徴値を横軸にとり，頻度を縦軸としたヒストグラムであり，CRT ディスプレイ上に表示される．表示の一例を **Fig. 8** に示す．設計者はこの機能を利用して，識別木の特定の節に採用しようとする特徴が適当であるか否かの判断を行い，同時に，その特徴を用いた場合に，目的とするカテゴリの図形データを他の図形データから分離するのに適した下限 a，上限 b の値を決定することができる．

Fig. 7 Data flow display.

Fig. 8 Feature value histogram display.

3.3 処理プログラム部

処理プログラムは，TBS の扱う対象およびその対象を表現するデータの形式に依存して大きく変わる．また，仮説をどのレベルで書くかによっても，その内容には大きな差が生じる．

筆者らは，データ・タブレット上に文字図形を描く際のペン先の座標系列をコンピュータに逐次入力し，この座標系列を圧縮して折線図形を得た[8]．この折線図形の表現のために，次の如きデータ構造を用いた．

すなわち，

(1) 折線の1つの線分に対し，1個の線分ブロッ

クを用意する．これらの線分ブロックは，つながれて線分リニア・リストを形成する．

（2） 各線分の端点に対し，1個ずつの点ブロックを用意する．点ブロックは，つながれて点リニア・リストを形成する．

（3） 線分ブロックと，その両端点を保持する2個の点ブロックとはリングを形成する．

（4） 連続するセグメントのうち，一筆で書かれる部分をストロークと呼び，線分ブロック中に印をつけて，その区切りを示す．

折線図形に対するこのデータ構造も，ML[6] ルーチン群を用いて取り扱われた．

このようなデータ構造を持つ折線図形に対し，識別木での仮説を，次のような幾何学的性質を用いて表わした．

- 与えられた2線分の実または仮想交点の位置．
- 1本または複数本の線分の全てを囲む最小の長方形で作られる領域の位置，大きさ，縦横比．
- 与えられた2線分の作る領域の相対位置．
- 与えられた2線分のなす角度．
- 与えられたストローク内の折れ．

このデータ構造を取り扱うための補助機能として，

- ブロックを指すポインタを動かす．
- ブロック内の特定フィールドの値を読み出す．
- 線分ブロックと同じ形の補助ブロックを作る．
- リニア・リストの各ブロックを辿り，指定されたフィールドを調べ，最大値，最小値，総和，平均値を算出する．

などがある．

これらを実行するのは，特徴ルーチンと呼ばれるサブルーチンであるが，識別木設計者の設定したいあらゆる仮説に対し，専用の特徴ルーチンを予じめ用意しておくことは困難である．そこで，設計者が特徴ルーチンを適宜組合せて，自分の設定したい仮説を表現できるような記述機能を設けた．

この記述法では，仮説はたとえば例1に示すような文字列として表現される．

例1 $*A=S(P(K(1),S),P(K(2),S))/$
$*B=V(5,*A)$

この文字列は，特徴ルーチンの1つである記述処理ルーチン PROG によって識別木の実行時に翻訳され，必要な他の特徴ルーチンを組合せて実行され，最終的に，この文字列の表現する仮説に対する1つの特徴値として出力される． 例1に示したような文字列は，TBS を用い，処理節での PROG 特徴ルーチンに対するパラメータとして，キーボードから与えられる．

一連の文字列は“／”で区切られるいくつかのステートメントから成る．各ステートメントは，

〈レジスタ名〉=〈関数〉

の形をしている．プログラマの使えるレジスタは 64 個ある．関数はいくつかの変数を持ち，関数自体も変数として使用できる．変数や関数の値は，数値そのものであるか，または，目的の数値を指すためにレジスタに入れられたアドレスかである．

例1の文字列は2つのステートメントを含み，次のように解釈される．まず，第1ステートメントの左辺の ＊A は，レジスタ A に右辺の値を入れることを意味する．右辺では，例1を説明するために文字列の下に付した番号の順序，すなわち，括弧の深いものから浅いものの順序に関数が評価される．各関数を評価した中間の値は，プログラマには隠されたレジスタに一時的に蓄えられる．例1の右辺の記述は，『第1ストローク 〈K(1)〉 の始点 〈P(K(1), S)〉 から，第2ストローク 〈K(2)〉 の始点 〈P(K(2), S)〉 へひいた線分 〈S (P (K(1), S), P(K(2), S))〉 により作成された線分ブロックのアドレス』を意味する．このアドレスがレジスタ Aに入れられる．第2ステートメントでは，レジスタ Aの指しているブロックの第5フィールドの値〈V(5, ＊A)〉が，レジスタBに入力される．線分ブロックの第5フィールドには，この線分の長さが入っている．記述処理ルーチン PROG には，例1に現われた K, P, S, V を含む 12 種類の関数が **Table 1** に示すように用意されている．

4. TBS による文字識別木の作成

TBS を利用して文字識別木を作成する手順の一例を述べる．

S1： 設計者が，文字の特徴に対する直観に基づいて識別木の初期案を作成し，CRT ディスプレイ上に表示してモニタしながらキーボードから入力す

Table 1 Twelve functions used in the PROG routine.

記号	機能	記号	機能
P	交点計算とポインタ移動	B	差の計算
S	線分ブロック作成とポインタ移動	X	数値の組の最大値
K	開始ブロックのサーチ	I	数値の組の最小値
M	領域ブロック作成	E	ブロック数の計数
V	ブロック中のフィールドの値	F	次のブロックへポインタ移動
A	和の計算	R	前のブロックへポインタ移動

る.

S2: 識別木の論理部を磁気テープにファイルし，必要があれば，識別木の図をラインプリンタに出力する.

S3: 磁気テープにファイルされている文字図形データを識別木に加え，正しい識別がされるかどうかの識別のテストを行う．識別を誤った文字図形データに対しては，節トレースの機能を利用して診断情報を収集する.

S4: 設計者は，診断情報に基づき，識別木の図上で修正すべき個所を検討する.

S5: TBS の節修正部を用いて，識別木の修正個所を入力する．S2～S5 の手続きを数回実行した後，手順S6 に移る．満足すべき動作が得られれば終了する.

S6: 識別木を大局的に見直すため，データ・フローおよび特徴ヒストグラムの機能を利用して診断情報を収集する．手順S4 に戻る.

遠隔描画装置上に書かれた手書き片仮名，英数字，ひら仮名をそれぞれ識別する識別木[8]を，実際に TBS を利用して作成した．このうち，片仮名識別木は最も規模が大きく，節の個数約 2,000 個に及び，訓練なしの筆記者 60 名に対し 98% の識別率を得た.

TBS によって作成された識別木の性質の概要を知る目的で，英数字識別木を対象とし，NEAC-3100 コンピュータを用いて，次のような計測を行った．まず識別木の全ての節を辿り，第 i 層の処理節の個数 n_i を調べた．結果は **Fig. 9** の曲線 A のようになった．最も長い枝は 29 層あり，n_i は山型の分布を示した．2分岐木を使用しているので，1つの処理節の下には2個の節がつくが，曲線 A から，第 12 層近辺より深い処では終端節が急激に増加することが判る.

各層の処理節が識別に寄与している程度を知るために，識別木の作成過程がかなり進み，文字図形データに対して終端節が充分に用意されているという仮定の下に，終端節のカテゴリを手がかりにした計測を行った．第 i 層に属する第 j 番目の処理節を根とする部分木を考え，この部分木中にあるカテゴリの総数を C_i^j とする．但し，2筆の A と3筆の A のように，書かれる時の筆数の異なるものは，別のカテゴリに数える．各層ごとの C_i^j の最大値 $\max_j C_i^j$ は，Fig. 9 の曲線Bのように変化する．この曲線から，悪い枝では，第 20 層近辺になっても最大7組程度のカテゴリを分離できていないことが判る.

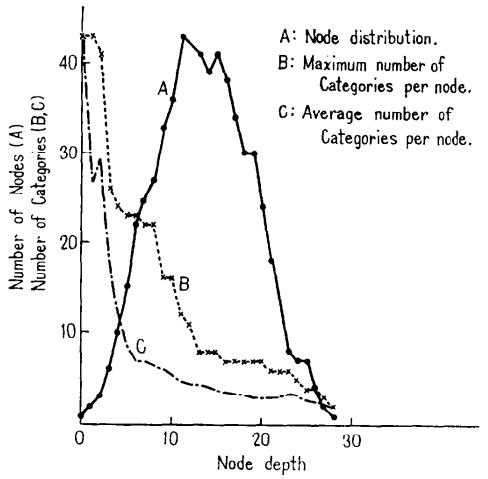

Fig. 9 Node distribution and number of categories in every node level measured for the developed Alpha-numeric recognition tree.

処理節の平均的な識別能力を知るために，各層の処理節の下につく平均カテゴリ数 $\sum_j C_i^j/n_i$ を算出して Fig. 9 の曲線 C に示した．もしも，各処理節ごとに，そこを通るデータのカテゴリを2分するような特徴系列が見つかれば，この曲線は，ほぼ指数関数 2^{-i} の形になるが，Fig. 9 の曲線 C はこれよりはるかに緩やかである．この原因の1つは，少ない個数の特徴を用いるだけで他のカテゴリから安定して分離できるようなカテゴリがいくつか存在することである．そして更に，人間が直観的に選ぶ特徴は，対象とする全カテゴリをほぼ2分するような一般的な特徴ではなく，ある特徴の強く意識された数個のカテゴリのみを，まず，他の多くのカテゴリから分離してくるような特徴であることも考えられよう．Fig. 9 は特定の識別木に関するものであり，今後いくつかの識別木について，このような計測を行ってみる必要がある.

5. TBS の評価と考察

分岐を基本とする木論理を用いる限り，ある節で，分離を意図しないカテゴリのデータを互いに反対側の枝に流してしまうことが起こり得る．これらを下方の適当な枝で再び合流させ，後の処理を1本にするためにジャンプ節の導入は有用であった．これによって，識別木格納のためのメモリ容量の不要な増大を避けることができた.

記述処理ルーチン PROG は，複雑な仮説の表現を可能にし，識別木設計の柔軟性を増した．PROG は

インタープリタであり，実行速度を遅くする要因とはなったが，この導入で得られた表現の拡張による利得は大きい．

識別診断部の3機能のうち，節トレース機能が最も有効に用いられ，識別木の部分修正のためのデータのほとんどが，この機能を利用して得られた．データ・フロー，特徴ヒストグラムは，共に，多量のデータを加えてみなければ精度が良くならず，反面，NEAC-3100の実行速度の制約のため，多量のデータに対しては時間がかかりすぎるというディレンマが大きな要因となり，あまり有効に利用できなかった．このような会話型の識別論理の作成システムにおいて，データ・フロー，特徴ヒストグラムに相当する機能の表示を短時間に行うためには，ハードウェアを一部併用して処理の高速化を計るとか，これらに代わる適当な診断機能によって代用する必要があろう．

TBS を利用することにより，特に識別木の論理部を修正する作業は極めて簡単になり，通常のプログラム形式で識別木を作成する場合[9]に比べて，格段に少ない労力と時間とで識別木を作成することが可能になった．

TBS を利用する場合に，大規模な識別木を一度に取り扱うことは，設計者が識別木全体のイメージを把えることを難しくする．このような問題を避けるためには，大規模な識別木を作る場合にそれを幾つかの階層に構造化し，各層の小さな部分木を TBS によって作る方法が考えられる．各部分木では，その入出力，すなわち識別対象文字のカテゴリの部分集合を予めはっきりと規定し，当該カテゴリを持つ文字図形データに対してはその部分木内で処理し，他の部分木に及ぶ影響を食いとめる．この場合の問題は，カテゴリの部分集合をどのようにして作るかである．1つの方法は，筆者らがこれまでに作ったような，設計者の直観に基づく一様構造の識別木の上位何層か，たとえば Fig. 9 で，処理節当りのカテゴリ数の曲線 C の折れ点付近までの層を第1の部分木とし，それらの出口に得られているデータのカテゴリ集合を，先のカテゴリ部分集合として採用することである．こうしてカテゴリ部分集合を固定したあとで，初めの部分木中で好ましくない分流が起これば，部分的な修正を行う．下方の部分木は，それぞれ固定されたカテゴリ部分集合に属する文字図形データに対してのみ識別を行うようにする．

もう1つの方法は，Blesser らが行ったような曖昧な文字図形を手がかりに定義した機能的属性 (functional attribute)[10] によって，互いに分離すべきカテゴリの組合せをデザインすることである．機能的属性の概念は，識別木設計の良い手がかりを与えるものと思われるし，特定の属性について，その属性を検出するような部分木を TBS を利用して実験的に作ることも，彼らの云う graphical context rule を見つけ出す意味で興味深い．

6. むすび

仮説検証型の文字認識方式において用いられる識別木を，人間の直観を充分に活用して設計するために，キーボードと CRT ディスプレイを用いた会話型の識別木作成システム (TBS) を開発した．TBS によって，実際に大規模な識別木を比較的容易に作成できるようになった．

TBS を用いて実際に識別木を作成した経験に基づいて TBS の機能の評価と，識別木作成時に起こる問題を述べ今後の課題を検討した．識別木を階層化し，小さな部分木に分けて作成するなどの設計手法を確立することは，それらのうちで大きな課題の1つである．

日頃御指導いただく中央研究所木地部長，有益な討論とデータ収集への協力を頂いた研究部各位に感謝する次第である．

参考文献

1) T. Marrill et al.: CYCLOPS-1: A Second Generation Recognition System, AFIPS Conference Proc. Vol. 24, FJCC, pp. 27～33 (1963).
2) L. N. Kanal: Interactive Pattern Analysis and Classification Systems: A Survey and Commentary, Proc. IEEE, Vol. 60, No. 10, pp. 1200～1215 (1972).
3) J. W. Sammon, Jr.: Interactive Pattern Analysis and Classification, IEEE, Trans. Comput. Vol. C-19, No. 7, pp. 594～616 (1970).
4) 花木真一他：遠隔端末からの実時間文字認識．情報処理，Vol. 11, No. 10, pp. 577～583 (1970).
5) 天満勉他：識別 Tree の Interactive な修正による片仮名文字の識別，電子通信学会研究会資料，PRL 72-126 (1973).
6) D. E. Knuth: The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Addison-Wesley, Mass. (1968).
7) K. C. Knowlton: A Programmer's Description of L^6, Comm. ACM, Vol. 9, No. 8, pp. 616～625 (1966).

8) S. Hanaki et al.: An On-line Character Recognition aimed at a Substitution for a Billing machine keyboard, Pattern Recognition, Vol. 8, No. 2, pp. 63～71 (1976).
9) S. Hanaki et al.: Real Time Character Recognition of Handprinted Characters, Proc. 2 nd Hawaii International Conference on System Sciences, pp. 345～348, Western Periodicals Co. (1969).
10) B. Blesser et al.: A Theoretical Approach for Character Recognition Based on Phenomenological Attributes, Proc. 1 st International Joint Conference on Pattern Recognition, pp. 33～40, IEEE, New York (1973).

(昭和50年7月22日受付)
(昭和51年10月12日再受付)