階層的パターンマッチングにおける簡易高速識別方法

5P-7

曽根広尚　加藤真　高橋弘晏
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

1.まえがき

文字認識を含むパターン認識の分野において、近年様々な方法が提案されている。特に漢字を含んだ日本語文字認識、音声認識の分野ではカテゴリが多い為、認識時間が長くなる傾向がある。これを解決する為に、階層的なアルゴリズムを用いたりハードウェア化による速度改善が多く提案されている。しかし従来の手法では階層的なアルゴリズムを用いると、識別プロセスが複雑になるためハードウェアが大規模なものになる傾向があった。

筆者らは階層的なアルゴリズムを実現しながら、非常に簡単で、ハードウェア化の容易な識別方法を考案試作したので報告する。

2.階層的パターンマッチングのモデル

最初に実行すべき階層的パターンマッチングの演算をモデル化する。

ここでは認識対象カテゴリが数千で、各カテゴリを記述する特徴ベクトルは、例えば3種類の特徴から構成され、各特徴はそれぞれL_1,L_2,L_3次元のベクトルであるとする。

入力パターンの特徴ベクトルとテンプレートの特徴ベクトルの構成をFig.1に示す。

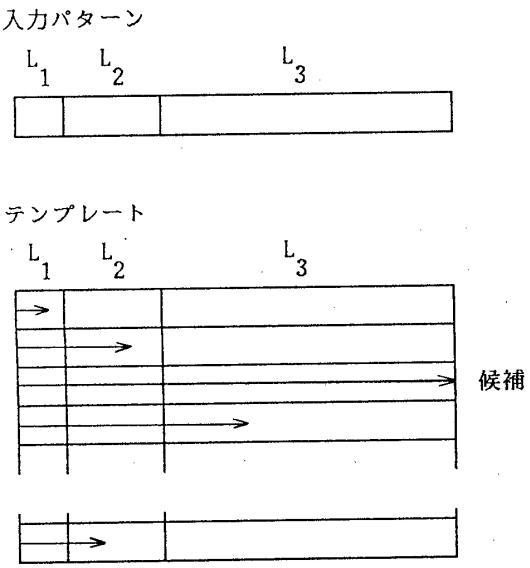

Fig.1　特徴ベクトルの構成

これらの特徴ベクトルが準備された後に実行される演算は以下のようである。まず入力パターンの第一要素と、テンプレートの第一要素間の距離$|I_i-T_i|$が計算され、L_1次元の特徴の閾値と比較される。ここで距離が閾値を超えていなければ、それぞれのつぎの要素間の距離が計算され距離のレジスタに加算され再び閾値と比較される。L_1次元の特徴に対する演算が閾値を超えることなく終了すると距離のレジスタがクリヤされL_2要素間の距離の計算が開始される。L_1の場合と同様にL_2次元の特徴にたいする演算結果が閾値を超えることなく終了するとL_3次元の特徴にたいする演算が開始する。さらにL_3次元の計算でも距離の加算結果が閾値を超えなければ、そのカテゴリは選択された候補として出力される。以上の過程において一度でも距離の加算結果が閾値を超えたならば、そのカテゴリに対する演算は打ち切られ、そのテンプレートの以後の要素についての演算は実行されず急速に候補をしぼりこむことができる。

ここで重要なことはどんな特徴を用いようと、同一のALUで処理できるように演算の形を統一化することである。ここでは一例としてシティブロック距離を用いたが、別の距離を用いる事も勿論可能である。

また認識対象がある特定のカテゴリに限定される場合(英数字のみ、カタカナのみ等)、テンプレートの配置を工夫することにより処理を高速化できる。例えばテンプレートの構成をそれを構成するカテゴリによりFig.2のようにしておくと、識別計算の実行をどこからどこまでと容易に指定することができる。ここで空白文字は簡便化のため数箇所に入れてある。Fig.2では漢字＋ひらがな、数字＋英字を指定する例を示している。

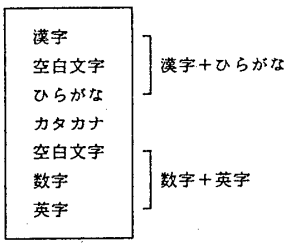

Fig.2 テンプレートの構成

A Fast Classification Method Using a Hierachical Recognition Algorithm
H.Sone　S.Katoh　H.Takahashi
Science Institute, IBM Japan,Ltd.

従来の手法では、辞書の中に文字種を指定するフィールドを設け、これを比較して指定された文字種かどうかを判定するものが多かったが、提案するシステムでは辞書自身には文字種フィールドはないし、特殊な判定ロジックもない。さらに従来の手法では、各特徴ごとにまとめて演算をおこなう(L_1ならL$_1$フィールドのみ)ため中間的な結果を保存するメモリが必要であった。しかし当システムでは、ひとつのL_1,L_2,L_3と順番に演算していくために、中間的な結果を保存する必要はない。

3.構成

2.のモデルで定義される演算を実行するハードウェアの構成図をFig.3に示す。

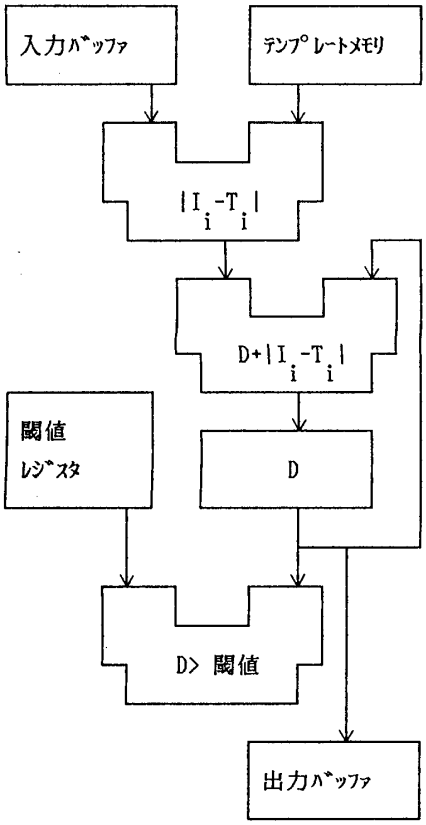

Fig.3 ハードウェア構成図

入力バッファとテンプレート　メモリが並列に読みだされシティブロック距離が逐次レジスタDに加算されていく。このときに32ビット幅（特徴ベクトルは8ビット幅）でメモリをアクセスし、距離計算のALUを4個用いて高速化を図っている。レジスタDの内容と、選択された閾値レジスタのうちのひとつがコンパレータにより比較され、閾値を超えていなければ演算が続行される。超えている場合には演算はそこでうちきられ、別のカテゴリとの間の距離計算がスタートする。

最後まで3種類の閾値を超えなかった場合にはそのカテゴリは選択された候補として出力バッファの中に距離とコードが書かれ、識別演算の終了後にホストプロセッサによりソートされ認識順位がつけられる。

テンプレートメモリのなかのどのエリアに対して演算を行なうかは、ソフトウェアによりセットされたポインタによりスタートアドレスとエンドアドレスが指定される。

ここで高速化のために用いた手法をまとめると

1)入力バッファとテンプレートメモリを並列にアクセスする。

2)メモリアクセスと距離計算をパイプライン的に実行する。

3)メモリへのアクセスを32ビット幅（特徴ベクトルは8ビット幅）で行なう。実質的に4倍の高速化が図れた。

4)階層的なパターンマッチングを行なう際に、演算の順序を最適化することにより中間的なバッファを不要とする。

5)テンプレートの構成を工夫することにより、テンプレート内に文字種を指定するフィールドを設けずに、文字種の指定を可能とする。こうすることにより、文字種判定の時間が不要である。

4.汎用プロセッサとの比較

本稿で提案するハードウェアでは、同じ演算を汎用16ビット　マイクロプロセッサMC68000のアセンブラ言語で記述した場合に比較して、100倍以上の実行速度が得られた（メモリアクセス速度は同じとして）。

5.まとめ

階層的パターンマッチングを高速に実行するアーキテクチャについて提案した。本稿で提案するアーキテクチャでは、階層的パターンマッチングを効率よく実行できるためハードウェアも極めて簡単に実現できる。この方式は認識に用いられる特徴(具体的には[1]の特徴を用いた)そのものとは無関係であり、様々な特徴を利用するシステムに組み込むことが可能である。識別に用いる距離としても、シティブロック距離とは別のALUを用意しておき選択的にそれをもちいることも可能である。

本稿で提案するハードウェアのゲートサイズは、テンプレート用、入出力バッファ用のRAMを除いて、約4800ゲート(2NAND)であり、現在のLSI技術をもってすれば容易に１チップ化できる規模である。

[参考文献]

[1] 高橋 "細線連結素方向による簡易手書漢字認識" 信学技報 vol.PRL82-8 pp.57-62,May 1982