2003－MUS－51 (2)
2003／8／4

オンライン手書き楽譜認識

宮尾 秀俊* 中村 隆**
*信州大学工学部
**富士通プログラム技研

あらまし 本報告では、タブレットのペン入力によって、通常の五線紙に手書きで記譜するのとほぼ同等の操作で音楽記号を入力することができ、さらに入力された記号を高精度に認識できるシステムを構築することを目的として次の手法を提案する。(1) 筆記された各ストローク記号に対して、DP マッチングとテンプレートマッチングを適用し、それらの認識結果を適切に組み合わせて認識する。(2) 認識記号群を音楽記号として確定できる段階で、効率良く、自動的に統合・出力する。これらの手法を使って実験を行った結果、ストローク記号について 96.7%の認識率、音楽記号については 95.5%の認識率を得ることができた。

キーワード 手書き楽譜認識、DP マッチング、テンプレートマッチング、楽譜エディタ

Online Handwritten Music Recognition

Hidetoshi MIYAO* Takashi NAKAMURA**
*Faculty of Engineering, Shinshu University
**Fujitsu Program Laboratories

Abstract In this report, a satisfactory method is described for the recognition of handwritten music symbols on a pen-based computer system. This system calls for two processes: (1) Recognition of handwritten stoke symbols using DP matching and template matching methods, and (2) Consolidation of the recognized symbols automatically when a music symbol could be produced. The experiment resulted in recognition rates of 96.7% and 95.5% for stroke symbols and music symbols, respectively.

Key words Handwritten Music Recognition, DP Matching, Template Matching, Music Notation Editor

1. はじめに

作曲、編曲活動では、五線紙に手書きで音楽記号を記譜していく方法がいまだに広く用いてられている。しかし、後に、楽譜の清書・印刷、編集、パートの分離、転調、自動演奏などを行うためには、コンピュータの理解できる形式の楽譜データに変換することが望ましい。

この変換を自動的に行う目的で研究されてきたのが、手書き楽譜の自動認識システムである。Bulis ら[1]は、音符画像の水平・垂直方向の黒画素の投影ヒストグラムを求め、それを標準テンプレートと比較することによって手書き音符の認識を実現している。Ng ら[2]は、音楽記号画像の骨格化を行い、水平線、垂直線、曲線に分解して認識を行い、さらに拍子や調に基づいた知識処理により、手書き音楽記号の認識を行っている。Watkins[3]は、音楽記号の要

素記号検出時にあいまい性の尺度を導入し、それを音楽規則に則った推論の中で用いることによって、音楽記号の認識をする手法を提案している。Yadid-Pecht ら[4]は、音符の検出にNeocognitron を用いて、音符へのノイズ、サイズ変化、位置移動、回転などに強い検出手法を提案している。これら、ほとんどの文献に記号認識率の記述がないので、実際の性能を客観的に比較・評価することはできないが、Yadid-Pecht ら[4]の手法では、音符についての認識率が 80%～90%であるとの結果が出ている。これより、オフライン手書き楽譜認識の認識率は現時点では低く、認識後の手動による修正作業に膨大な時間がかかり、実用的ではないことがわかる。

一方、直接、コンピュータに音楽データを入力する手法も種々開発・提案されている。特に市販のソフトウェアでは、マウス、コンピュータキーボード、鍵盤楽器を用いた入力法を採用しているものが一般的である。この方法は、複雑な楽譜も入力できる反面、コンピュータの操作に習熟している必要があり、使用方法を学ぶのにも時間がかかるという欠点がある。Anstice ら[7]の実験では、これらの方法でデータ入力した場合、手書き楽譜を描く場合の約 3 倍の時間がかかるという結果が得られており、入力時間にも問題がある。つまり、一般の作曲・編曲家にとって、これらの問題点が、コンピュータによるデータ入力より、手書きで楽譜作成することを選ぶ要因になっていると考えられる。

そこで、よりユーザーに使いやすい環境を提供することとデータ入力時間の短縮を目的に、ペン入力装置を使ったデータ入力方式が提案されてきた[5][6]。中でも、Anstice ら[7]、Ng ら[8]、永井ら[10]は、頻繁に出現する音楽記号については、ペン入力表記法を定めて(認識しやすい記号形状を選んでいる)、それに従ったペン入力をさせ、残りの記号はメニューから選択できるようにして、データ入力時間を大幅に短縮している。しかし、これらの方法では、各音楽記号に対応するペン表記法を覚えなければならないという問題がある。

本手法では、これらの問題点を解決するため、ペン入力の表記法を覚えなくとも、通常の五線紙に記譜するのとほぼ同等の操作で音楽記号をペン入力することができるシステムとし、さらに入力された記号を高精度に認識できるシステムを構築することを目的とする。そこで、筆記ストローク記号の認識精度向上のために、DP マッチングとテンプレートマッチングを組み合わせて認識する手法を提案し、さらに認識記号を音楽記号として効率良く、統合・出力する手法についても述べる。

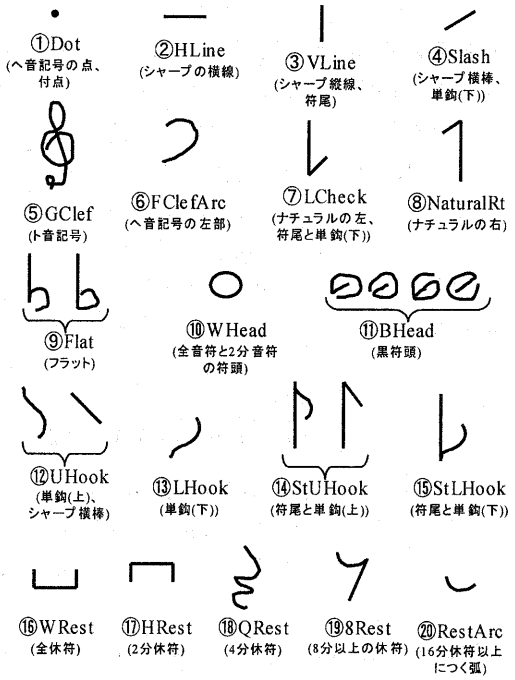

図 1　認識するストローク記号

2. ストローク記号とその入力法

ペン入力で認識可能な記号を図 1 に示す。これらの記号は前述のように、実際に手書き楽譜で表記される記号形状になるべく近くなるように設定した。ただし、黒塗りの符頭(黒符頭)については、筆記を簡略化するために、図 1⑪のように○印に斜線を引いた形状としている。また、各記号はペンのダウンからアップの一筆で描くことを条件とするが(よって、以後、この記号をストローク記号と呼ぶ)、筆記の始点、終点は拘束しないことにする。例えば 4 分休符(図 1⑱)を左上から書き始めても、右下から書き始めても対応できることにする。

これらのストローク記号を合成して生成可能な音楽記号は、ト音記号、ヘ音記号、シャープ、フラット、ナチュラル、連鉤を含まない音符全般、休符全般である。各ストローク記号が、どの音楽記号に属する可能性があるかを図 1 の各記号名下の括弧内に示した。ストローク記号群を音楽記号として自動統合する都合上(後述)、音楽記号単位で、それに属するストローク記号を続けて描く必要があるが、その中での筆記順には拘らないことにする。例えば、シャープを構成する 4 つのストローク記号は連続して描かなければならないが、その中での筆記順は拘束しない。

3. ストローク記号の認識

ペン入力によって描かれたストローク記号の筆記跡は 8 方向のフリーマンコードによって記録される。各ストローク記号について、フリーマンコードの標準テンプレートを用意しておき、それらとのマッチングを取ることによって認識を行いたい。長さの異なるパターン系列のマッチングには DP マッチングが適している。入力ストローク記号のコード列を$\{a_1, a_2, ..., a_I\}$とし、第 k 番目の標準テンプレートのコード列を$\{b^k_1, b^k_2, ..., b^k_J\}$とすると、DP マッチングにおいて、2 つのコード列$\{a_1, a_2, ..., a_i\}$と$\{b^k_1, b^k_2, ..., b^k_j\}$が最適にマッチングした場合の累積距離 $g_k(i, j)$は次式で計算される。

$$\text{初期値}: \begin{cases} g_k(0,0)=0 \\ g_k(i,0)=g_k(i-1,0)+1, for 1\le i\le I \\ g_k(0,j)=g_k(0,j-1)+1, for 1\le j\le J \end{cases}$$

$$\text{漸化式}: g_k(i,j)=\min\begin{cases} g_k(i-1,j-1)+d(i,j) \\ g_k(i-1,j)+1 \\ g_k(i,j-1)+1 \end{cases}$$

ここで、$d(i, j)$はコード a_iと b^k_jの間の距離を意味しており、同一コードの場合には値 0 を、異なる場合には値 2 を設定して計算している。上式を用いて、2 パターン間の最小累積距離 $g_k(I, J)$を算出することができる。更に求められた経路をバックトラッキングすることにより、経路長を求め、その距離で最小累積距離を正規化して、G_k を算出する。正規化した距離 G_kは 0 から 1 の値を持ち、小さな値ほど、2 つのパターンが似通っていることを示す。全ての標準テンプレートについて、上記の計算を行い、最小の値を持つ距離 G_{min}を求め、それに該当したテンプレートの記号をその入力ストローク記号の認識記号とする。なお、ストローク記号の入力始点を拘束していないため、始点と終点を逆転させたフリーマンコードについても、同様の計算を行っている。

ストローク記号の始点の位置によってフリーマンコードが大きく異なった系列になる場合がある。例えば、白抜きの符頭(図 1⑩)は、筆記の始点位置によって、異なるコード列が得られる。このような、記号については、DP マッチングは適しておらず、筆記軌跡を 2 次元画像としてテンプレートマッチングを行った方が良いと考えた。そこで、ストローク記号の筆記軌跡を太線化し、画像の大きさを幅 128 画素、高さ 128 画素に正規化して、2 次元画像化し、同様の方法で作成した標準テンプレート画像との単純類似度を計算する。入力ストロークの画像を $P(x, y)$、第 k 番目の標準テンプレートの画像を $Q_k(x, y)$とすると(P、Q_kの各画素値は太線化した線上部分を 1、そうでない部分を 0 とする)、2 つのパターンの単純類似度 S_kは次式で定義される。

$$S_k=\frac{\sum_x\sum_y P(x,y)Q_k(x,y)}{\sqrt{\sum_x\sum_y\{P(x,y)\}^2}\sqrt{\sum_x\sum_y\{Q_k(x,y)\}^2}}$$

ここで類似度 S_kは 0 から 1 の値で表され、大きな値を持つほど、2 つのパターンが似通っていることを示している。

全ての標準テンプレートについて、上記計算を行い、最も大きな類似度を示すテンプレート記号を認識結果とし、その時の類似度を S_{max}として記録する。

なお、点、水平線、垂直線(それぞれ、図 1 の①、②、③に示す)のストローク画像については、大きさの正規化をすると、ほぼ全ての画素が黒となり、同一の画像となってしまう。このため、これらの記号については、DP とテンプレートマッチングの前段階で、ストロークの外接矩形の幅と高さの情報に基づく識別を行っている。

次に 2 つのマッチングによって得られた認識結果が異なる場合、どちらの記号を採用する

かを決定しなければならない。実験を通して、マッチング手法により、識別の得意な記号とそうでない記号があることがわかっている。そこで、各手法の各記号に対する認識信頼度となる量を次のように導入することにする。

学習パターンとして、各ストローク記号を100個ずつ入力し、これらについての認識を行う。この結果に基づき、ストローク記号 k に対する認識信頼度 w_k を次式で定義する。

$$w_k = \frac{\text{正確に記号 k と判定された数}}{\text{記号認識全体で記号 k と判定された数}}$$

認識信頼度は 0 から 1 の値であり、大きな値ほど、その記号に対する認識の信頼度が高いと言える。DP マッチングとテンプレートマッチングの両手法について、この式を用いて、各ストローク記号に対する認識信頼度を求めておく。

いま、DP マッチングの認識結果として得られた記号を m とし、その記号に対する認識信頼度を $w_m{}^d$、テンプレートマッチングの認識結果記号 n に対する信頼度を $w_n{}^t$ とすると、

$$\begin{cases} (1-G_{\min})w_m^d \\ S_{\max}w_n^t \end{cases}$$

の 2 つの値を計算・比較し、下式の値が大きければ、テンプレートマッチングの結果(記号 n)を採用し、そうでなければ DP マッチングの結果(記号 m)を採用することにする。

4. 認識したストローク記号の統合

認識したストローク記号は、最終的に意味のある音楽記号に統合して出力する必要がある。簡単には、一塊の音楽記号を構成するストローク記号群を描いた時点で、手入力で何らかの命令を与えて統合すればよいが、これではわずらわしい操作が増えてしまい、使い勝手が悪くなってしまう。そこで、音楽記号として成立するストローク記号が揃った時点で、そこまでの記号列を音楽記号として統合、出力するようにしたい。しかし、音楽記号には、いくつかのストローク記号が揃えば、そこで確定ができる記号(以後、確定可能記号と呼ぶ)と、そうでない記号(確定不可記号)が存在する。例えば、ナチュラル記号は、ストローク記号 LCheck(図 1⑦)と NaturalRt(図 1⑧)が揃えば、そこで確定が可能だが、2 分音符は、和音として、いくつの符頭 WHead(図 1⑩)がつくかは入力の途中ではわからないため、確定ができない。よって、確定不可記号については、統合のタイミングを決める必要がある。そこで、次の方法で統合・出力をすることにした。

- ストローク記号群が確定可能記号に合致した場合は、その音楽記号を結果として出力する。
- 今まで入力された未統合のストローク記号群に対し、新しいストローク記号が図形的に十分に距離の離れた位置に入力された場合、それぞれを別の音楽記号に属する記号群と仮定し、前段までのストローク記号群の統合・出力を試みる。
- 上記で、図形的に近くても、未統合のストローク記号群に新しいストローク記号を加えると、音楽記号として成立しなくなってしまう場合は、前段までのストローク記号群の統合・出力を試みる。
- 統合を試みた結果、音楽記号として成立しないストローク記号群は消去する。

ここで、ストローク記号群がどの音楽記号と合致しているかを調べるために、各音楽記号に対し、構成するストローク記号列の組み合わせデータを用意している。データの構造は、例えば、ヘ音記号ならば、「FClefArc が 1 つ、Dot が 2 つからなる確定可能記号」、2 分音符ならば、「WHead が 1 つ以上、VLine が 1 本、Dot が 0 個以上からなる確定不可記号」のように構成している。描かれる記号の変形を許容するために、音楽記号によっては、複数の組み合わせデータを持つものもある。今回の音楽記号では、全部で 21 種類の組み合わせデータを使用した。

5. 実験結果と考察

本システムはパソコン(CPU : Pentium 4、1.8GHz、メモリ : 512MB)にタブレットを接続して構成し、プログラム開発は Linux 環境下でライブラリ GTK+、C 言語を用いて行った。実験では、標準テンプレートとして、DP マッチング用のフリーマンコードとテンプレートマッチング用の画像データ(128×128 画素)を、各ストローク記号について 1 つから 4 つ、合

計で 26 パターン用意して使用した。

各ストローク記号を 100 個ずつ入力した場合の各記号の認識率を表 1 に示す。表は、ストローク記号 8Rest(8 分休符)を例にとって説明すると、DP によって判定し、正解であった数が 67 個、間違った数が 2 個、テンプレートマッチング(表では TM と表記)によって判定し、正解であった数が 31 個、間違った数が 0 個であったことを示す。なお、比較のために、DP のみ、テンプレートマッチングのみを使って、それぞれの記号認識を行った場合の正解率も合わせて表記した。この表から、DP マッチングのみを用いて認識を行った場合の平均認識率が 92.6%、テンプレートマッチングの平均認識率が 72.2%であるのに対し、本手法の平均認識率が 96.5%(Dot、HLine、VLine の認識率は算入していない)となっており、両者のマッチング手法の良い部分をうまく統合できていることがわかる。特に、DP マッチングの認識率が落ちている記号 UHook や WHead についても、テンプレートマッチングの結果を使って、うまく補えているのがわかるだろう。

一方、認識信頼度導入の妥当性を検証するために、認識信頼度による重み付けをしないで、認識結果の統合を行った結果、認識率は DP マッチングの認識率とほぼ同等であった。この結果から、認識信頼度による重み付け操作の有効性も確かめられたと言える。

記号 Dot、HLine、VLine の結果を算入にいれた総合認識率は 96.7%であり、十分な認識率が得られたと考えられる。

次に、5 枚の楽譜、全 42 小節中に含まれる合計 200 個の音楽記号を、楽譜に倣って入力した結果、未検出記号数が 6 個、誤認識記号数が 3 個発生し、残りの 191 個(95.5%)の音楽記号については正確に認識・統合が行われた(音符、臨時記号については、五線上の位置も評価している)。上記 9 個の誤りについて確認したところ、同じ音楽記号に属するストローク記号であるにもかかわらず、記号の位置が離れて描かれたために分離してしまい、記号を消去してしまった場合が 1 件、単純にストローク記号の記号認識を誤ってしまった場合が 8 件であった。音楽記号認識率が 95.5%と高く、うまく認識できなかった記号はその場で簡単に消去・再入力が可能なので、誤認識部は、使い勝手にそれほど大きな影響を与えないと考えられる。

図 2 に本システムの実験結果の一例を示す。上部ウィンドウの五線譜にストローク記号を描き込むと、認識・統合処理後、清書された記号が下部ウィンドウに表示されるようになっている。

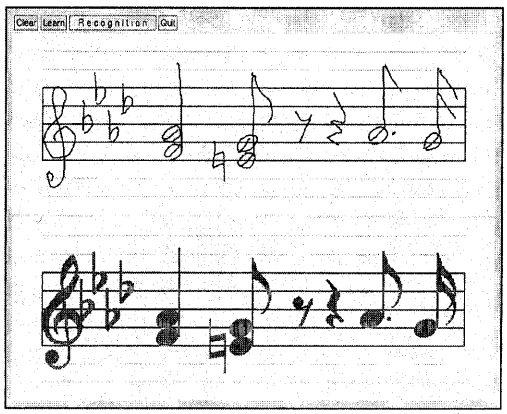

図 2　認識実験の実行例

6. まとめ

本報告では、ペン入力によって、通常の五線紙に手書きで記譜するのとほぼ同等の操作で音楽記号を入力することができ、さらに入力された記号を高精度に認識できるシステムを構築することを目的とし、DP マッチングとテンプレートマッチングを組み合わせて記号認識する手法を提案し、さらに認識記号群を音楽記号として効率良く、統合・出力する手法について提案した。実験の結果、各 100 個のストローク記号について 96.7%の認識率、200 個の音楽記号については 95.5%の統合認識率を得ることができた。これより、十分に実用に耐えられるシステムを構築できたと考えられる。

ただし、現在のシステムで扱えるのは基本的な音楽記号だけであり、今後は更に多くの記号の認識ができるように機能を拡張する必要があるだろう。特に音符の連鉤のように、複数の音符に影響を及ぼす記号をどのように扱うかが問題となる。また、一度確定した記号に対しても、自由に編集が可能な機能を追加する必要もあるだろう。

参考文献

[1] A. Bulis, R. Almog, M. Gerner and U. Shimony, "Computerized Recognition of Hand-Written Musical Notes" Proc. of International Computer Music Conference, pp.110-112, 1992.

[2] K. Ng, D. Cooper, E. Stefani, R. Boyle and N. Bailey, "Embracing the Composer: Optical Recognition of Handwrtten Manuscripts" Proc. of International Computer Music Conference, pp.500-503, 1999.

[3] G. Watkins, "A Fuzzy Syntactic Approach to Recognising Hand-Written Music" Proc. of International Computer Music Conference, pp.297-302, 1994.

[4] O. Yadid-Pecht, M. Gerner, L Dvir, E Brutman and U. Shimony, "Recognition of handwritten musical notes by a modified Neocognitron" Machine Vision and Applications 9 (2), pp.65-72, 1996.

[5] W. Buxton, R. Sniderman, W. Reeves, S. Patel and R. Baecker, "The Evolution of the SSSP Score Editing Tools" Computer Music Journal 3 (4), pp.14-25, 1979.

[6] K. Silberger, "Putting Composers in Control" IBM Research, Vol.23, No.4, pp.14-15, 1996.

[7] J. Anstice, T. Bell, A. Cockburn and M. Setchell, "The Design of a Pen-Based Musical Input System" OzCHI'96: The Australian Conference on Computer-Human Interaction, pp.260-267, 1996.

[8] E. Ng, T. Bell and A. Cockburn, "Improvements to a Pen-Based Musical Input System" OzCHI'98: The Australian Conference on Computer-Human Interaction, pp.239-252, 1998.

[9] A. Forsberg, M. Dieterich and R. Zeleznik, "The Music Notepad" ACM Symposium on User Interface Software & Technology, pp.203-210, 1998.

[10] 永井、天野、野口、松島、"オンライン手書き楽譜入力システムの試作" 情報処理学会研究報告音楽情報科学 MUS-47-11, p.59-64, 2002.

表1　ストローク記号の認識結果

ストローク記号名	正答数[個]			誤認識[個]		正解率[%]		
	前処理による判定	DPによる判定	TMによる判定	DPによる判定	TMによる判定	DPのみの場合	TMのみの場合	本方式
Dot	100	0	0	0	0			100
HLine	93	0	0	7	0			93
VLine	100	0	0	0	0			100
8Rest		67	31	2	0	98	68	98
BHead		98	0	0	2	100	98	98
FClefArc		61	36	3	0	93	78	97
Flat		97	0	3	0	97	61	97
GClef		100	0	0	0	100	4	100
HRest		17	83	0	0	100	88	100
LCheck		99	0	0	1	100	88	99
LHook		93	0	7	0	93	33	93
NaturalRt		87	13	0	0	100	64	100
QRest		84	3	13	0	84	42	87
RestArc		86	6	3	5	97	80	92
Slash		93	0	7	0	93	100	93
StLHook		95	0	5	0	95	76	95
StUHook		100	0	0	0	100	87	100
UHook		12	88	0	0	88	100	100
WHead		17	80	1	2	42	96	97
WRest		90	5	0	5	94	64	95
					平均	92.6	72.2	96.5